# Informatic Synthesis

## -2013 Summer-

イショティハドゥス

# 序

本書は、東京大学に入学された君たちが、英語一列をはじめとするあらゆるクソ期末試験を乗り越えた先に待ち受ける、「情報」というこれもまたクソ科目の期末試験に対して、なるべく素早く、そして的確に対応するために、情報の試験に出るところだけをなんとかわかりやすく解説したつもりのものである。

諸君の中には、情報の授業に出ていなかったと後悔している人もいるだろう。出なくても後悔していない人もいるだろう。大丈夫。このシケプリを大いに活用して、少なくとも 50 良、願わくば 90 優を取りにいってほしい。

第 I 部では、本題に入る前の傾向と対策を載せた。第 II 部以降では、教科の内容を順番に解説していき、第 IX 部（p.60〜）には練習問題をつけた。練習問題は、部ごとにわかれているので、各部を勉強した後の章末問題ならぬ部末問題として解くのも効果的だろう。

第 VI・VII 部の執筆や早い段階から制作に関わり校正を担当してくれたミヤダイ氏、過去問の解答の一部を引用させてもらった『only my information』の作者であるマシュー氏、ところどころ意見をくださり質問に請け合っていただいた東京大学大学院情報理工学系研究科コンピュータ科学専攻 須田礼仁教授、至高の参考書 Perfect Reader で英語一列の期末試験を助けてくれた ossan-arrow 氏、エロゲのキャラクターを語ってくれたさなきち氏、イショハラの被害届を東京大学ハラスメント相談所まで持ち込んだマサオ氏、精神面で支えてくれながら生命科学の宿題の答えを教えてくれたすみれっこ氏、そのほか数々の周りの不思議な方たちに、心から感謝いたします。

2013 年 7 月 29 日
イショティハドゥス

# 目次

# 第 I 部
# 本題に入る前に……

## 1　宙乃（そらの）ちゃんについて

筆者の座っている右側の壁には、我が愛する妹・宙乃ちゃんのタペストリーが貼ってあってとてもかわいい感じであるが、宙乃ちゃんについてご存知ない方も多いと思う。この後も何度か登場するので、一応ここで解説しておく。

宙乃ちゃんは、近衛ヶ原学園の 4 回生の女の子。（換算すると 16 歳くらいだと思う。）9 月 6 日のおとめ座。小惑星ファーレンフルスにあるエレウタリア王国の次期女王候補。地球に留学に来た兄を追いかけて、はるか遠くからやってくる。

人見知りな子で、家族以外とはほとんど面識がない。それでもお兄ちゃんのことが大好きな可愛い女の子。詳しくは `http://products.alchemist-net.co.jp/products/gackoh/` にて。

Lump of Sugar 作・学☆王 -THE ROYAL SEVEN STARS-に電話の声だけで登場するが、後に発売した PSP 版でその可愛い姿がお披露目。是非プレイして、その可愛い姿や言動を目にしてほしい。

## 2　試験の方式

共通問題と、教員ごとの個別問題にわかれている。試験時間は教員によるが、一般的には両方合わせて 90 分（共通問題のみのクラスは 60 分）。持ち込みは不可。計算用紙が 1 枚ついている。解答は、共通問題・個別問題両方を、1 行 30 文字くらいの罫線の、全科目共通解答用紙 1 枚（両面）に書くものと予想される。共通問題に関してはあとで詳しく解説する。

個別問題は、それぞれ担当の教員が作るが、教員によって問題がまったく違ってくる。一応個別問題にもある程度対応できるようなシケプリづくりは行ったが、完全なものは各クラスで毎年行ってもらいたい。

## 3　共通問題の傾向と対策

過去のことを知らずして、未来のことは語れない。

共通問題の作成は、毎年どの教員が作るかは不明である。しかし、情報科のなかで相談はしているはずであるので、あまり作成教員に関係なく、傾向や対策を練ることは可能であろう。

基本的に大問 3 つ構成。2007 年からは、大問 3 が A と B の二つに分かれ、選択形式となっている。本人によって得意分野が異なると思うが、基本的に A が社会・制度に関する記述モノ、B がアルゴリズムを考えさせる問題となっている。例外があり一概には言えないので、なるべくその場で両方の問題に目を通して選ばれたい。

以下、数年分の過去問の構成を解説する。各年のものに興味のない者は、下の表まで読み飛ばしてもらって構わない。共通問題の過去問は情報科公式サイト `http://www.edu.c.u-tokyo.ac.jp/edu/information.html` に掲載されている。解答は、マシュー氏著『only my railgun』を参照されたい [*1]。作成教師名は、個別問題のフォントや製作方法から推定したものであり、正しくない可能性がある。

[*1] 2006 年から 2010 年までの解答解説集。UTaisaku-Web に掲載されている。`http://todai.info/sikepuri/search/show.php?id=666`

## 3.1　各年の共通試験の概要

2006 年

大問 1　誤り訂正 4 問。記述式。量子化・符号化、インターネットの仕組み、暗号に関する問題。かなり容易。

大問 2　プログラミング。小問 4 つ。解答は数字 3 問、記述 1 問。書いてあるとおりに追っていけばできるが、自己参照型（再帰的）なのでめんどくさい。だが、計算の意味に気付けばラク。

大問 3　インターネットの仕組みからの出題。クライアントとサーバーの仕組みと、その具体例に関する記述。記述分量指定なし。

2007 年

大問 1　共通鍵と秘密鍵方式の暗号化に関する問題。基礎的だが仕組みを知っていないと解くのは無理だろう。

大問 2　インターネットの仕組み。階層構造と木構造。小問 3 つ。すべて記述式。階層構造の本質が理解できていればそこまで難しくない。

大問 3A　記述 3 題。分量指定なし。1 問目は法律だが、著作権法とか調べてれば十分書きやすい。2 問目は情報リテラシー。情報リテラシーをもっていれば解ける。3 問目は、GUI と CUI の違い。どれも常識的で非常に難しいというわけではない。

大問 3B　分割型処理の問題。数字を答える問題が 3 問、記述が 1 問。実際には中身は 2 進数なので数学的に、いや算数的に解ける。

2008 年

大問 1　階層構造を現実社会に当てはめさせる問題。「仮定が不足する場合は自分で補い」が「うわぁ」となりそうなところ。悪問臭がする（仮定によって答えが無茶苦茶にできる）。記述式。

大問 2　情報量に関する問題、というが、実際は確率を計算していくだけである。後半は面倒。もちろん情報量の定義は知っておかなければならない。7 問中 6 問が計算、1 問説明記述。

大問 3A　記述 2 問。1 問目は GUI と CUI の、特殊な場合における役割を問う（この場合は絵かき）。ちょっとした配慮がものをいう。2 問目は猥褻なロリ画像に関する問題。違う。不正アクセスの理解に関する問題。宙乃ちゃんかわいい。

大問 3B　プログラミング。命令集合がそのまま書いてあるので解きやすい。流れを追えば終わり。

2009 年　作成 田中哲朗

大問 1-1　量子化と標本化。説明記述 1 問、計算 2 問。計算が煩雑だが、問題自体は簡単。

大問 1-2　インターネットを利用した予約システムに関する文章の穴埋め。不要選択肢はない。教科書を読んでおけば（読んでなくても）解けるようなラクな問題。

大問 2　ボーリングの点数を計算するプログラム。問題文がマジで長い。問題はプログラムを書かせているため、白紙解答も多かったろう。時間があったら解けという感じの面倒な問題。

大問 3A　Google 図書館に関する問題。記述 5 題、各 2〜3 行。そこまで難しくないよね。

大問 3B　論理回路。真理値表の穴埋めと MIL 記法 3 題。MIL 記法をするのが結構めんどい。

2010 年　作成 山口泰

大問 1　ファイル操作を丁寧に追う。問題内でファイルを操作してるヤツにムカツキそうな難問。すげえウザい。もはや闇。木構造のデータモデルなどを把握しておきたい。

大問 2　誤り訂正と穴埋め。分野は文字コード、ビット数、GUI、著作権、暗号化と様々。語群が用意されているので難しくない。ただ悪問と言わざるを得ない問題がちょこちょこと。

大問 3A　前 2 問はクライアント・サーバーに関する記述。2006 年の第 3 問とほぼ同じ。明らかに怠慢。後 3 問はインターネットのチケット予約の仕組みの記述。ネタだけ昨年度と同じ。これは適当にどうにでもなる。宙乃ちゃんをかわいがっていれば解ける。

大問 3B　天秤で最も重い金貨を決めるアルゴリズムの計算量に関する問題。小問 2 つ。計算量のオーダーは適当にこんなもんだろーでも慣れていればいけるはず。時間との相談でしっかりやろう。

2011 年　作成 田中哲朗

大問 1-1　誤り訂正と説明記述。出題形式が少しヘン。分野は、インターネット・メール・DNS・公開鍵暗号方式の仕組みと、通信時間。通信時間の問題がとっつきにくかったが、それ以外は普通。2006 年の過去問の使い回しが見られる。

大問 1-2　プロトコルに関する問題。文章の穴埋めで、選択肢形式なので、なんとかなるでしょう。

大問 2　プログラミング。嫌らしいが、ゆっくり把握すれば解けるはず。コードを書かせる問題が 1 問と、変数値を求めさせる問題が 2 問。

大問 3A-1　GUI と CUI に関する文章を穴埋めする問題 8 問。選択肢は長文穴埋め（1 問）以外ない。そこまで難しい印象はない。

大問 3A-2　時事からの出題。かなり記述量が多いが、著作権と個人情報に関するものなので適当にかいていればよろしい。

大問 3B　情報量に関する問題。8 面体のサイコロがテーマ。なるべく情報量を小さくするにはどうすべきかを知っている必要がある。

2012 年　作成 福永アレックス

大問 1　ネットワークモデルに関する問題。ウジウジ頑張れば用語を答える問題以外はできる。

大問 2　プログラムと計算量。コードを実際に書かせる問題が 2 問、計算量のオーダーを調べる問題が 1 問。コードがかければ難しくない。

大問 3A　Winny に関する問題。記述が 4 題。うち 3 行程度が 2 問、5 行程度が 1 問。普通に勉強していれば解けるような問題になっている。

大問 3B　符号化の問題。小問は 4 つで、すべて数字で答える問題。だがしかし問題文が意味不明。

2013 年　作成 田中哲朗

大問 1　量子化・標本化と、ユーザインタフェースについての穴埋め。難しくない。

大問 2　立方根を求める 2 つのアルゴリズムに関する問題。回数を数えるのが面倒だが、アルゴリズム的には読み解きやすい。

大問 3A　チケット予約システムについての記述問題 5 つ。比較的簡単だが、細かいところまで説明しないといけない。

大問 3B　太郎君と花子さんがコインを投げたり情報を受け取りあったりしてイチャイチャする、情報量

についての問題。平均情報量をウダウダ計算すればいいが、グラフを書く問題に関しては、教科書 P.43 のグラフを知らないとキツかったかもしれない。

以上のような具合である。これを一覧にしてみたのが次の表である。

## 3.2　一覧表

| 分野 | 2013 | 2012 | 2011 | 2010 | 2009 | 2008 | 2007 | 2006 |
|---|---|---|---|---|---|---|---|---|
| 基礎 | | | | ○ | | | | |
| インターネット・情報システム | △ | | ○ | ● | △ | | ○ | ○ |
| データモデル | | ○ | | ○ | | ○ | | |
| プログラムとアルゴリズム | ○ | ○ | ○ | ● | ○ | ● | ● | ○ |
| 暗号 | | | △ | △ | | | ○ | △ |
| 標本化・量子化 | △ | ● | | | △ | | | △ |
| 情報量と確率 | ● | | ● | | | ○ | | |
| 論理回路 | | | | | ● | | | |
| ユーザーインターフェイス | △ | | ▲ | △ | | ▲ | ▲ | |
| 著作権 | | ● | | △ | ● | ▲ | ▲ | |
| 情報と社会 | | | ▲ | | | | ▲ | |

大問 1 つ分出ているものに○、小問のみでしか出ていないものには△をつけている。選択問題の 1 つである場合には塗りつぶした。

プログラム・アルゴリズム、インターネット・情報システムの出題が多い。また、著作権に関するものは選択問題に多いことがわかる。ただ、教師別問題で残りを埋めるように出ているので、全般的に勉強しておくほうが得だろう。どの作成者からも良問・悪問が出てくるみたいなので、もしかしたら作成教員はまとめているだけなのかもしれない。とまぁ、過去問分析はこんなところである。

ここから、分野ごとに、知っておいてほしいことをまとめる。これを読んでいる学生諸君は、一夜漬けかもしれないが、寝ないで頑張って読んでいってほしい。

# 第 II 部
# コンピュータと計算の知恵

この部では、プログラミングや論理回路など、コンピュータを構成する仕組みについて解説する。また、計算をするためのプログラミング・アルゴリズム技術についても解説する。得意にできれば大きな得点源となることは間違いないだろう。

## 4　プログラミング

内容としては、教科書第 5・6 章にあたる。いろいろ難しい用語だのデータモデルだの出てくるが、試験としては、プログラムのコードを与えられて、読み解け or 自分で書いてみろ、というものがほとんどである。毎年のように出ており、選択問題になっていないことも多い。したがって、これをマスターする必要はあるだろう。しかし、実際問題、感覚や慣れで決まるところがあるので、あまりに理解不能なら捨てても可。

プログラムは、試験では「擬似プログラミング言語」という、実際のものとはかけ離れたものを使うが、プログラミングができる人にもできない人にも、まあまあわかりやすい作りになっている。ただし、教員別問題のうち、藤垣のもので C 言語のものが出題されているが、この先生は毎年同じものを出しているので大して差し支えない。

すごく丁寧に説明しているつもりだが、それでもわかりにくいところがあるだろう。そういう時は、クラスに 1 人はいる、コミュ障で引きこもりがちなあのオタクや、やばいエロゲーマー、それか twitter 廃人に尋ねるとわかるかもしれない。諸君がこれを徹夜で読んでいる間、彼らは徹夜でパソコンの画面に向かっているに違いない [*2]。

### 4.1　変数と配列

プログラミングをする際の、基本的な概念。知っておこう。

#### 4.1.1　変数

変数というのは、値を入れる箱のようなもの。基本的には、値（プログラム中でコロコロ変わる）の名前だと思っておこう。

#### 4.1.2　配列

プログラミングをする際に、配列というものを使うことがよくある。とても扱いやすいからだ。たくさん（たとえば 100 個くらい）の大量の同じようなデータを扱うときに、いちいち変数を作っていたらキリがない。そういうときに、配列が役に立つ。

配列は、背番号のついた変数が、順番に並んでいるもの。その背番号を「添字」、変数の値を「要素」と表す。

配列の要素は $a_i$ のように表す。$a$ が配列の名前、$i$ が添字だ。

[*2] 彼らはきっと今何かに夢中に違いない。邪魔をしないようそっと尋ねるべきである。なお、この情報のシケプリを作った人は、今寝ているか、エロゲに没頭しているに違いないので、聞いても無駄である。あ、起きていたら twitter にリプライを送ると気づくかもしれない。

## 4.2　構文

プログラミング言語の表記の仕方の集まりのことを、構文（syntax）という。それについて一つ一つ解説していく。といっても、たったの 4 つ。その役割と計算の仕方をさっさと覚えて次に行こう。

代入 assignment

```
（変数）←（値 or 式）
```

これだけ。変数に、値（もしくは計算された値）を入れる作業。変数に値を入れると、それまでの値は消える。具体的には以下のように用いる。

```
p ← 宙乃ちゃん　　（変数 p の値を「宙乃ちゃん」にする）
y ← x + 14　　（変数 y の値を x + 14 にする）
```

ね? 簡単でしょう?

条件付き処理 conditional processing

```
if （条件）
  then （条件が成立した場合に行う処理）
  [else （条件が成立しない場合に行う処理）]
endif
```

こんな感じ。条件が成立しない場合になにも処理しない時は、else の行は要らない。（endif は要る。）具体例を見てみよう。

```
if（宙乃ちゃんを可愛いと思う）　　（宙乃ちゃんを可愛いと思うか。）
  then  俺が認める　　（思うなら、俺が認める。）
  else  お前とは分かり合えない　　（思わないなら、お前とは分かり合えない。）
endif
```

どうでしょう。わかりにくい? 実際の数字を使うとこんな感じ。

```
if (n > 10)
  then n ← n - 3
  else n ← n + 3
endif
```

$n$ が 10 より大きかったら、$n$ に $n-3$ を代入して、そうじゃなかったら $n$ に $n+3$ を代入する。たとえば $n$ が 5 なら $n$ は 8 になって、$n$ が 13 なら $n$ は 10 になる。それだけ。じゃあ下を見てみよう。

```
if（year を 4 で割った余りが 0）then
  if（year を 100 で割った余りが 0 でない）
    then うるう年と表示する
  endif
  if（year を 400 で割った余りが 0）
```

```
    then うるう年と表示する
  endif
endif
```

変数 *year* に年を代入すると、うるう年かどうかを判定するプログラム。理屈がわかればすぐにわかるだろう。then から続ける内容が多い時は、上の例の 1 行目のように、1 行に if〜then を書くこともあるが、同じ意味である。

反復処理 repetitive processing

```
while （条件） do
  （繰り返し行う処理）
done
```

こんな感じ。条件が成立したら、書いてある処理をする。その処理が終わってもう 1 回条件が成立するかどうかたしかめて、成立したらもう 1 回処理をする。またその処理が終わってもう 1 回条件が成立するかどうかたしかめて……。で、条件が成立しないなら、done まで行って、もう処理は行われない。実際に見てみよう。

```
i ← 1
n ← 5
while i ≤ n do
  a ← i² + n - i
  a を表示
  i ← i + 1
done
```

なんだか、プログラムのコードらしくなってきた。

まず、最初 $i = 1$ で $n = 5$ なのだから、$i \leq n$ が成立する。その後、$a = 1^2 + 5 - 1 = 5$ を表示する。表示し終わったら、$i$ が 1 増えて 2 になる。これが $i$ が 5 になるまで処理が行われる。最後は $i$ が 6 となって、処理が行われなくなる。

表示結果は「5 7 11 17 25」となるだろう。$i$ が 6 以上になったら do 〜 done の間の処理は行われない。だから、「a を表示」の行は 5 回しか行われないことになるのだ。

わかったかな?

解の出力

```
return 値
```

なんじゃこりゃ? と思うかもしれない。「答えですよ」っていうことを主張するコードである。実際には以下のように使う。

```
x の平方根を求める計算は次のように表せる。
y ← 0
d ← 0.00001
while (y + d)² < x do
```

```
    y ← y + d
done
return y
```

こんな感じ。$y^2$ が $x$ を超えないように、$y$ を 0.00001 刻みで大きくしていく、というシンプルなコード。一応計算してみたところ、$x = 2$ のとき $y = 1.41421$ となって終了した。もちろん「y ← y + d」の行は 141421 回実行されている。

構文はこんなもんである。ちょろいね。

なお、if〜endif の間や、while do〜done の間の行が字下げ（行頭が右側にちょっとズラすこと）されているが、これは単に見やすくするためのものである。必要というわけではないが、あとから理解しやすくするために、つけておこう。

## 4.3　計算の意味

実際の計算には、意味がある。意味のない計算をする人はあまりいない。したがって、コードから計算の意味を読み解く必要があるときがある。（他人のコードを引用するとかいうときには必須の事項であるから、プログラミングをする必要が将来ありそうな人たちには、是非理解してもらいたい。）

以下の例を見てみよう。

```
while n ≠ 0 do
  文字列 <s> に、文字列 <s> の直前に <n % 2> の値を付け加えたものを代入する
  n ← ( n - ( n % 2 ) ) ÷ 2
done
return <s>
```

おわかりいただけただろうか。いや、わからないってな。

n % 2 は「$n$ を 2 で割った余り」を意味する。（プログラミング言語ではよく使うので覚えておこう。Visual Basic だと n Mod 2 とかになるのかな。）

これは $n$ に数字をぶち込むと、2 進数表記になったものが $s$ として返ってくるプログラムである。

実際のところ、この分野は得意不得意、慣れ不慣れで結果がわかれてしまう。「計算の意味」に関して、理解できそうだな、もしくは理解したいな、と思う学生諸君は、2006 年の第 2 問、2008 年の第 3 問 B を解いておくといいだろう。センター数学Ⅱ·B のプログラミングの問題も有効である。（センターは言語が BASIC なのですこし構文が違うがそこは許してほしい）。

## 4.4　アルゴリズムの計算量とそのオーダー

アルゴリズムとは、その計算の方法のこと。その計算をするプログラムのコードのことだと思ってかまわない。

たとえば、以下のコードを見てみよう。

```
i ← 2
d ← 0
while i ≤ a-1 do
  if a % i = 0
```

```
    then d ← 1
  endif
  i ← i + 1
done
if d = 1
  then「合成数」と表示する
  else「素数」と表示する
endif
```

$a$ が素数かどうかを判定するプログラムである。この「計算量のオーダー」を求めることを考えよう。

計算量は大事である。パラメータによって計算量が変わってくるが、なるべく速いプログラムのほうが嬉しい。当たり前だ。オーダーは、計算量が、どんな値に比例するかを表す。これだけ。

具体的に見てみよう。計算量はだいたいループしている回数に比例する。$n$ 回実行される反復処理があれば、その計算時間は $n$ に比例する。$n$ 回実行される反復処理の中に、さらに $n$ 回実行される反復処理が入っている（while 文のなかにさらに while 文があるとか）なら、$n^2$ に比例する。この $n$ とか $n^2$ とかが計算量に比例する時、これが計算量のオーダーだ。$n$、$n^2$ 以外にも、$\log_2 n$ や $n!$、$2^n$、$\sqrt{n}$ などが答えとなり得るので、その場に応じて計算の特徴を見極めよう。

ちなみに計算量が $n^2$ に比例するとき、計算量のオーダーは $\mathrm{O}(n^2)$ と書くことになっている [*3]。

上の素数判定アルゴリズムの場合は、while の反復処理が $a-2$ 回実行されるので、計算量のオーダーは $\mathrm{O}(a)$ である。[*4]

では、下のようなアルゴリズムだとどうだろう。

```
i ← 2
b ← √a
d ← 0
while i ≤ b do
  if a % i = 0
    then d ← 1
  endif
  i ← i + 1
done
if d = 1
  then「合成数」と表示する
  else「素数」と表示する
endif
```

これは、while 文のループの回数が $\sqrt{a}-1$ くらい（実際は整数回なのでそれより少し少ない）なので、計算量のオーダーは $\mathrm{O}(\sqrt{a})$ である。

同じことを求めるプログラムでも、様々な計算方法（アルゴリズム）があるので、それによって、全然計算量が違う。なるべく計算量の少ないアルゴリズムを求める方法の調査が、情報科学科という闇の学科で、日夜行われている……。

[*3] これは単なる表現方法で、「計算量のオーダーは $n^2$ である」と言ってもまったく問題ない

[*4] $a-2$ に比例するんだから、計算量のオーダーは $a-2$ じゃないか! と思うかもしれないが、$a$ がめちゃめちゃ大きい時を普通は考えるので、定数項を無視して $a$ としてよい。一般的に、計算量のオーダーは定数項を無視してしまう。

とりあえず、どんな計算方法をしてて、ループが何重になってるかを把握すればいいのよ（適当）。

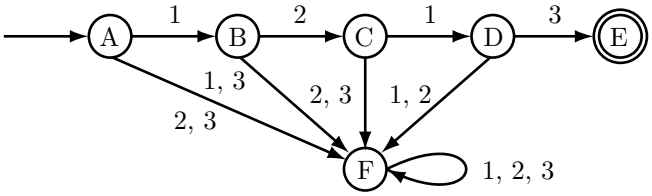
### 4.5　C 言語の構文

書こうと思ったが、突然出すことはないだろうということで、削除。C 言語やりたかったら、自分で「C の絵本」でも買ってやってみるといい。

## 5　有限状態機械（オートマトン）

ちょっとかっこいい名前が付いているこいつ。コンピュータの劣化版である。教員別問題では毎年のように誰かから出ているので、今年も誰からでるかはわからない。したがって、勉強しておく必要はそこそこあるだろう。

実態は遷移図を読み解く or 書くのが基本である。遷移図は以下のようなものである。




1, 2, 3 の 3 つのボタンがあって、「1213」と順番に押すと解錠する有限状態機械を表している。教科書 P.139 あたりに書いてある。

丸が状態、矢印が遷移を表している。どこからともなく矢印が引っ張ってあるやつが、最初に遷移している状態。この場合は A。ゴールは 2 重丸で表されたヤツ。この場合は E。

矢印は、たとえば「A の状態で、1 を押したら B の状態になって、2 か 3 を押したら F の状態になるよ」という意味を表している。この機械には、A〜F の 6 つの状態しかない。**状態**が**有限個**しかない**機械**だから、**有限状態機械**という名前が付いている。

逆に、メモリを持っていて、0 と 1 の情報を大量に保持できるものを、**チューリング機械**という。マス目状に区切られたテープ状のモノに、0 と 1 が順番に記録してあって、そのテープを右に動かしたり左に動かしたりしながら、有限状態機械を使って、それを読み取ったり、それに書き込んだりするというもの。[*5]

## 6　論理回路と論理関数

苦手な人は苦手だろう。ハッハッハ。大丈夫だ。キミならできる。コンピュータのすべての動作をつかさどる CPU の中には、この論理回路を施したトランジスタが入っている。だから、これがコンピュータの最小単位といっても過言ではない。

この章は、論理回路の種類から、回路の読み方・書き方について述べた後、それの応用である、半加算器・全加算器と、フリップフロップについて解説する。フリップフロップは**情報の分野としては重要ではあるが**テストではほとんど出ないので、読み飛ばしても構わない。

[*5] チューリング機械の仕組みなんてほとんど聞かれないので、この説明は適当に流してください。

## 6.1　論理回路の種類

なーに、AND と OR と NOT だけだ。他にもいろいろあるけど出ないらしい! この 3 つだけ!!!

なんか 2 つ（NOT だけ 1 つ）0 か 1 をぶち込んだら（入力したら）、0 か 1 かが返ってくるよ（出力されるよ）っていう感じ。これを表にしてまとめてみよう。

| MIL 記法 | 説明 |
|---|---|
|  | **AND　論理積**<br>両方 1 だったら 1 を返す。それ以外は 0。 |
|  | **OR　論理和**<br>両方 0 だったら 0 を返す。それ以外は 1。 |
|  | **NOT　否定**<br>0 と 1 を逆にする。0 なら 1 に、1 なら 0 になる。 |

ほらこれだけ。これ以外にも NAND、NOR、XOR、EQ などがあるが、基本的に試験ではこれらが何を意味するかは与えられている。

## 6.2　真理値表

真偽値表、真偽表ともいう。筆者はずっと真偽表だと思ってた。

入力の 2 つの値（0 or 1）の組み合わせと、その出力（返答）がどう対応してるかを表す表のこと。

たとえば下みたいな感じ。(2 つの入力を $x$ と $y$、出力を $s$ とした。)

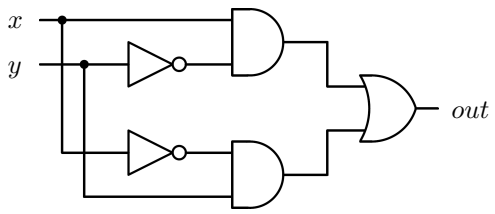
| $x$ | $y$ | $s$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

上は AND の真理値表になっている。OR とか NOT とかも書いてみてね（練習問題参照)。

## 6.3　論理回路の読み方

実際は、上の表にあったカマボコみたいなやつを線でつないでその回路を作る。その論理回路の書き方を MIL 記法とか言ったりする [*6]。

たとえばこんなヤツ。




[*6] MIL は Military Standard の略。アメリカ軍が物資の調達用に開発した規格である、MIL 規格のうちのひとつ（MIL-STD-806）だったのだ。現在では ANSI（米国国家規格協会： American National Standards Institute）などが同じものを定めている。

入力の $x$ と $y$ に対する出力 $out$ がわかればよい。

$x=0$、$y=0$ のときを試しに考えよう。上にある AND にとって、入力は 0 と 1 だから、上の AND が返す値は 0 である。同じように、下の AND にとって、入力は 1 と 0 だから、下の AND も返す値は 0 である。一番右の OR にとって、入力は 0 と 0（さっきの 2 つの AND から出てきたやつ）だから、$out$ は 0 になる。わかったかな?

これの真理値表は下のほうに書いといたので、自分で実際に書いてみてから確認してほしい [*7]。

## 6.4　論理回路の書き方

ある真理値表を与えられて、それを満たすような論理回路を書けるようになりたい [*8]。というわけで、その方法を見ていこう。

たとえば、下のような真理値表があったとする。2 つの入力を $x$ と $y$、出力を $out$ とした。

| $x$ | $y$ | $out$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

### 後ろの 2 人の関係は? 〜1st step〜

$x=0$ のとき、後ろの 2 つ（$y$ と $out$）の関係は、$out=\mathrm{NOT}(y)$ となっている。

$x=1$ のとき、後ろの 2 つ（$y$ と $out$）の関係は、$out=1$ となっている。

さて、これを加味して真理値表を下のように書きなおしてみよう。

| $x$ | $out$ |
|---|---|
| 0 | $\mathrm{NOT}(y)$ |
| 1 | 1 |

### 論理関数を書こう 〜2nd step〜

次のテンプレだけ覚えてしまおう。これだけ覚えていればなんとかなる。

$$\mathrm{OR}(\mathrm{AND}(\boxed{x=0\text{ のときの }out},\mathrm{NOT}(x)),\mathrm{AND}(\boxed{x=1\text{ のときの }out},x))$$

$\boxed{x=0\text{ のときの }out}$ と $\mathrm{NOT}(x)$ を AND でつないで、$\boxed{x=1\text{ のときの }out}$ と $x$ も AND でつないで、さらにその 2 つの AND の出力を OR でつないでしまえばいい。

今回はこんな感じ

$$\mathrm{OR}(\mathrm{AND}(\mathrm{NOT}(y),\mathrm{NOT}(x)),\mathrm{AND}(1,x))$$

---

[*7] 真理値表は左のとおり。

| $x$ | $y$ | $out$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

ちなみにこれは、論理回路のうち、XOR（排他的論理和）と等価。MIL 記法だと  になる。

[*8] 等価な論理回路はいくらでもあるけど、なるべく短いモノを作ろう。

だけど、$\mathrm{AND}(1, x)$ は $x$ と等価なので、実際は

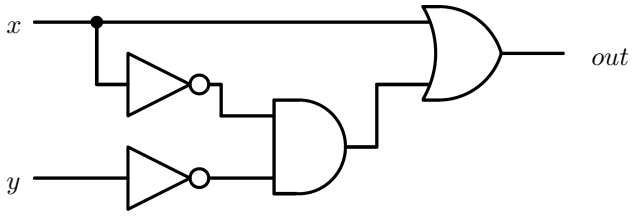
$$\mathrm{OR}(\mathrm{AND}(\mathrm{NOT}(y), \mathrm{NOT}(x)), x)$$

とすればよい。

回路を書こう ～final step～

上の論理関数をもとに、論理回路を書いてみる。手で書くとレイアウトが変になったりするので、一回下書きでどんな回路になるかを把握してから書こう。

実際に書いてみたのが下の図。同じ図になっただろうか。




入力が $x$ と $y$ で出力が $out$ の真理値表は $2^{(2^2)} = 16$ 通りあるが、{ AND, OR, NOT } だけで、すべての種類の真理値表に対応する回路を作れる。同じように、入力が $x$ と $y$ と $z$ で出力が $out_1$ と $out_2$ の真理値表は $(2^2)^{(2^3)} = 4^8 = 65536$ 通りあるが、これも { AND, OR, NOT } だけで、すべての種類の真理値表に対応する回路を作れる。

一般に、入力が $n$ 個、出力が $m$ 個の真理値表は、$(2^m)^{(2^n)}$ 通りあるが、それぞれのパターンに対応する回路を、{ AND, OR, NOT } だけで作ることができる。これを **{ AND, OR, NOT } は完備性をもつ**という。なお、先ほどちらっと紹介した NAND や XOR は、それ自体で完備性をもっている。つまり、NAND だけですべての真理値表に対応する論理回路を作れるのである。NAND すごい!

## 6.5　半加算器と全加算器

### 6.5.1　1 ビット [*9] 半加算器

半加算器とは、1 ビットの 2 進数同士の足し算を行う論理回路のこと。具体的には、$x$ と $y$、2 つの 1 ビットの 2 進数（0 か 1）の和 $x + y$ の答えの、1 の位を $s$、2 の位（下から 2 桁目）[*10] を $c_{out}$ として出力する論理回路のこと。

真理値表は以下の右表ようになる。

| $x$ | $y$ | $x+y$ |
|---|---|---|
| 0 | 0 | 00 |
| 0 | 1 | 01 |
| 1 | 0 | 01 |
| 1 | 1 | 10 |

| $x$ | $y$ | $s$ | $c_{out}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

さあ、これをもとにして、練習だと思って回路図を書いてみよう!

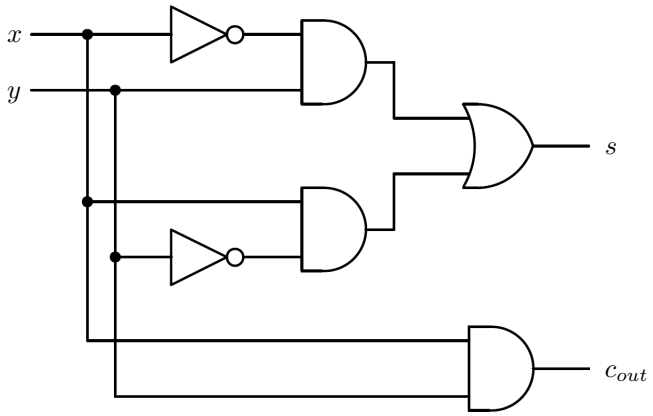
[*9] ビットというのは、何桁の 2 進数であるかを表す。8 ビットの 2 進数というのは、8 桁の 2 進数（0～255）である。

[*10] 10 進数の「7356」の「5」が 10 の位（くらい）であるように、2 進数の「1101」の「0」を 2 の位というし、16 進数の「A53D」の「3」も 16 の位という。同様に、2 進数の「1011」の「0」を 4 の位という。一般的に、$n$ 進数の、下から $k$ 桁目は、$n^{k-1}$ の位と言う。

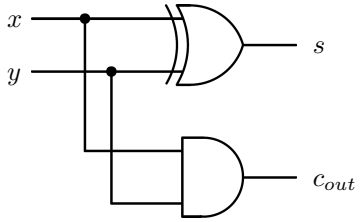
$s$ に関しては、$\mathrm{OR}(\mathrm{AND}(y, \mathrm{NOT}(x)), \mathrm{AND}(\mathrm{NOT}(y), x))$ になる。

$c_{out}$ に関しては、$\mathrm{OR}(\mathrm{AND}(0, \mathrm{NOT}(x)), \mathrm{AND}(y, x))$ つまり $\mathrm{AND}(x, y)$ になる。

これを合わせると、下のような感じになる。




$x$ と $y$ から $s$ を出力する回路に、実は XOR というものがあって、$s = \mathrm{XOR}(x, y)$ とかけるので、簡略化のため回路もこうする。すなわち、上と下の論理回路図は等しいものである。




### 6.5.2　1 ビット全加算器

（10 進数の）足し算を筆算でやるときのことを思い出してみてほしい。1 の位同士を足して、答えの 1 の位の求めるが、それが 10 以上だったら、その超えた分を 10 の位に繰り上がりさせる。10 の位同士と、その繰り上がってきたものを足したものが、答えの 10 の位となるが、それが 10 以上だったら、その超えた分を 100 の位に繰り上がりさせる。

……という風に計算しているだろう。2 進数でも同じである。

1 ビットの足し算をしていて繰り上がったら、次の桁に繰り上げる。逆に、**繰り上がってこられる桁の方は、それも加味して足し算をしなければならない**。

その「繰り上がってきたやつ」まで計算させる加算器が、全加算器だ。

その加算器が計算する桁（$n$ の位としよう）の値を、$x$、$y$ として、繰り上がってきたやつを $c_{in}$ とする。そしてその出力として、答えの $n$ の位を $s$、$n$ の位の 1 つ上の桁（$2 \times n$ の位）への繰り上がりを $c_{out}$ とする。

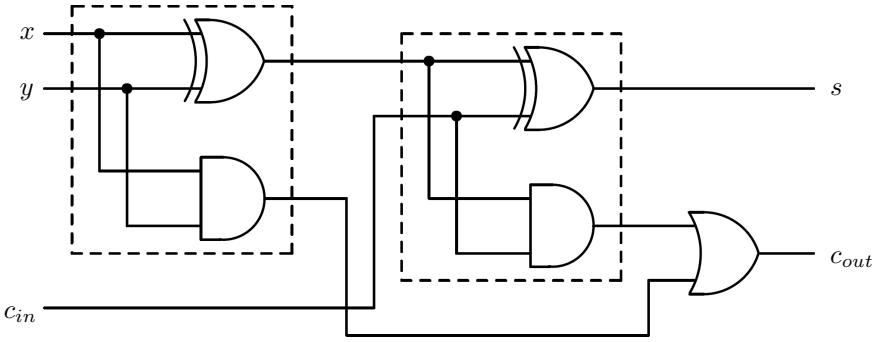
すなわち、半加算器の時の $c_{out}$ も、上の桁への繰り上がりだったのだ。

これで真理値表を書いてみる。結果的には $x$ と $y$ と $c_{in}$、3 つの 1 桁の 2 進数の和 $x + y + c_{in}$ の答えの、1 の位を $s$、2 の位を $c_{out}$ として出力する論理回路の真理値表となる。（右が真理値表）

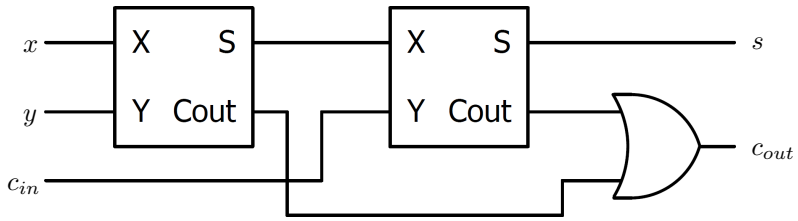
| $x$ | $y$ | $c_{in}$ | $x+y+c_{in}$ |
|---|---|---|---|
| 0 | 0 | 0 | 00 |
| 0 | 0 | 1 | 01 |
| 0 | 1 | 0 | 01 |
| 0 | 1 | 1 | 10 |
| 1 | 0 | 0 | 01 |
| 1 | 0 | 1 | 10 |
| 1 | 1 | 0 | 10 |
| 1 | 1 | 1 | 11 |

| $x$ | $y$ | $c_{in}$ | $s$ | $c_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

これの論理回路を書いてみよう! というのを最初からやるのも面倒くさい。そんなわけで、さっきの半加算器を利用して書くと、半加算器 2 個と OR だけで作れたりする。下図の点線の中が半加算器 1 つ。




半加算器がでかくてうっとうしいので、まとめるとこんな感じ。




これで、1 ビットの 2 進数の足し算は完成だ。これは覚えやすいし出しやすいので、個別問題で出る可能性のあるクラスは覚えておくと楽ちん。

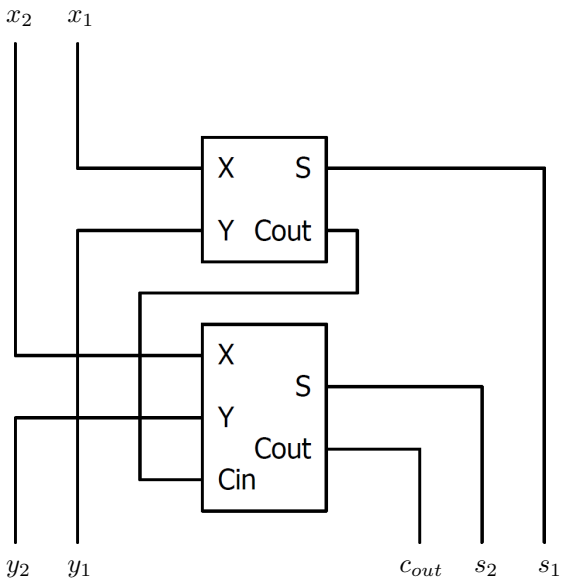
### 6.5.3　複数ビット加算回路

1 ビットだけではつまらないので、何ビットも計算できるように拡張してみよう。

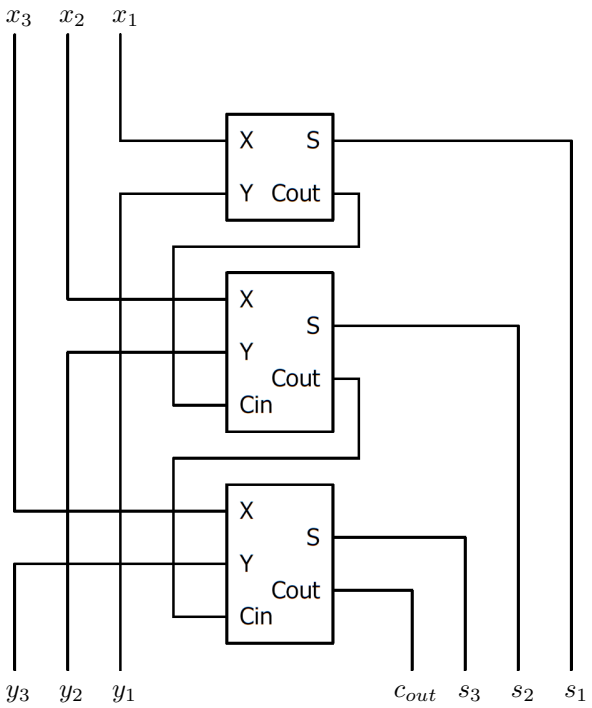
1 の位に関しては、何も繰り上がってくることがないので、半加算器でよい。2 の位以降は、1 つ下の位から繰り上がってくる可能性があるから、全加算器にしよう。

半加算器 or 全加算器からでてきた繰り上がり $c_{out}$ を、1 つ上の桁の計算をする加算器の $c_{in}$ につなげてやると、繰り上がりを処理できる。

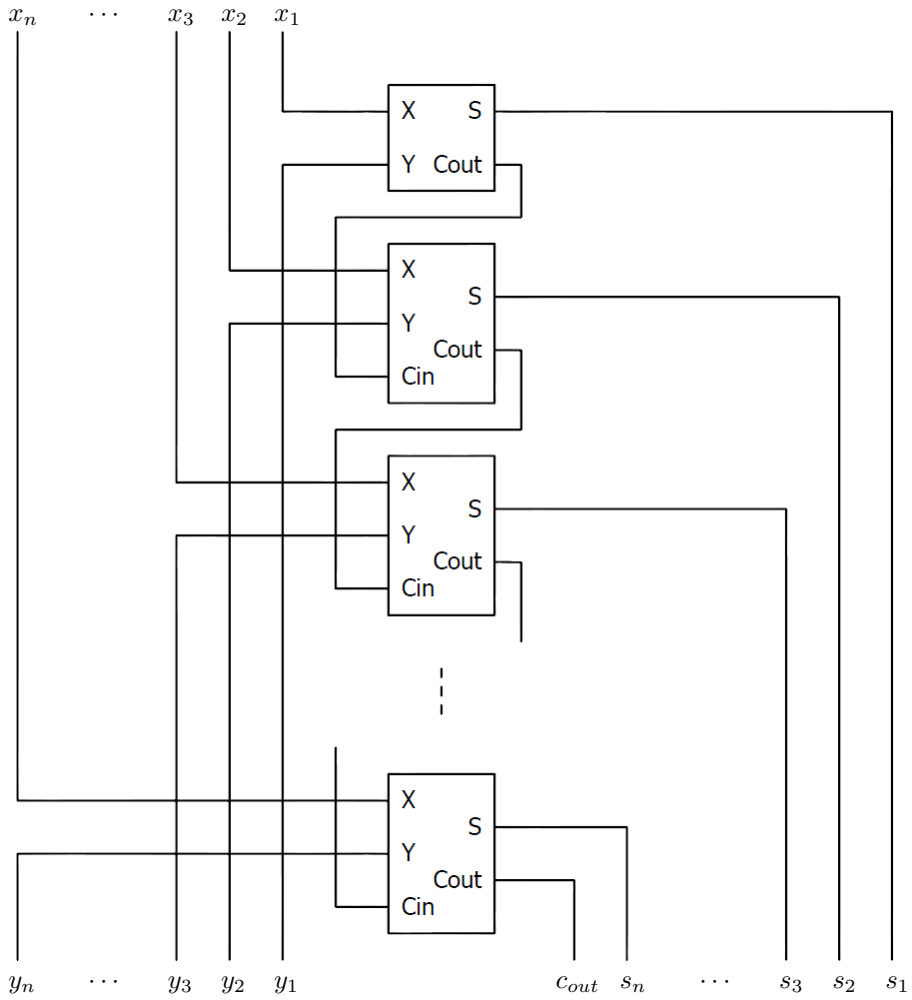
そんなかんじで組み合わせて、2 ビットの 2 進数同士の足し算をする論理回路は下図のようになる。1 の位が $x_1$ で 2 の位が $x_2$ の 2 進数と、1 の位が $y_1$ で 2 の位が $y_2$ の 2 進数の和が、1 の位が $s_1$ で 2 の位が $s_2$ で 4 の位が $c_{out}$ の 2 進数となる。(わかりづらい!! (笑)) 面倒なので、全加算器もまとめて表記した。




さらにくっつけて 3 ビットの加算器。




だんだんごちゃごちゃしてきた。これをたくさんつなげて、結局こんなかんじの回路になる。




これで、IC チップがある限り、何ビットの足し算もできる。

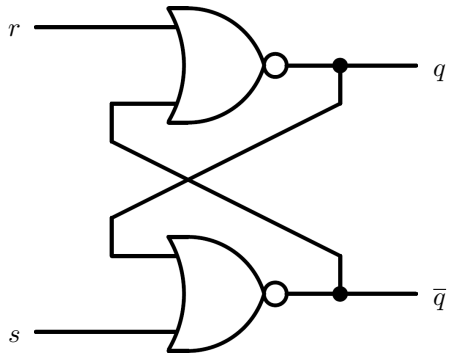
## 6.6　フリップフロップ

2013 年 5 月 22 日発売の豊崎愛生の 8th シングルです。関係ないです。

上の論理回路をうまくゴチャゴチャやれば、2 つの 2 進数の足し算をする機械は作れる。しかし、2 つの数の足し算だけじゃなくて、166 個の数の足し算をしたいかもしれない! 373 個の数の足し算をしたいかもしれない! そういう時に困る、というわけで、今まで足した分を記憶しておく領域を作ろうと考えたのだ。

簡単にいえば、一時的な記憶領域、メモリだ。

回路はこんな感じ。見たことあると思う。見たことなかったら、授業に出てないってことだね。




これが「非同期式 RS フリップフロップ」というもの。なんかループしてるけどよくわからない。
RS フリップフロップの特性表はこんなかんじ。

| $s$ | $r$ | $q_{i+1}$ | |
|---|---|---|---|
| 0 | 0 | $q_i$ | 状態保存 |
| 0 | 1 | 0 | リセット |
| 1 | 0 | 1 | セット |
| 1 | 1 | - | 禁止 |

さて、これもよくわからない。いや、授業受けた人にはわかると思うけど。

$s$ と $r$ が入力で、$q$ はメモリ（内部状態・保存されている値）を確認するモノだと思ってもらえればいい。$\overline{q}$ は $q$ の必ず否定になっているので、あまり考えなくていい。

$s$ と $r$ の両方に 0 が入力されたとき、メモリはそのまま変わらない（状態保存）。もともと 1 だったら 1 のままで、もともと 0 だったら 0 のまま。

$s$ に 0 が、$r$ に 1 が入力されたとき、メモリは 0 になる（リセット）。

$s$ に 1 が、$r$ に 0 が入力されたとき、メモリは 1 になる（セット）。

$s$ と $r$ の両方に 1 を入力すると、おかしなことになるので、この入力をしてはいけない（禁止入力）。

こういうこと。

ほかにもいろんなフリップフロップがあるんだけど、ぶっちゃけ試験に出ないからどうでもよくね!?
（ここ数年で、フリップフロップの問題が出たことはありません）

## 6.7　組み合わせ回路と順序回路・計算可能性

セクションの名前にそろそろ無理が出てきた気がするが気にしない。

それぞれの言葉の定義をしておく。

### 組み合わせ回路

組み合わせ回路とは、上の論理回路のこと。ただし、フリップフロップなどのメモリ機能をもつ回路を持たない回路のこと。半加算器、全加算器などがこれである。

### 順序回路

組み合わせ回路にくわえて、フリップフロップなどのメモリ機能をもつ回路を持つ回路のこと。

計算可能性

その計算が「どの機械で計算できるのか」ということ。単純な計算は論理回路でもできるが、難しい計算はコンピュータでないとできない。

一般的に、計算機のレベルの高さは、

組み合わせ回路 < 順序回路（有限状態機械） < コンピュータ（チューリング機械）

とされている。

有限状態機械の遷移図で表せる計算は、順序回路で実現できる。また、チューリング機械で行える計算は、コンピュータで実現できる、ということらしい。

# Coffee Break

お疲れさん。よくここまで読んだね。ちょっとコーヒーでも飲んで休憩しよう。

ところで、英語の文献って「ちょっと休憩」とか「コラム」の部分が「Coffee Break」になってるのあるけど、これコーヒーじゃなくてもよくないかな。コーラとかココアとかでもいいじゃんね。

いや、著者自身もつかれたのです。いちいち論理回路書くのタイヘンなのよ。著者は「水魚堂の回路図エディタ」で論理回路を書いてる。回路書いてキャプチャして TEX に画像ぶっこんでるだけ。$x$ とか *out* とかは、別に TEX 上で合成してる。わりと綺麗に出力できるので、TEX で論理回路書く必要があるときは試してみてね。

## どうでもいいプログラミングの話

まあ小さい頃からプログラミング言語自体は触ってたんだけど、なにせ Java だったんだよね。C は知らん、ってな。

なんか C++ が主流ですよね、C++ が書けなかったら雑魚ですよね、みたいな風潮あるけどツラい。ALESS よりはツラくないけど。

知り合いのプログラマーに、Java キライキライ言ってる人いて、ツラい。ALESS よりはツラくないけど。

わりと Java はオススメです。安全だし、わかりやすい。猿でもわかる。なんか入門書でも買ってやってみると、意外とハマったりするかも?

## どうでもいい進学の話

理学部情報科学科に行くと、いろんなアルゴリズムを学ぶことになるし、グラフ理論（巡回セールスマン問題とか)、計算量理論、離散数学とかも学ぶことになるらしいです。面白い人には面白いと思うので、興味があったら進学資料でも読んでみましょう。

同じ情報系でも、工学部電子情報工学科・電気電子工学科のあたりでもこういうことやるらしいです。プログラミングとか電子回路とかが楽しい人にはオススメ、かな。

疲れたよ。だけど、まだまだシケプリは続くよー! 次は情報量とか量子化とか!

# 第 III 部
# 情報とデジタルデータ

この部では、情報量やデジタルデータの符号化、量子化などのコンピュータ・データの理論について解説していく。あまり出題が多い分野ではないが、コンピュータに比べると断然わかりやすい。教科書の第 2 章にあたる。

## 7　情報量とは

### 7.1　情報量

$$-\log_2(\text{その情報の示す確率})$$

至ってシンプル。「log ってなんじゃらほい?」っていう人は、p.53 参照。

たとえば、「サイコロを 1 回振ったら 6 が出た」という情報は、$\frac{1}{6}$ の確率で起きるから、その情報量は $-\log_2 \frac{1}{6} = 2.58$ である。「ロト 6 を 1 口買ったら 1 等があたった」という情報は、$\frac{1}{6096454}$ の確率で起きるから、その情報量は $-\log_2 \frac{1}{6096454} = 22.54$ である。

こんな感じ。起こる確率が小さいほうが、情報量が大きいというのは、感覚的にも理にかなっている。

### 7.2　平均情報量

個々の情報量は定義できた。つぎに 1 つのメッセージ（1 符号）あたりが持つ情報量だ。

$$\sum \{(\text{情報量}) \times (\text{確率})\}$$

これだけ。例を見てみよう。

アルファベット 26 文字が、すべて等確率で出現するとすると、1 文字の平均情報量は、

$$\sum_{n=1}^{26} \underbrace{-\log_2 \frac{1}{26}}_{\text{情報量}} \times \underbrace{\frac{1}{26}}_{\text{出現確率}} = \log_2 26 = 4.700$$

となる。しかし、物事はそう簡単ではなく、アルファベットの出現頻度は、文字ごとにまったくちがう。それを表にすると以下のようになる。

| 文字 | 出現確率 | 文字 | 出現確率 | 文字 | 出現確率 | 文字 | 出現確率 |
|---|---|---|---|---|---|---|---|
| e | .118344 | r | .064540 | p | .020989 | k | .009289 |
| a | .087610 | h | .044135 | f | .020583 | j | .002132 |
| t | .084894 | l | .040176 | g | .018882 | x | .001827 |
| i | .073956 | d | .040150 | y | .018603 | q | .000863 |
| o | .073143 | c | .033146 | w | .017613 | z | .000812 |
| s | .072331 | u | .030633 | b | .017055 | | |
| n | .070225 | m | .027714 | v | .010355 | | |

これを使って 1 文字の平均情報量を計算しよう。

$$(-\log_2 0.118344 \times 0.118344) + (-\log_2 0.087610 \times 0.087610) + (-\log_2 0.084894 \times 0.084894) + \cdots$$

というふうに、ひたすらに情報量と確率をかけ合わせたものを足していく。結果は **4.1918** くらいになる。珍しい情報量が、珍しくない情報量に相殺されてこんな感じに情報量が減ってしまう。

こんな感じ。

# 8　情報と符号

## 8.1　符号化と平均符号長

やはり情報をコンピュータに扱わせるには、その情報を 2 進数で表示しなければならない。そんなわけで、以下のあ〜この 20 種類の文字を、それぞれ 0 と 1 の 2 種類の文字で符号にして、下の 100 文字からなる文字列 [*11] を符号化することを考えよう。

> かくここえけこくえきくかききうかくけきえうおけえこおあけおうけけききあかいおかここきかくこいあおこきあうこおこうくあうくくくおけおけうかくこいこけおいいけうああうくこいうくかいこかえけくこうおかうあけ

符号化の方法は様々にある。たとえば、一番シンプルな例。

| 文字 | あ | い | う | え | お | か | き | く | け | こ |
|---|---|---|---|---|---|---|---|---|---|---|
| 符号 | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 |

平均の符号長（1 文字に対する平均の符号の長さ）は、**4** だ。この符号に基づいて、上の文字列を符号化すると、

> 0101 0111 1001 1001 0011 1000 1001 0111 0011 0110 0111 0101 0110 0110 0010 0101 0111 ⋯

となる。(空白が入れてあるのは視認性のためであり、実際はいらない。)

ところで、それぞれの文字の出現回数を見てみよう。

| 文字 | あ | い | う | え | お | か | き | く | け | こ |
|---|---|---|---|---|---|---|---|---|---|---|
| 回数 | 8 | 7 | 12 | 5 | 10 | 10 | 8 | 13 | 12 | 15 |

けっこうばらつきがあることがわかる。なんとなくでも、「**出現回数の多い文字に長い符号を当てるのはもったいない**」と感じるだろう。

というわけで、可変長 [*12] の別の符号化を試みよう。今回は以下のようにしてみた。

| 文字 | あ | い | う | え | お | か | き | く | け | こ |
|---|---|---|---|---|---|---|---|---|---|---|
| 符号 | 11101 | 11110 | 001 | 11111 | 0001 | 1101 | 11100 | 100 | 1011 | 01 |

出現回数の少ない「あ」「い」「え」「き」などは符号の長さが 5、出現回数の多い「う」「く」「け」「こ」などは符号の長さが 2〜4 になっている。

---

[*11] 「文字列」というのは、複数の文字の羅列のこと。特にプログラミングとかだと頻繁に使う用語だし、この後も何回かこの表現を使うので、是非覚えてほしい。それが意味を持とうが意味を持たまいが、コンピュータには関係ない。

[*12] 割り当てられている符号によって、その符号の長さが異なること。一般に、固定長よりも効率的に符号化できる（と思う）。このような、可変長の符号を**エントロピー符号**といい、このような符号化をすることを**エントロピー符号化**という。対義語は固定長（符号の長さはみな同じ）。

これをもとにして、最初の文字列を符号化してみると、

1101 100 01 01 11111 1011 01 100 11111 11100 100 1101 11100 11100 001 1101 100 ⋯

となる。さっきと同じ文字の分だけ符号化したが、少し短くなっているのがわかるだろう。

ここで、平均符号長を以下のように定義する。(これは定義なので覚えましょう。)

$$\sum \{(\text{符号の長さ}) \times (\text{その符号が使われる確率})\}$$

これをつかって、先ほどの符号化の、平均符号長を求めよう。

$$\underbrace{5 \times 0.08}_{\text{あ}} + \underbrace{5 \times 0.07}_{\text{い}} + \underbrace{3 \times 0.12}_{\text{う}} + \underbrace{5 \times 0.05}_{\text{え}} + \underbrace{4 \times 0.1}_{\text{お}} \cdots = \mathbf{3.73}$$

さっきのが **4** だったので、それにくらべて **0.27 小さくなっている**のがわかる。

じゃあ、別の符号化を考えてみよう。[*13]

| 文字 | あ | い | う | え | お | か | き | く | け | こ |
|---|---|---|---|---|---|---|---|---|---|---|
| 符号 | 1111 | 0101 | 011 | 0100 | 001 | 000 | 1110 | 101 | 100 | 110 |

この平均符号長は、**3.28**(自分で計算してみてね)。さっきの 2 つよりも、もっと小さい。

ところで、この場合の 1 文字あたりの平均情報量を求めてみよう。平均情報量は、$\sum \{(\text{情報量}) \times (\text{確率})\}$ だから、

$$\underbrace{\left(-\log_2 \frac{8}{100} \times \frac{8}{100}\right)}_{\text{あ}} + \underbrace{\left(-\log_2 \frac{7}{100} \times \frac{7}{100}\right)}_{\text{い}} + \underbrace{\left(-\log_2 \frac{12}{100} \times \frac{12}{100}\right)}_{\text{う}} + \underbrace{\left(-\log_2 \frac{5}{100} \times \frac{5}{100}\right)}_{\text{え}} + \cdots \simeq \mathbf{3.259}$$

この平均情報量 3.259 と、平均符号長を比べてほしい。平均符号長の方が大きいことがわかるだろう。

**平均符号長は、**符号の決め方によって決まってくるが、**平均情報量より小さくなることはない。**これを、シャノンの**情報源符号化定理**(Shannon's source coding theorem)という。

## 符号化の方法に関する注意

テスト問題で「符号化しなさい」みたいな問題が出されることがある。

基本的には、**曖昧でないように**それぞれの文字(or 文字列)[*14] に対応する符号を決めて、割り当てた符号に基づいて、もとの文字列を符号化すればよい、という話である。ここで、「曖昧でないように」符号化しなければならないのだが、その「曖昧でない」とはどういうことかを説明する。

簡単にいえば、符号を読んでいったコンピュータ or 人が困らない符号にしましょう、ということである。

たとえば、以下のような符号割り当てを考えたとしよう。

[*13] この符号化は、もっとも効率的な符号化方法である「ハフマン符号化」を用いて符号化している。具体的な方法は、p.56 を参照すべし。

[*14] さきほどの例では「あ」〜「こ」の 1 文字ずつに対して符号を割り当てたが、さらに「きき」とか「くこ」などに符号を割り当てて、それを利用してもかまわない、という意味である。また、最初から「ああ」〜「ここ」の 100 種類に対して符号を割り当てて符号化することもできる。こうすると、もとの文字一文字あたりの平均符号長をさらに短くすることができる。

| 文字 | あ | い | う | え | お | か | き | く | け | こ |
|---|---|---|---|---|---|---|---|---|---|---|
| 符号 | 000 | 0010 | 011 | 0001 | 111 | 110 | 001 | 010 | 101 | 100 |

かなり効率的に見える。これを用いて、先ほどの文字列（かくここえけこくえきくかききうかく…）を符号化すると、

110 010 100 100 0001 101 100 010 0001 001 010 110 001 001 011 110 010 ···

となる。コンピュータ内では、空白で区切れができているわけではないので、詰めて書くと、

1100101001000001101100010000100101011000100101111 0010 ···

となる。読みにくいね。

これを、はじめから読んでいって、元の文字列に戻してみよう [*15]。

110　— か
010　— く
100　— こ
100　— こ
000　— あ
110　— か
110　— か

気付いただろうか。**元の文字列と違う!**

これは「000」が「あ」で割り当てられているのに、「000」から始まる符号が「あ」と「え」で 2 種類あるために起こる。コンピュータは、はじめから読んでいって「000」にあたったとき、それが「あ」の「000」なのか、「え」の「0001」の最初の 3 文字なのかが区別できないのだ。(同様のことが、「き」と「い」にも言える。)

すなわち、「**すでに使った符号」から始まる符号は、符号として使ってはいけない**のだよ、といういことである。逆に使ってしまって、元の文字列やデータにするときに、コンピュータの誤解を招くような符号化を「曖昧な符号化」という。

なお、p.56 に載せた「ハフマン符号化」の手順を覚えている人は、曖昧な符号化に関する配慮はほとんどいらない。ハフマン符号化で符号を作る際には、もともと曖昧な符号ができないような仕組みになっている。

# 9　アナログ情報のデジタル化

アナログデータである音声や写真や映像をデジタル化するにはどうすればいいか。それについて学んでいく。

アナログというのは、とても連続的だ [*16]。

例えば、虹の色を想像してほしい。赤、オレンジ、黄色、緑、青、藍と日本人は分けるが、その色は、7 色ではなく無数の色からなっている。しかし、コンピュータにはそれができない。16777216 色しか扱えないのだ。

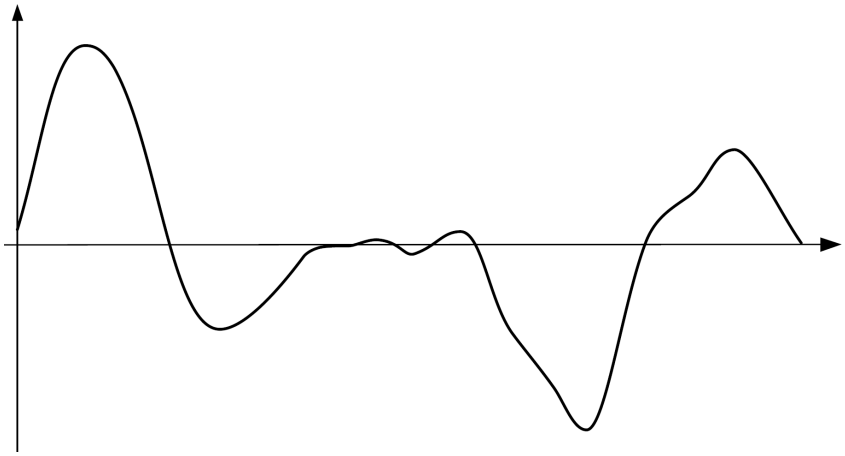
[*15] 復号化という。
[*16] というよりも連続的なものをアナログというのだが。

例えば、音の波形を想像してほしい。綺麗ななめらかな線で、いろんな周波数の音がたくさん混じって、素敵な音色を奏でている。しかし、コンピュータにはそれができない。22050Hz までの音しか扱えないのだ。

それが、どういうことなのか。アナログ量をデジタルデータにする作業の中でみていこう [*17]。

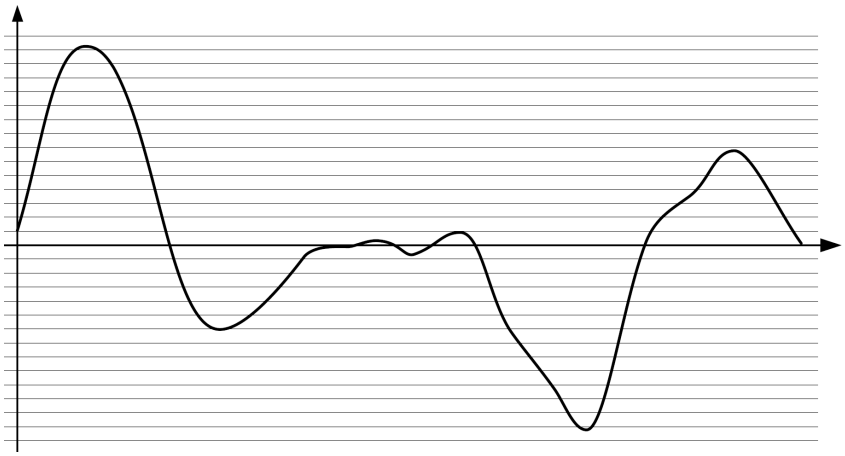
## 9.1　音楽データの場合

音は波である。以下のような波形のデータがあったとしよう。縦軸は音の強さ（音圧)、横軸は時間とする。



■量子化　量子化というのは、あるポイントでの連続的でアナログな値を、離散的 [*18] な値（整数）に変換（近似）することである。音のデータの場合は、その「ポイント」というのは、タイミングである。ある時刻での、音の強さを、整数値で表す。音楽 CD などでは、16bit の 2 進数で表現している。「16bit の深さで量子化する」などという表現を使う。

上の波形を量子化すると、下のようなイメージになる。



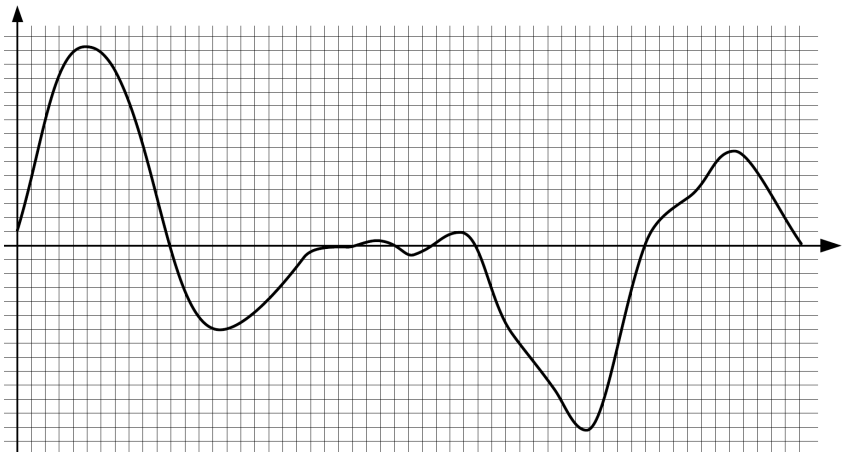
音の強さを、整数値で区切った。デジタルデータは、これよりも細かい音の強さの表現はできなくなる。

[*17] ここで紹介する以外にも様々な方法がある。フーリエ変換で周波数解析をして……などというやつとかだが、面倒だし難しいので説明は省く。

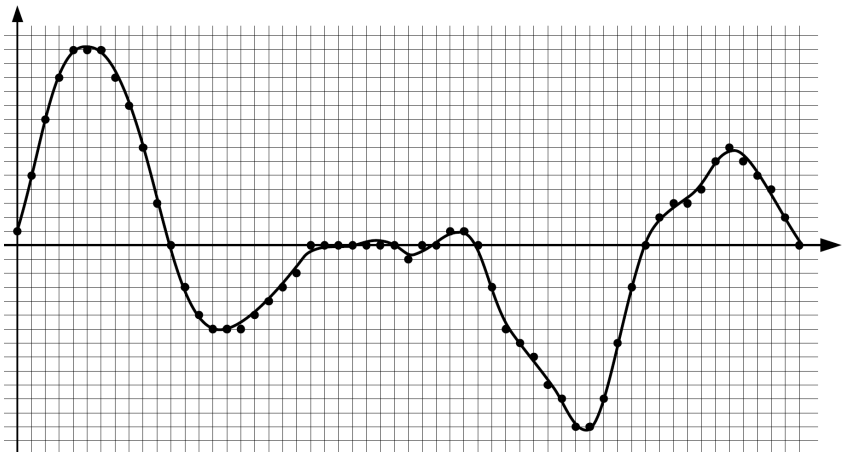
[*18] 連続的ではないこと。とびとびの値であること。整数とか。自然数とか。

■標本化　標本化というのは、量子化したポイントを作ること。音楽のデータであれば、何秒かごとに音の強さのデータをとることだ。音楽 CD などでは、1 秒間に 44100 回のペース（周波数 44100Hz）で、その音の強さを測ってデータ化している。

上の波形を標本化すると、下のようなイメージになる。



マス目状になったので、この波形を整数値で近似する（近いところの点をとって記録する）。イメージ的には以下の様な感じ。



周波数 44100Hz で標本化した場合、周波数 22050Hz 以上の音は、正しくデジタル化されない。この標本化の周波数の $\frac{1}{2}$ の周波数のことを、**ナイキスト周波数**という。

ナイキスト周波数以上のデータが正しくデジタル化されないために、歪みが起こることがある。この現象を**エイリアシング**とよぶ。

逆に、すべての周波数の音をデータにしたい場合は、一番周波数の高い音の、2 倍の周期で標本化すればよい。このことを**標本化定理**という。

人の耳は、子供でもだいたい 20000Hz の音までしか聞こえないので、44100Hz の標本化である程度十分といえる。

■データ量の計算　1 時間の音楽データを、16bit 44100Hz 2ch（ステレオ [*19]）で収録したとき、何バイトになりますか。という問題を聞かれることがある。その計算方法を説明しよう。

[*19] 左と右で、別の音を収録すること。左と右が完全に同じ音のものは、モノラルという。モノラルの音のほうが、かなり物足りないが、データ量的には 2 分の 1 になる。

単位時間あたりのデータを 16bit で表しており、それが 1 秒間に 44100 回。さらに、それが左側の音と右側の音の両方で行なわれているので、

$$16[\text{bit}] \times 44100[\text{Hz}] \times 60[\text{sec}] \times 60[\text{min}] \times 2 = 5080320000[\text{bit}]$$

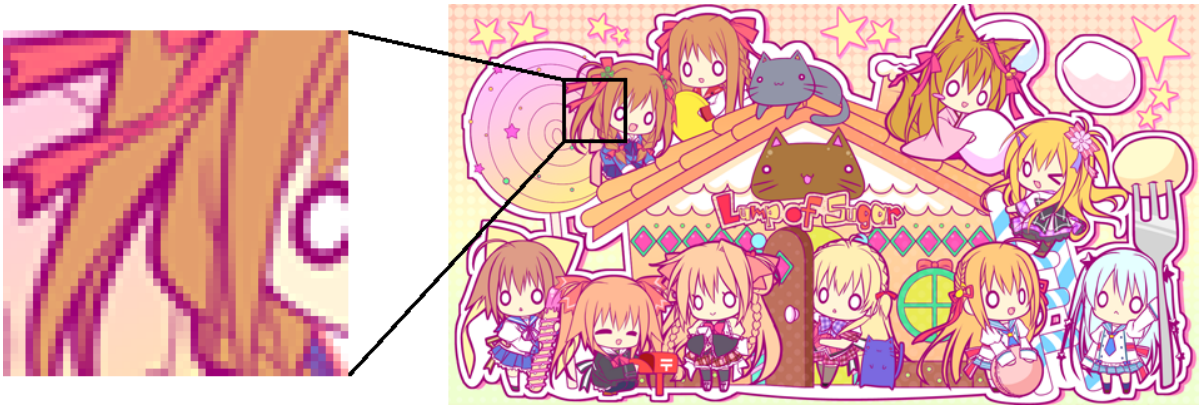
これを byte に直すと [*20]、

$$5080320000[\text{bit}] \div 8 = 635040000[\text{byte}]$$

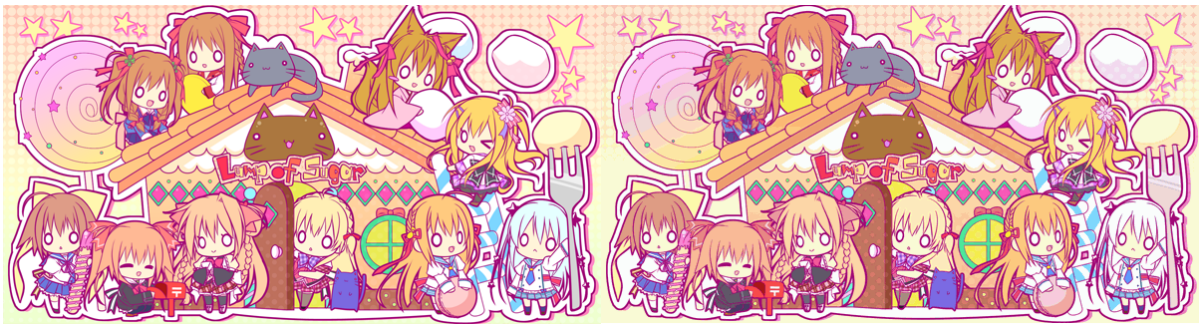
となる。実はこれがだいたい CD1 枚分の量になっている [*21]。

## 9.2　画像の場合

画像は下のように、たくさんの色のついた正方形のマス目が並んで、絵をなしている。



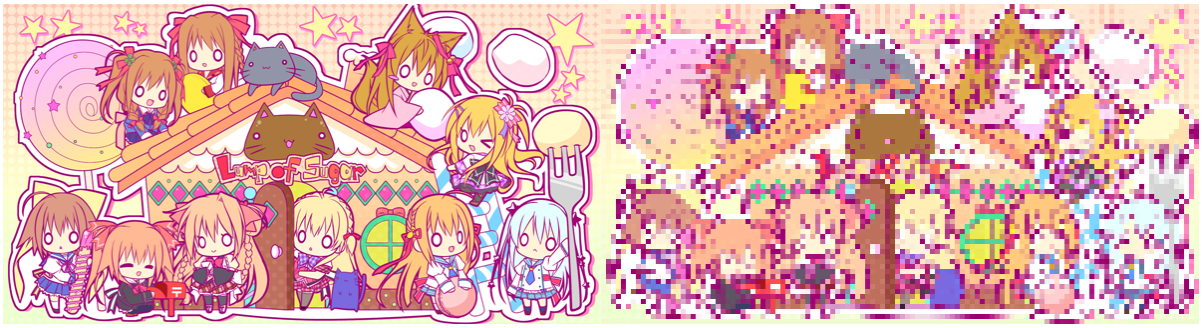
量子化は、絵の場合は「その正方形の色の種類」である。この絵だと、1 マスに赤・青・緑の光の三原色の強さが、それぞれ 0〜255 の 256 段階（8bit）で指定できるので、合計で $256^3 = 16777216$ 種類の色を表現することができる（深さ 24bit）。人間の目の色の識別能力は、だいたい 1500 万色が限界なので、ちょうどよいくらいである。この色の数を、全部で 256 色（8bit 量子化）にすると、以下のように少し色に鮮やかさがなくなる。（左が 24bit 量子化、右が 8bit 量子化。キャンディの部分のグラデーションが汚いのが目立つ。）



標本化は、元のアナログデータを正方形に区切ることである。標本化が細ければより繊細な画像に、標本化が粗ければより粗い画像になる。画像の場合は、標本化の細かさを「解像度」や「画素数」などとよぶ。（右は左の 20% の解像度。右の絵の方が 5 倍粗い。）

[*20] データ量の単位や、単位の変換がわからない人は、p.53 参照。

[*21] 第九（ベートーベン交響曲第 9 番ニ短調 op.125「合唱付」）が、だいたい全部で 60 分で、これをすべて収録できるようにしよう、ということで、CD はこの規格になったといわれている。CD の容量は聞かれることが多いので、だいたい 650MB と覚えておこう（情報系の常識）。



画像でのエイリアシングは、上の図のように粗い標本化を行った時に「粗い」と感じるそれそのもの。

上記のように、アナログなものをデジタル化すると、標本化や量子化という作業で、情報が失われる。しかし、アナログデータにはノイズが必ず含まれているものなので、デジタル化してノイズが取り除けている場合などには、情報が失われた、とは必ずしも言えないのもまた事実である。

サンプル画像について

サンプル画像は Lump of Sugar スマートフォン向け公式サイトより（`http://www.lumpofsugar.co.jp`）。絵は（たぶん）奏ナコト。©Lump of Sugar

# 10　文字の符号化

文字も 0 と 1 で符号化されている。当然だ。その内容を詳しく見ていこう。

文字には、それぞれに番号がわり当てられている。たとえば「A」なら「65 番」、「p」なら「112 番」というように。この割り当てる操作を「エンコード」という。

でも、実はその割り当て方（文字コード）にはたくさん方法がある。日本語が使えないのは、ASCII や ISO/IEC 8859 などがある。この 2 つはともに 1 バイト文字といわれ、1 文字を 1 バイトの整数（8bit の 2 進数）で表している。だから、256 種類の文字しか使えない [*22]。

逆に日本語が使えるのには、JIS やシフト JIS、EUC-JP などがある。これらは 2 バイト（16bit の 2 進数）で表している。だから、65536 種類の文字を扱うことができる。

全世界の文字を一度に扱えるものには、UTF-8 などがある。Unicode で文字コードを世界統一にしようぜとか言ってる人たちが作っている。UTF-8 は、8 というだけあって、1 文字を 8 バイト（64bit の 2 進数）で表している。こうすることで、全世界のさまざまな言語の文字を扱うことができるのだ。

間違って、書いた時と違う文字コードで読み込んでしまうと、どうなるか。たとえば、EUC-JP で「宙乃ちゃん愛してる」と書いて、それをシフト JIS だと思い込んで読むと「ｿ霽ｫ､ﾁ､网・ﾇ､ｷ､ﾆ､・」となる。EUC-JP では「宙」に当たる番号が、シフト JIS では「ｿ」に対応しているのだ [*23]。これが文字化けの正体。

文字化けを解決するためには、コード体系の**標準化・統一化**を図ることが必要だが、歴史的経緯や利害関係からまとまらないことも多々ある。なにも文字に限ったことではなく、コンピュータメディア全般にわたって蔓延する問題である。

[*22] 正確には、1 バイトのうちの 1 ビットは別の情報（システム関連）に当てられていることが多い。JIS や UTF-8 などでも何ビットかがそうである。実際扱える文字数は、ここで述べている数字よりもずっと少ないことになる。

[*23] と書いたが、実際は違う。シフト JIS の「ｿ」は 1 バイトで「霽」は 2 バイト、EUC-JP の「宙」は 2 バイトだから、シフト JIS の「ｿ」と「霽」の前半分が、EUC-JP の「宙」に対応している。

# 第 IV 部
# データモデル

データモデルというのは、データを利用しやすいように、うまく格納する方法のことだ。データの整理方法みたいなもの。単にデータベースを作る、といっても、その作り方はさまざまで、データが出し入れがしやすいように、上手く片付けておかなければならない。

教科書だと第 4 章にあたり、教科書では 6 つくらい紹介されている気もするが、出るのはほとんど 1 つだけなので、それだけ詳しく理解しておいて、あとは「こんなのがあるんだ〜」程度に把握しておけばよい。

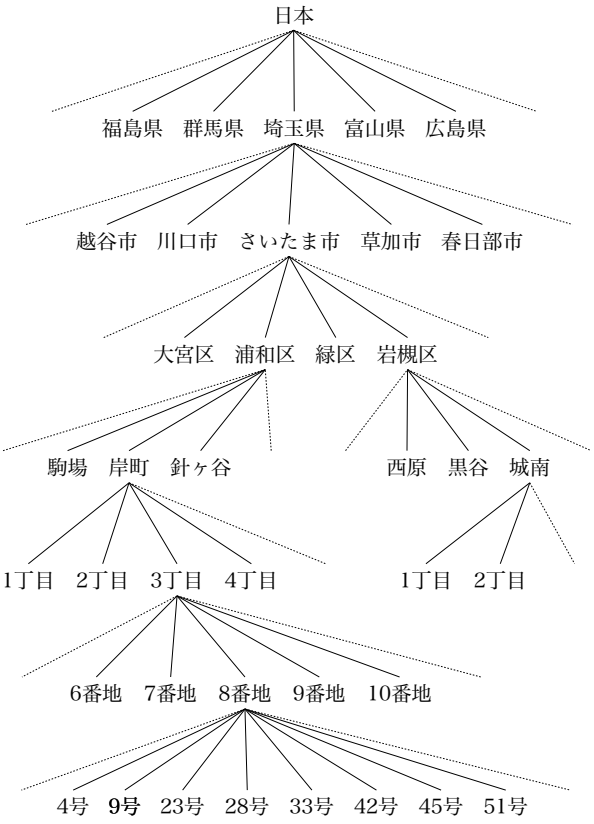
## 11　階層型データモデル

データモデルの代表であり、試験ではほとんどこれしかでない。

階層型データモデルというのは、データを**木構造**で表したデータモデルのことだ。すべてはこの「木構造」という言葉に尽きる。

パソコンのフォルダの管理や、図書館の本の分類、住所、会社内の役職などがこれにあたる。右図は住所の例。都道府県、市町村、区、町……と階層型にわかれている。その要素が何に所属しているかなどを表すときに有効である。

根っこからどんどん枝分かれしていって、最小単位までたどりつく。その際、枝と枝が繋がることはない [*24]。




[*24] ここでは住所の例を出したが、住所の例でも、枝と枝が繋がることがある。たとえば、埼玉県さいたま市浦和区大原には 1〜5 丁目あり、6 丁目と 7 丁目は大宮区にある。すなわち、浦和区と大宮区から出た枝がもう一回大原で繋がってしまうことになる。したがって、住所も厳密にいえば木構造（階層型）になっているわけではない。だが、そこはめをつぶってほしい。

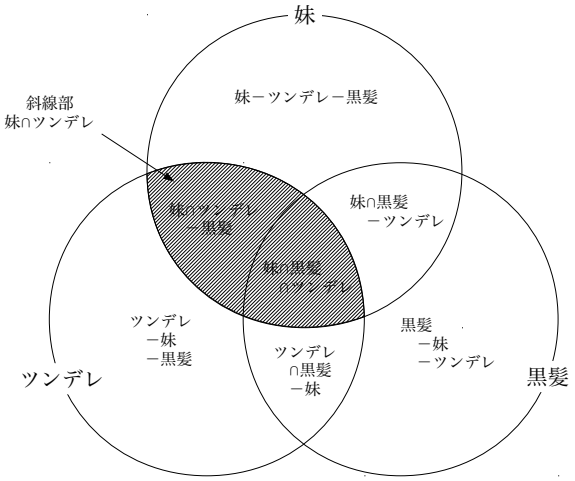
# 12　その他のデータモデル

まあ、読んどきなよ。

## 12.1　集合モデル

なんか、集合で表すんだよ。

集合 $A$ と集合 $B$ の共通部分（積集合）を $A \cap B$ って書いたり、集合 $A$ と集合 $B$ の両方の要素が全部含まれる集合（和集合）を $A \cup B$ って書いたり、集合 $A$ の要素から集合 $B$ の要素を取り除いた集合（差集合）を $A - B$ って書いたりする。




ベン図みたいなのを想像するとわかりやすいかも。例が錯乱しているけど気にしない。宙乃ちゃんだったら、妹で黒髪だけどツンデレじゃないから、集合「妹 ∩ 黒髪 – ツンデレ」の要素だし、集合「妹 ∩ 黒髪」の要素でもあるし、集合「ツンデレ ∪ 黒髪」の要素でもあるし、そもそも集合「妹」や集合「黒髪」の要素でもある。だけど、「妹 ∩ ツンデレ」の要素ではないし、「妹 – 黒髪」の要素ではない。

他にも、桐乃（俺の妹がこんなに可愛いわけがない）だったら、集合「妹」と集合「ツンデレ」の要素であって、集合「黒髪」の要素でないから、集合「妹 ∩ ツンデレ」（斜線部）の要素だし、集合「ツンデレ – 黒髪」の要素でもあるけど、集合「黒髪 ∩ ツンデレ」の要素じゃないし、集合「妹 – ツンデレ」の要素でもない。

あとは、好きなキャラクターを当てはめて遊んでください。全部の領域を埋めてみよう!

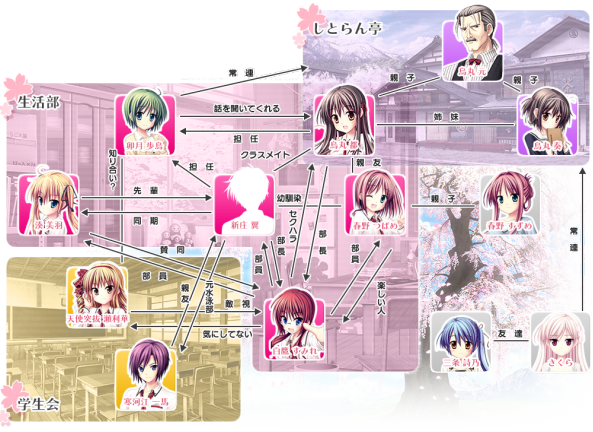
こんな感じ。

## 12.2　ネットワークモデル

1 つ 1 つのデータ（ノード）が、線（エッジ）で結ばれてる感じ。でも、階層型データモデルみたいに、データ自体の上下関係はない。枝分かれした奴がもう一回繋がってもいい。

インターネット上のサーバー同士のつながり[*25] や、路線図などがそれにあたる。

右図のキャラクター相関図[*26] もその例。キャラクター（ノード）同士が、線（エッジ）でつながっている。キャラクターが集合（生活部、学生会など）に属してるから、集合モデルとの融合型でもある。




[*25] インターネットの仕組みの章で詳しく解説する

[*26] 『さくら、咲きました。』公式サイトより。http://www.sorahane.org/

## 12.3　関係モデル

これが一番データベースっぽいやつ。リレーショナルデータモデルともいう。

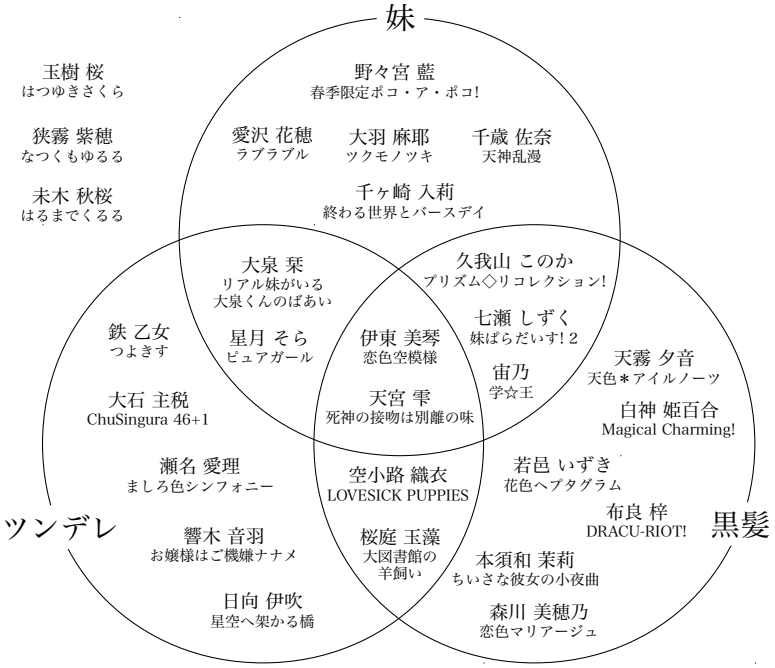
例として、木星の衛星を取りあげてみよう。1 番目の衛星は、1610 年にガリレオが発見した、イオ（Io）だ。赤道半径は 1821km、質量は $8.94 \times 10^{22}$kg （木星に対して $4.70 \times 10^{-5}$）、木星の周りを 1.7691 日で一周する。これは「イオ」「1.7691 日」「1821km」「$4.70 \times 10^{-5}$」という、「名前」「公転周期」「赤道半径」「木星に対する相対質量」の関係があるといえる。他の衛星にも同じ事が言えるはずだ。これを表にまとめると、以下のようになる [*27]。これこそが、いわゆるデータベースというやつだ。

| 番号 | 名前 | 公転周期 [日] | 半径 [km] | 木星に対する相対質量 |
|---|---|---|---|---|
| 1 | Io | 1.7691 | 1821 | $4.70 \times 10^{-5}$ |
| 2 | Europa | 3.5512 | 1565 | $2.53 \times 10^{-5}$ |
| 3 | Ganymede | 7.1545 | 2634 | $7.80 \times 10^{-5}$ |
| 4 | Callisto | 16.6890 | 2403 | $5.67 \times 10^{-5}$ |
| 5 | Amalthea | 0.4982 | — | — |
| 6 | Himalia | 250.57 | 85 | — |
| 7 | Elara | 259.65 | 40 | — |

関係を表にまとめられることで、論理的にわかりやすく、プログラムを書くのが他のデータモデルに比べて簡単で、データベースとして使われることも多いため、結果的にデータベースの代表となった。

## おまけ

集合モデルの、「好きなキャラクターを当てはめよう!」の解答例（協力:さなきち氏）。義妹も妹としました。2 つを区別する人、ごめんなさい。




[*27] 理科年表より。実際にはもっとたくさんあるが、ここでは一部を抜粋した。

# Coffee Break ～そもそもなぜ情報を学ぶのか～

『情報』は、比較的新しい概念。日本語の語彙としては森鴎外がクラウゼヴィッツの「戦争論」の翻訳で用いた造語の軍事用語が起源と言われている。もともとの定義は「敵と敵国に関する知識の全体」

——百野栞（11eyes）

なんでこんな教科を大学まで来てやらなければいけないのか。
高校時代に、パソコンをいじるだけの教科だった情報を、どうして大学でもやるのか。
もはや高校時代は情報の授業なんて形式上でしか存在しなかったのに、なんで大学でやるのか。
そういう人もいるかもしれない。

## 情報とは何か

情報というのは、形がない。「道端で歩いてたら 100 円玉を拾った」「アシアナ航空の飛行機が落ちた」「明日は雪が降るらしい」「『ぷ。』がボウリングしてる人に見える」「昨日彼女とデートした」——全部情報だ。それに、科学的根拠があるのかどうか、なにか具体的な物体として存在するのか、そんなことは知ったことではない。情報は単なる概念だ。だから、**情報の授業はコンピュータの扱い方を学ぶ授業ではない**。情報の扱い方を学ぶ授業だ。

## 情報の多面性

情報は多面的だ。表現や伝達などの人間に関わる面、情報機械を用いるなどした問題解決に関わる面、そして、情報システムが繁栄した社会に関わる面をもっている。

多面的である以上、それぞれをそれぞれに学ぶだけでは足りない。お互いの関連を把握しながら、総合的・系統的に学ばなければならない。

## 速すぎる技術の進歩に

読者もわかるとおり、情報技術の進歩は早い。しかも、それは身近におこっている。だから、「専門家じゃないから問題から目を背けてればいいよね」とか「便利に使えさえすればよくて、問題がでてきたらそれは他人の責任だよね」とか、そういうことは言っていられない。

そのためには、情報社会人としての基本的素養を持つことが必要で、それを教えるのが、高校・大学の情報という科目である。

読者諸君は、この受験戦争の頂点である、東京大学に入学された。これから、新たな技術を開発し、新たな地を開発し、未来や時代を引っ張っていく存在になりうる。そういった諸君が、情報をうまく利用したり、問題から逃げず真っ向から議論することで問題解決を図ったりする際、ここで学んだ知識が少しでも役に立てば光栄である。

以上。情報の教科書のはじめにおよび第 1 章を適当にまとめてみた。こんな高尚なこと考えながら授業してるんですかね……。してないよね……。このシケプリもそんなこと考えてつくってないもん……。

次は、インターネットの仕組みとか!

# 第 V 部
# インターネットの仕組み

今や世界中で利用されている、インターネット。情報を運ぶ手段であるそのインターネットの仕組みについて学ぶ。細かい通信の内容は、おって学べばよい。全体的に出やすい内容が入っているが、最後の電子署名はほとんどでない。教科書では第 3 章にあたる。

## 13 Web ページを見よう

Web ページを見る際、どんなふうにコンピュータたちがやり取りをしているのだろうか。それを、手順を追って見ていこう。

### 13.1 DNS により相手コンピュータの IP アドレスを調べる


Web ページを見る際には、相手のコンピュータに「データを送って?」と尋ねるところから始まる。だけど、それよりも前に、相手のコンピュータがどこにあるかがわからないと困る。

コンピュータのネット上の位置は、IP アドレスという 4 つの 8 ビットの整数で決まっている（0.0.0.0〜255.255.255.255 の 4294967296 通り）[*28]。だけど、これでは人間が覚えにくいので、DNS（Domain Name System）というシステムで、人間の覚えやすい文字（URL）と、番号（IP アドレス）を関係付けている。

今回の例 `http://www.lumpofsugar.co.jp/` では、相手のコンピュータ（Lump of Sugar のサイトが乗っているコンピュータ）の IP アドレスは、124.146.200.148 だ。

### 13.2 HTTP でデータの送信を依頼するようなメッセージを送る


「データを送って!」と頼まないと送ってくれないので、相手のコンピュータ（例の場合なら 124.146.200.148）に、「データを送れｺﾗｧ」という内容のメッセージを送る。

具体的には、以下の様な手順で送る。（これもちゃんと覚えてね）

#### 13.2.1 メッセージをパケットに分割する

メッセージをパケット（小包）に分割する [*29]。一度に大量のデータを送ると、途中でミスってた場合に送り直すのが大変なので、ちょっとずつ小分けにして送って、送られた方がもう一回順番に戻して、それを読むことになっている。

逆に細かく分け過ぎると効率が悪いので、ちょうどいいくらいのサイズ

[*28] この方法（IPv4）では、約 42 億通りと、世界の人口分にも満たないので、新たな IPv6 というアドレスが開発されている。こちらは約 340澗（$3.40 \times 10^{38}$）通りに対応している。日本 Google（`http://www.google.co.jp/`）は、IPv4 だと 74.125.227.159、IPv6 だと 2607:f8b0:4000:801:0:0:0:1018 という番号がついている。2 つの番号には特に関連がなく、IPv4 のアドレスしかもっていない Web サイトがほとんどである。

[*29] 具体的な大きさ特に決まっていない。NTTdocomo の通信料の計算では、1 パケット 128 バイトを基準としている。

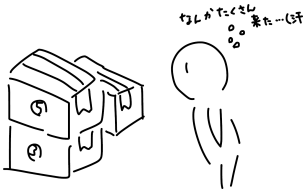
になるようになっている。

### 13.2.2　パケットごとに相手コンピュータに送る


さっき作ったたくさんのパケット（小包）を、自分のコンピュータの 80 番のポート（後で解説）から、相手コンピュータ（124.146.200.148）に送る。

このとき、直接相手コンピュータとケーブルでつながっていることはまずない。いろんなコンピュータをあちこち寄り道しながら、相手コンピュータを探す旅に出たパケットたちが、いつしか辿り着く。

### 13.2.3　相手コンピュータに届く




さっきの小包が相手コンピュータに届く。HTTP（あとで解説）のメッセージは、相手コンピュータの中でも、80 番のポートに送られることになっている。

ポートというのは、データの出入り口の番号みたいなもの。同じ出入り口に大量にいろんなものが通ると、混雑してごっちゃになったり、何が何だかわからなくなったりするので、メッセージの種類ごとにメッセージの出入り口を分けることになっている。

ほかに、メールの受信（POP3 メッセージ・これもあとで解説）は、110 番とか 995 番とかと決まっている。全部が全部ポートの役割が決まっているわけではなく、好きなように使っていいポートの番号もたくさん用意されている。

### 13.2.4　相手コンピュータがパケットを並べ替えてメッセージを読む

最後の段階。パケットはそれぞれいろんな道を通っているので、もとの順番どおりには届かない。相手コンピュータは、届いたパケット（小包）を開けて順番を並べ替えて、そのメッセージを読む。

なお、壊れてるパケットがあるな、と思ったら、送信元のコンピュータに「もう 1 回送ってよ」というメッセージを送って、もう 1 度メッセージを送ってもらう。

## 13.3　データを返す

相手のコンピュータ（124.146.200.148）がメッセージを読んで、「データを欲しがっている」ということを理解したら、124.146.200.148 のコンピュータにあるデータを、同じようにパケットに分割して、もとのこちらのコンピュータに送りつける。

それを並び替えて読んで表示することで、こちらのコンピュータはめでたく Lump of Sugar のサイトを見ることができる。

これが Web ページをブラウザ [*30] で見る際のだいたいの仕組みである。細かいことをいえばちょっと違うかもしれないが、だいたいこれで把握しておこう。

[*30] Internet Explorer、Chrome、Firefox、Safari など、君たちがネットサーフィンするときに使うアレ。

## 14　プロトコルとは?

プロトコル、というのは、メッセージを送る時の約束事のこと [*31]。メッセージをどういう順番に書くか、とかどういう順でやり取りするか、などが書かれている。はがきの表に郵便番号・住所・宛名を書く場所が決まっているのと同じような感じ。時と場合によって、そのプロトコルを使い分けながらメッセージを送る。

インターネットで一番根本にあるのは、**TCP/IP** [*32] というプロトコル。このプロトコルのおかげで、いろんな国のコンピュータや、種類のコンピュータ同士が通信できる。

その上で、ネットをパソコンで見るときに使われるのが、HTTP [*33] というプロトコル。これによって、好きな Web サイトを、自分のコンピュータのブラウザで見ることができる。

さっきちらっと説明した POP3 [*34] は、メールを受信するときのプロトコル。メールサーバ [*35] からメールを持ってくるときに使う。

## 15　ドメイン名とホスト名と URL

さっき、DNS によって、URL と IP アドレスが関連付けられていると述べたが、厳密には違う。

DNS（Domain Name System）は、IP アドレスとホスト名を結びつけているものである。

こんがらがってきたね。

Lump of Sugar の Magical Charming!の公式ホームページのアドレスは、`http://www.lumpofsugar.co.jp/product/magicha/index.php` である。これは、インターネット上の**ファイル** [*36] の場所を明確に示している。この `http://`から始まる長い名前を URL（Uniform Resource Locator）という。

じゃあドメイン名とホスト名とは何か。

ドメイン名はこの場合は `lumpofsugar.co.jp` で、ホスト名は `www.lumpofsugar.co.jp` である。何が違うねん、てな。

ホスト名はコンピュータの名前を指し、ドメイン名はそのコンピュータのグループの名前を指すと思ってもらっていい。ドメイン（グループ）の中にホスト（コンピュータ）が 1 つしかないときもあるし、複数あるときもある。ちなみに、Lump of Sugar のドメインには他に、`lumpofsugar.co.jp` という名前のホストもあり、こちらはオフィシャルファンクラブなどに使われている。

なんでこんなに面倒な仕組みになっているかというと、それは DNS というシステムの問題だ。

例えば一箇所に全部のコンピュータの IP アドレスとホスト名の対応が書いてあるととてもわかりづらい。そこにアクセスが集中して重くなったり、IP アドレスやホスト名が変わったとかしたら、それこそ面倒だ。

---

[*31] コンピュータ同士じゃなくて、人間同士でもプロトコルというので、インターネットとかの場合は特に「情報プロトコル」と言ったりもする。

[*32] Transmission Control Protocol / Internet Protocol の略。前者が、1 対 1 の正確なデータを転送する時のプロトコル。後者が、IP アドレスを定義しているプロトコル。ともに、インターネットや様々なネットワークで多く用いられているプロトコル。

[*33] HyperText Transfer Protocol の略。正確には、Web ブラウザと Web サーバの間で HTML（Web ページの内容）や画像・音楽・動画などのコンテンツを送受信する際に用いられるプロトコル。

[*34] Post Office Protocol Version 3 の略。他にも IMAP（Internet Message Access Protocol）などを使って受信できる。前者はメールサーバからメールをダウンロードした後、メールサーバからそのメールを削除するが、後者はメールサーバにメールをおいたまま、それを見たり管理したりする。

[*35] コンピュータにメールが届く前にとっといてくれるところ。手紙が届く前の郵便局みたいなところ。

[*36] 正確にはリソースであるが、ここでは気にしない。

そこで、DNS を階層的に分けた。

ドメイン名の最後に.jp というのが付いているのに気付くだろう。これは「日本の DNS であとは管理しますよ」という印である。コンピュータは、とりあえず最後に「jp」って書いてあるから、日本の DNS に問い合わせよう、となるのだ。

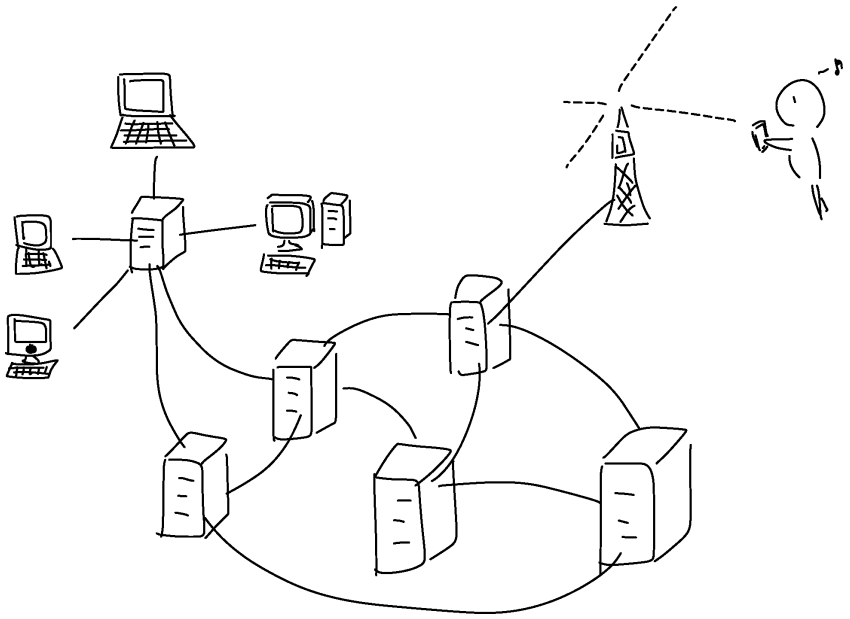
そこで、.co の部分の DNS で問い合わせろと言われる。「日本の企業の DNS さんに問い合わせるとわかりますよ」と日本全体の DNS に言われるのだ。そうすると、lumpofsugar.co.jp が持っている DNS のある場所を教えてくれる。

最後に、lumpofsugar.co.jp の持っている DNS で、www.lumpofsugar.co.jp というホスト名のついたパソコンの IP アドレスを教えて下さい、と頼むと、みごとめでたく 124.146.200.148 という番号を教えてくれる。

こういう、たらい回しシステムにすることで、それぞれの DNS の管理を分散させて負担を軽くしているのだ。

こうして IP アドレスを教えてもらってコンピュータを発見できるようになったわけだが、その後このコンピュータに「キミのコンピュータにあるはずの/product/magicha/index.php というファイルを送ってくれヨォ」と頼むことで、指定した URL のファイルが送られ、みごと Magical Charming!のオフィシャルホームページが見られることになる。

## 16　ネットワーク概観



ネットワークはだいたいこんなかんじになっている（適当）。

パソコンから送られたデータは、たいていルータ（サーバ）[*37] という箱を通って、外界の世界に流れていく。ひとつのルータからいくつかのパソコンに伸びている小さな世界を LAN（Local Area Network）、ルータ同士がつながっている広い世界を WAN（Wide Area Network）という。インターネットそれ自体も、WAN である。

上の説明では、いろんなコンピュータを通って、と書いたが、実際には、コンピュータ→ルータ→ルー

[*37] もちろんルータとサーバは違うもの。正確な定義ではないが、とりあえず「中継地点はルータで、何かを送信してくれるところは Web サーバ」くらいに覚えておこう。

タ→……→ルータ→コンピュータ、が正しい流れ。

# 17　暗号化

お互いに秘密にしながら通信したいことがある。

たとえば、住所や電話番号などの個人情報や、パスワードは、流出したら困る。他にも、内緒話とか彼女との会話とか聞かれたら困る。

そういうときに、暗号化をすることを考える。基本的には大きく 2 種類ある。

## 17.1　共通鍵暗号方式

暗号化と復号化のときに同じ鍵を使う方法。

たとえば、「あ」→「い」、「く」→「け」、「そらのちゃんかわいい」→「たりはつゅあきをうう」[*38] などと文字を一文字ずらす暗号を考えよう。「文字を一文字ずらす」という行為が、この場合の鍵である。

これは、暗号化するときも復号化するときも一文字ずらせばいいので、これは共通鍵暗号である。

でも、この方法（鍵）がバレたら、全部会話が筒抜けである。さて、これをどう回避するか。それが 2 つ目の方法だ。

## 17.2　公開鍵暗号方式

公開鍵と秘密鍵の 2 つの鍵を用意しておく。

公開鍵で暗号化すればそれを秘密鍵で復号化できて、逆に秘密鍵で暗号化すれば公開鍵で暗号化できる。

そんな特殊なペアの鍵を作る。

メッセージを送るときは、公開鍵で暗号化する。鍵は公開されてるから、だれでも同じように暗号化できる。でも、秘密鍵を持っているのは、正しい受信者だけ。他の人が盗み見ても、秘密鍵は世界に 1 つしかないので、復号化はできない。

そんな仕組みの暗号。理解しにくいし、雲をつかむような方法だ。だけど、素因数分解の難しさを利用してリベスト、シャミル、エイドルマンが RSA 暗号というものを開発し、それを実現させることができたのだ。

$p$、$q$ を素数とし、$n = p \times q$ とする。$p \times q$ はコンピュータで計算するのは簡単だが、$n$ から $p$ と $q$ を割り出すのは、$n$ が大きくなるととても困難なのである [*39]。公開鍵を $n$ を利用する鍵、秘密鍵を $p$ と $q$ を利用する鍵にすれば、公開鍵から秘密鍵を推測できない暗号ができる。

今使われている暗号は、最速のスーパーコンピュータで計算しても何百年とかかるので、理論的には公開鍵から秘密鍵が推測できるが、現実的には不可能な暗号となっている [*40]。

[*38] 「ん」→「あ」とした

[*39] 例えばともに素数だとわかっている 173501 と 761227 の積が 132073645727 なのは（計算すれば）すぐにわかる。だが、2 つの素数の積である 320879757151 が、何と何の積なのかは、かなり調べないとわからない。こういう感じ。

[*40] 逆に、素因数分解を高速で行う方法を発見すれば、全世界の暗号化されたネット通信が読み取れてしまうことになるので、みんな挑戦してみよう。

# 18　電子署名

電子署名（デジタル署名）は、公開鍵暗号を利用して、電子文書が作成者以外の人に改ざんされていないかを確かめるためのもの。普通の文書データだけじゃなく、Web サイトでも同じように用いられている。サービスをする側の送った Web サイトと、今君たちがパソコンで見ているサイトが同じものかどうかをチェックすることができる。

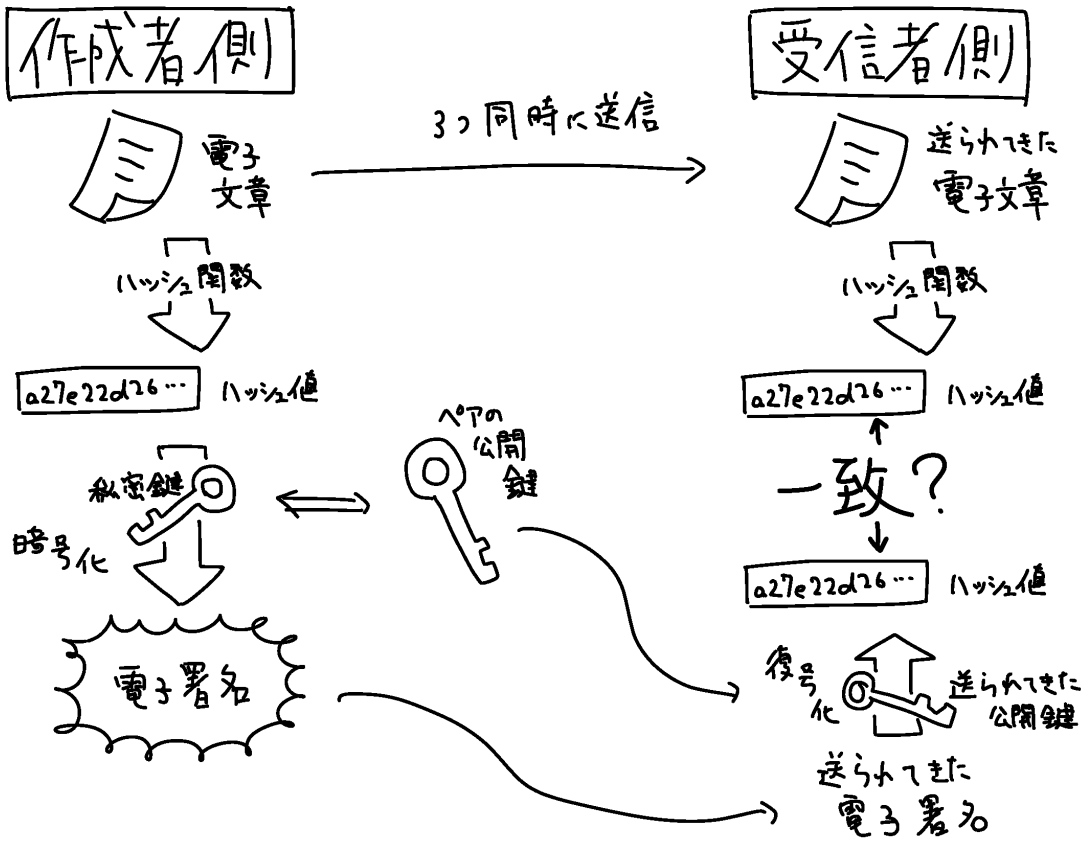
まず、電子文書を「ハッシュ関数」というものを用いて、「97949B4DC9566D14C……」といった英数字の羅列（ハッシュ値）にする。ハッシュ関数には SHA-1 や MD5 など数種類ある。また、ハッシュ関数で作られた英数字の羅列からは、元の電子文書を推測することはできない。

こうしてできたハッシュ値、自分の持っている秘密鍵で暗号化する。

電子文書に、自分の公開鍵と、秘密鍵で暗号化したハッシュ値をくっつけて送信する。その暗号化されたハッシュ値が電子署名である。

作成者本人しか秘密鍵を持っていないから、逆にその暗号化されたハッシュ値が作成者の証明となる。これが電子署名といわれる所以だ。

送られた相手は、一緒についてきた公開鍵で、電子署名を復号化する。そうして、作成者が作った文章のハッシュ値を求めることができる。これと、送られた相手自身が文章から計算したハッシュ値と比べる。この 2 つが同じであれば、作成者と送られた人の見ている電子文書は、完全に同じものであるというのがわかる。



# Coffee Break

## ～解けない問題と解けたかどうかもわからない問題～

### チューリング機械の停止問題

停止性問題（Halting Problem）ともいう。

プログラムを入力したとき、たまに無限ループが起こって、止まらなくなるプログラムがある。そこで、プログラム $A$ に値 $x$ を入れて、そのプログラムがちゃんと終了するかどうかを判定してくれる、素晴らしいプログラム $T(A,x)$ を作ろう。

このプログラムは素晴らしいから、ちゃんとどんなプログラムや値を入れても、それが終了するかどうかを「YES」か「NO」という答えで返してくれる。すなわち、必ず終了するプログラムであるとする。

そこで、プログラム $T(A,A)$ が YES と返した時に止まらない、NO と返した時に止まるという、プログラム $M(A)$ を作ろう。$T$ の 2 つめの $A$ は、プログラム $A$ が表す数字か文字か何かだと考えていい。

じゃあプログラム $M(M)$ を考えたら、このプログラムは止まるだろうか?

- プログラム $M(M)$ が止まった→プログラム $T$ はプログラム $M(M)$ を「止まらないプログラム」と判定したことになる→でも $M(M)$ は止まってるじゃん→矛盾
- プログラム $M(M)$ が止まらなかった→プログラム $T$ はプログラム $M(M)$ を「止まるプログラム」と判定したことになる→でも $M(M)$ は止まらないじゃん→矛盾

明らかにおかしい。だから、そんな素晴らしいプログラム $T$ は存在しない。

コンピュータはどんな難しい問題でも解けるようにみえるが、こんな問題は解けない。これは「矛盾のない理論体系内では、必ず証明できない命題がある」というゲーデルの不完全性定理 [*41] を、コンピュータに当てはめたものともいえる。

### チューリングテスト

コンピュータは人間の知能に追いつけるか。

これは、人間の知能というものを正しく定義しないと解けない。でも、いまだにそれは十分に定義されておらず、解けない。

知的だと考えられるいろんな行為（チェスとか将棋とか）で検証することは可能だが、一般的にその問題を解くのは、不可能だというのはわかるだろう。

コンピュータが知的かどうかを判断する試験に**チューリングテスト** [*42] というものがある。「人間とコンピュータを文字だけで [*43] 会話させて、会話の後でその人間が、会話の相手がコンピュータだったかどうかを判定できるか。判定できなければ、そのコンピュータは知的である」というもの。

それだけで機械が知的かどうかを判定はできない気もする。相手の言うことをなるべく多く予測してプログラムしておけば、人間を模倣することができるし、それは知的と言えるのか。

議論はおさまらない。

---

[*41] Gödelscher Unvollständigkeitssatz。2 つあり、もう 1 つは「ある理論体系に矛盾がないとしても、矛盾がないことをその理論体系内では証明できない」というもの。

[*42] 2012 年に、開一夫の個別問題で出題。

[*43] 声や表情などの生体的な特徴から、人間かどうかを判断させないようにするため。

# 第 VI 部
# 情報と社会

現代社会では情報技術開発の成果は社会全体のみならずその社会を構成する 1 人 1 人に影響するため、各自が情報技術について知り、情報リテラシーを身につける必要がある。

試験では、法の解釈や事件の詳細などについて聞かれるため、おおまかにざっと把握する程度であるが、法律用語や事件名が出てきたときは注意して読んでいこう。

どちらかというと練習問題をやるほうが効率的かもしれない。選択問題で出やすいので、逃げることも可能。教科書だと第 10 章にあたる。

## 19　情報技術の変化と影響

まずは大まかな歴史的な流れを見よう。

1960〜70 年代　大きなコンピュータでの巨大データベースの管理と大規模計算
　→情報集積による権力の集中、プライバシー保護が問題
1980 年代　マイクロコンピュータが消費者個人単位で市場に流通。コンピュータ産業・ビデオ産業も誕生
　→ソフトウェアの所有権の問題や人工知能の論議
1990 年代　インターネット技術が普及
　→情報コミュニケーション技術（ICT）[*44] が論議され、プライバシー保護も再び問題に

このように短期間で発展してきた情報技術は、社会を支配する権力や体制の構造を変化させるものと考えられることがある。従来のコンピュータ技術は、こちらの端末と相手側のホストコンピュータ [*45] による小さな範囲のものであったが、インターネット技術により端末のつながりや情報伝達の範囲が世界レベルに広がったからである。こういった論考はメディア論と呼ばれる。

情報技術の発達により、われわれは情報を

1. 送受信者のいる位置に関わらず〈場所の制約からの解放〉
2. 超短時間で〈時間の制約からの解放〉
3. 経路を選ばず〈経路の制約からの解放〉
4. 低コストで

送ることができる。これにより地理的要素にとらわれない自由なコミュニケーションが可能となった。

## 20　情報に対する権利 —著作権—

情報に関して権利等の問題が生じるのは、情報の無形性と複製可能性によるものが大きい。

無形性は、情報がデジタル化されたことで従来の「もの」概念とは異なる性質をもった形のないものであるということ。以前はレコードや写真などの「情報の実物」があったが、今では CD やメモリーカード

[*44] Information and Communication Technology の略。今までの IT（Information Technology）に代わる表現。ネット上で情報の共有を念頭においた表現である。
[*45] 相手側で、こちらにデータを送ってくれるコンピュータ。インターネットの仕組み（第 V 部）出てきた「ホスト名」のついたコンピュータのこと。

に保存でき、さらにはインターネット上までに情報を残せるようになったため、情報の実体が見えなくなった。

複製可能性は複製（コピー）とオリジナルの区別がない、つまり全く同じものをいくつも作れるということをいう。

情報技術が発達したことで音楽、映像、文章などは手に取れる「もの」だけでなく、電子的な情報（電子媒体）の形で市場に流通するようになった。電子媒体はそれが（あるいはその権利が）他人に所有されているという意識を希薄にし、コピーなど著作権侵害行為への抵抗を減らしてしまっている。ここに著作権の問題が生じる。

著作権の保護のために著作権法が存在する。知的な創作活動によって何かを作り出した人に対して、他人に無断で利用されない権利を付与する制度が知的財産権であり、著作権はその 1 つである。情報技術関連ではコンピュータプログラム、ディジタルコンテンツ等が保護される。

## 20.1　コンピュータプログラムの著作権

著作権が付与される（すなわち著作物として認められる）のはソースプログラム、アプリケーションプログラムなどの「作られた」プログラムであり、プログラムを作るためのプログラム言語などは対象にならない。このようなプログラムを無断複製し商業的に利用するなどすれば著作権侵害となる。

## 20.2　デジタルコンテンツの著作権

上で述べた電子媒体で扱われるものがデジタルコンテンツである。この形式はコピーや情報交換が容易であるため、デジタル著作権管理（DRM）[*46] が必要となる。

このテーマについて教科書では Winny 事件について取りあげられているが、Winny についてはこのシケプリの過去問および解説を見てもらえば十分なので省略する。一応、この事件をふまえた情報技術における著作権の論点を羅列しておく。

1. 匿名でファイル交換ができるため著作権を意識しなくなる
2. 匿名性を規制していいのか（プライバシー、情報交換の自由度の観点から）
3. 匿名であるがゆえの情報倫理の必要性
4. ソフトを開発した研究者の責任かソフトの利用者の責任か
5. 作成者と受益者の利益のバランスがとれた設計をめざす
6. 著作権を保障しないとコンテンツの作成者が正当な利益が得られずやる気をなくす

# 21　プライバシーとセキュリティ

コンピュータ技術により膨大な個人情報が蓄積されるようになり、ネットワーク技術により情報交換の規模も大きくなっている。このとき、個人情報は保護されなくてはならないというのがプライバシーの議論であり、個人情報や機密情報の安全をいかに守るかというのがセキュリティの議論である。

[*46] Digital Rights Management の略。複製を防止するなどの、デジタル著作権を保護する技術の総称。

### 21.1 プライバシーと個人情報保護法

個人の経歴や行動が悪用されたり監視に利用されたりすることは、民主主義における自由を侵害することになる。そこで日本では**個人情報保護法**が 2005 年 4 月に施行された（公布は 2003 年 5 月）。国際的にはプライバシー保護と個人データの国際流通についてのガイドラインが OECD に示されている。

個人情報保護法が基本法以外に民間事業者に課しているルールは、

利用、取得に関するルール （個人情報は適切な目的でしか使っちゃダメよ）
適正、安全な管理に関するルール （ちゃんと安全に管理しなきゃダメよ）
第三者提供に関するルール （本人の同意なしに第三者に流しちゃダメよ）
開示などに応じるルール （個人情報の主から求められたら、ちゃんと開示・訂正・利用停止しなきゃダメよ。苦情が来たら、適切に迅速に対応しなきゃダメよ）

の 4 つである [*47]。

### 21.2 セキュリティ

情報システムにおいて情報が流出・漏洩しないような安全性の確保を情報セキュリティという。このセキュリティに求められることは

機密性 認可されたものだけが情報にアクセスできるようにすること
完全性 情報が正確で完全であること
使用可能性 必要なときに必要な情報資源にアクセスできること

の 3 つがある。それぞれに問題があると情報漏洩、改ざん、使用妨害につながる。また、この内特に機密性に関しては**不正アクセス禁止法**が 2000 年 2 月に施行されている（公布は 1999 年 8 月）。

情報セキュリティ確保の技術として暗号化技術があるが、これについては暗号化の節（p.43）を参照。

## 22 技術の中立性

技術を中立と考える立場には、技術は社会の状態に関係なく発展するという技術資本主義、そして技術が何を社会にもたらそうと技術に善し悪しはないという価値中立論がある。一方で、社会がそのとき必要な技術を自ら選択し、技術はその中で淘汰されてきたとする立場を技術の社会構成主義という。この考え方は技術は中立ではないと考える。

## 23 情報リテラシー

情報リテラシーはパソコンなどの情報機器を操作できる能力と、操作したときに実際に何が行われているのかある程度想像がつくという能力のことをいう。情報技術は日に日に技術革新してより日常生活に浸透してきているため常に様々な問題も発生しており、これを解決するための議論に参加するためには、情報を批判的に読み自ら取捨選択できる情報リテラシーを身につけることが求められる。

[*47] 詳しくは教科書 P.245 の表 10.2 を見てね。

# 第 VII 部
# そのほかの細かいところ

とりわけ部を作ってまで説明するほどのことでもないが、それでもテストには出るので説明しておく。この 2 つも選択問題として出るので、逃げることは可能。

## 24　情報システムの裏側

ここでは教科書第 8 章の内容を扱う。そもそも情報システムとは一般にコンピュータとネットワークを用いて情報サービスを提供する仕組みのことで、現代ではウェブサービスから家電、携帯機器などに用いられている大変身近なものだ。

ただしその種類は多様であるため、ここでは情報システムが具体的にどのようなものなのか、チケット予約システムを例にとって解説する。

### 24.1　システムの構成

インターネットでチケットを予約する際は、パソコンで開いたウェブブラウザからウェブサーバにアクセスし、そこでチケット予約サービスを提供するウェブページを閲覧して購入することになる。この流れを細かく見ていくこう。

1. パソコンでウェブブラウザを開く
2. チケット予約のウェブページの URL を入力する（クライアントがサーバに URL を送る）
3. サーバが公演情報やチケット販売状況などを表示するページを作成し、そのファイルをウェブブラウザに送り返す
4. ブラウザが受け取ったファイルを解析し、ページを表示する
5. 購入者が個人情報を入力する
6. 入力された情報に基づき決済が完了し、チケットが購入者に送られる

Web ページを閲覧する仕組みについては第 V 部「インターネットの仕組み」を読んでいればわかるはずなのでここでは省略するが、いくつか説明が必要なポイントがある。

一つ目がウェブブラウザとウェブサーバの関係で、これは**クライアント/サーバ型**の構成といわれる。ネット上でサービスを提供するプログラムである「サーバ」にサービスを要求するプログラムが「クライアント」（この場合はウェブブラウザ）という関係である。「サーバ」は情報が集積されたデータベースや、アプリケーションと呼ばれるプログラムなどを一手に管理している。（ちなみ「サーバ」は上記のようなプログラムが置かれているコンピュータを指すこともある。）

次に今説明に出したアプリケーションという言葉だが、これは適用（application）プログラムと呼ばれるもので、サーバが要求された処理を行うときに個々の処理に応じて呼び出されるプログラムのことをいう。アプリケーションがデータベースにアクセスして情報を得るとき、アプリケーションとデータベースの関係はクライアント/サーバの構造となっている。

## 24.2　システムに求められること

チケット予約システムのような情報システムは、どんな人であっても初めてページを見て利用できるようにわかりやすく作られていること、また、利用者に不便を感じさせないつくりであることが求められる。さらに、個人情報を扱う場合は情報保護の対策も必要となる。

### 24.2.1　トランザクション処理とロールバック

利用者が予約のための情報を入力しているとき、座席指定の情報などデータベースのデータが更新される必要がある [*48]。セッションと呼ばれる一連の作業の中、この処理はトランザクション処理と呼ばれる。万が一トランザクション処理が中断された場合、実態とは異なる状態にデータが更新されてしまっていることになるため、その処理が行われる前の状態にデータベースの内容を戻す必要がある。この処理をロールバックという。

### 24.2.2　排他制御

チケット予約システムの場合は同時に多数の人がアクセスするため予約が重複しないような処理が必要であり、この処理を排他制御という。具体的には、データベースを更新するときに、その予約をした人（のクライアント）以外がデータベースに触れないようにしておく（ロックをかけておく）という方法がとられる。

### 24.2.3　個人情報の保護

ポイントは 3 つで、情報が外に漏れないこと、なりすましを防ぐこと、情報が利用者の知らないところで使われないこと、が求められる。

1 つ目は情報通信の暗号化が対策となるが、これは暗号化の節（p.43）を参照のこと。他人になりすましての購入は、利用者コード（ID など）とパスワードによる管理や生体認証により防がれる。3 つ目の、情報の漏洩や販売などは個人情報保護法で規制されており、サービス提供者が示したプライバシーポリシーに同意した利用者のみが情報を提供するようになっている。

### 24.2.4　アクセスの監視

ネットワーク上の情報システムはサービス停止攻撃（不正な大量アクセスによってサーバに過大な負荷をかける行為）を受ける危険性があるため、アクセスの監視やアクセス記録の収集が不可欠である。ただしこれももちろん「プライバシーを侵害しない範囲で」行なわなければならない（たとえば、そのアクセス記録を適当に公開したらいけない）。

# 25　ユーザインタフェース

インタフェース（interface）は英語で「中間面、境界面」を表す言葉であり、特に情報技術について扱う場合には、情報機器と情報機器あるいはプログラムとプログラムなどを仲介するもののことをいう。ここでは教科書第 9 章にならって、そのなかでもコンピュータとその利用者（ユーザ）の間にあるユーザインタフェース（ヒューマンインタフェース）と呼ばれるものについて解説する。

[*48] ある人がある席をとりました、という情報を受け取ったら、すぐにデータベースに反映しないといけないということ。そうしなきゃ他の人が同じ席をとれちゃうかもしれない。そしたらダブルブッキングだ。

## 25.1　ユーザインタフェースとは

### 25.1.1　コンピュータのユーザインタフェース

前節にあるような情報システムを利用する場合、利用者はユーザインタフェースを介してシステムを利用することになる。コンピューターの場合ではキーボード、マウス、ディスプレイなどがこれに挙げられ、利用者とコンピュータが情報を伝達しあうための装置がユーザインタフェースと呼ばれる。

### 25.1.2　インターフェースの二重接面性

ユーザインタフェースは利用者がシステムの機能を最大限に利用できるように設計されている必要があるが、それは同時に利用者にとってわかりやすいものでなければならない。ここにインタフェースの二重接面性という問題がある。

コンピュータにおける二重接面性を簡単に説明する。まず利用者が目的とするコンピュータ上の仕事（タスク）があるとき、

1. 利用者によるコンピュータの操作（利用者 $\longleftrightarrow$ コンピュータ）
2. プログラムによるタスクの達成（プログラム $\longleftrightarrow$ タスク）

という過程を経てタスクは行われる。過程 1、2 におけるインタフェースをそれぞれ第一接面、第二接面とすると、利用者の目的は第二接面にあるにもかかわらず、利用者が操作できるのは第一接面である。この不一致が二重接面性であり、利用者の直感的操作をしづらくさせる要因となっている。

二重接面性は、情報の入出力の方法が限られていながら多様なソフトウェア（プログラム）によって多様な機能を実現できるというコンピュータのインタフェースに特有のものである。

## 25.2　ユーザインタフェースの種類

コンピュータのインターフェースにはおおまかに 2 種類ある。1 つは利用者がコンピュータを操作する際のインターフェース「入力デバイス」、もう 1 つがコンピュータからの情報を利用者に与え操作の良し悪しを評価させる際のインターフェース「出力デバイス」である。

### 25.2.1　入力デバイス（キーボード、マウス、マイクなど）

この内マウスはポインティング・デバイスという 2 次元の位置情報を入力するデバイスの一種である。ポインティング・デバイスは直接触れて位置を指定できる直接入力型デバイス（タッチスクリーン [*49] など）と、位置を指定するもの（ポインタ、座標等の数値など）を操作・入力する間接入力型デバイス（マウスなど）の 2 種類に分けられる。

### 25.2.2　出力デバイス（ディスプレイ、プリンタ、スピーカーなど）

出力デバイスにおいて重要なのは、GUI と CUI というディスプレイの表示方法に関する 2 種類のインタフェースである。テストによく出る。

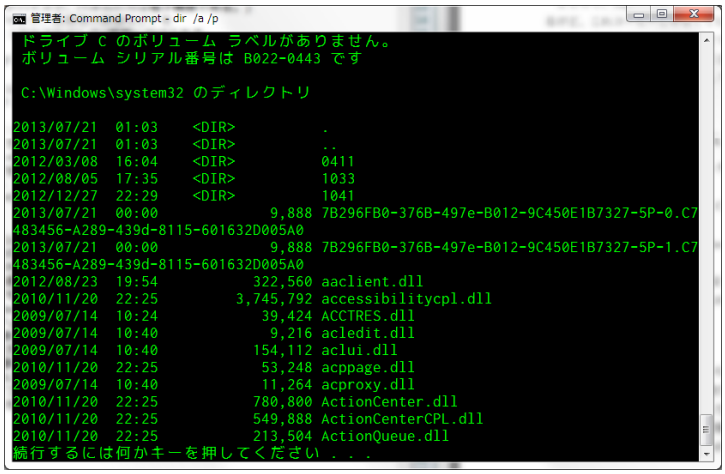
CUI[*50] とは、キーボードで文字列を入力することで命令を与え、操作の結果も文字列のみで出力するインターフェースである。初心者には扱いづらいが、慣れればすばやく作業ができる……らしい。

[*49] タッチパネルとは違う。タッチパネルは、タッチスクリーンの上に貼り付いている膜のことをいう。スクリーンの方はデバイスで、パネルの方は電子機器である。

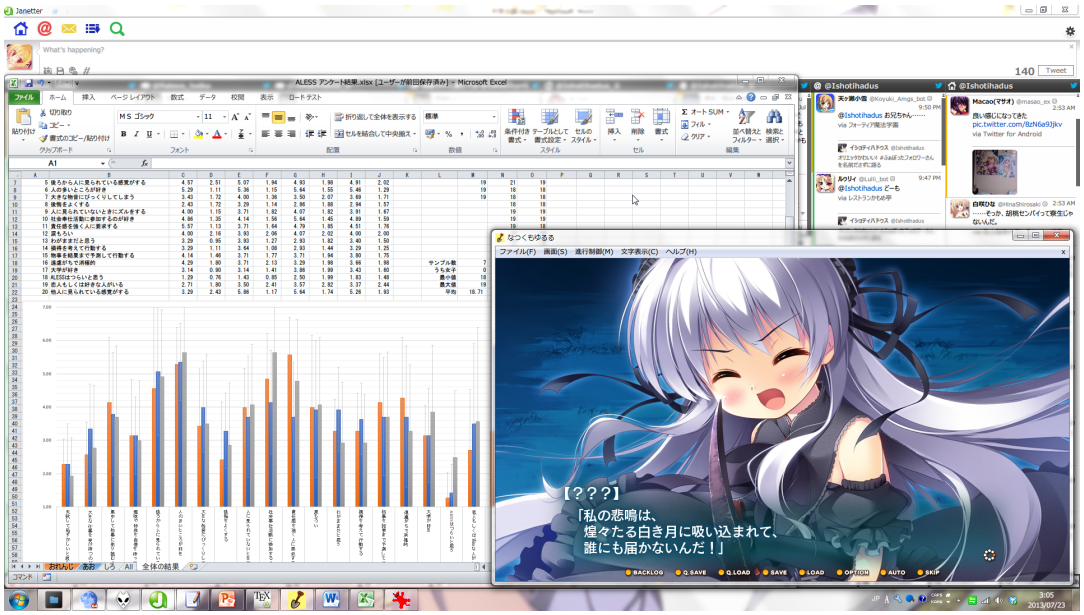
[*50] Character User Interface

Windows でいうコマンドプロンプト *51、Mac でいう端末だ。

GUI*52 は、表示に絵や画像を用いており、視覚的な操作とコンピュータ内部での実際的な操作が対応する（例えば、データファイルが複製されると画面上でもファイルを表す絵が複製されるなど）デスクトップメタファというものが通常採用されている。これは Windows などを思い浮かべてもらえばよいが、このような GUI はウィンドウ（Window）・アイコン（Icon）・メニュー（Menu）・マウス（Pointer）により構成されるため、WIMP システムとも呼ばれる。GUI の背景には「その装置あるいは表示を見れば、どのように実行可能かが即座にわかるようになる」ことを意味する**直接操作**という考え方がある *53。




CUI の例。Windows のコマンドプロンプト。初心者には使いにくいが、コンピュータの内部までしっかり操作することができる。




GUI の例。色んなことを視覚的に、感覚的に、しかも同時にできる *54。

*51 スタートメニュー→すべてのプログラム→アクセサリの中に入っている。

*52 Graphical User Interface

*53 さらにこの考え方の背景には、「外界の環境や事物が、生体の活動に供するべくもっている情報」を意味するアフォーダンスというギブソンによる概念がある。ここに書いている以上もちろんこれも試験範囲である。

*54 だが、図のようにエロゲをしながら ALESS はだめである。しかも後ろに Twitter が待機されている。よくない。コンピュータ的には同時並行ができても、人間は同時並行ができない。ちゃんとまじめに ALESS かエロゲに集中すべきだ。

# 第 VIII 部
# 付録

特に教養情報として説明しておくまでのことはないもの。

## 26 計算などの方法

ここに載せているものは、すべて知っておくべき計算である。付録の中でも、このセクションだけは十分に把握しておいてもらいたい。

### 26.1 log とは

$x = a^y$ であるとき、$\log_a x = y$ と書き、「$y$ は、$a$ を底(てい)とする $x$ の対数である」という。この場合の $x$ を真数(しんすう)とよび、これは正でなければならない。また、底（この場合の $a$）は正であり、1 であってはならない。

たとえば、$\log_2 1024$ は、$2^{10} = 1024$ だから、$\log_2 1024 = 10$ となる。

以下は log を計算する際の基本的な計算法則。

- $\log_a(MN) = \log_a M + \log_b N$
- $\log_a \left(\frac{M}{N}\right) = \log_a M - \log_a N$
- $p \log_a M = \log_a M^p$
- $\log_a M = \frac{\log_b M}{\log_b a}$

### 26.2 データ量の単位

コンピュータのデータは、bit や byte などの単位で、その量を表す。2 進数 1 ケタで 1bit、2 進数 8 ケタで 8bit=1byte となっている。また、1024byte=1KB (kilobyte) と表すことになっている。同様に、1024KB=1MB (megabyte)、1024MB=1GB (gigabyte) である。まとめると、以下の様な感じ。

1byte = 8bit
1KB = 1024byte = 8192bit
1MB = 1024KB = 1048576byte = 8388608bit
1GB = 1024MB = 1048576KB = 1073741824byte = 8589934592bit

## 27 二分探索

第 2 章では、少しアルゴリズムを紹介した。二分探索は、ちょっと理解が難しいが、それでもアルゴリズムとしては初歩的なものなので、教養情報の範囲にもなっている。ぜひ理解してもらいたいアルゴリズムの 1 つである。

配列の要素は小さい方から順番になっているが、その要素は飛び飛びのものを考える（$a_1 \leq a_2 \leq \cdots \leq a_n$ となっている、下のような配列）。

| 添字 $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 要素 $a_i$ | 0 | 3 | 7 | 8 | 9 | 12 | 13 | 15 | 18 | 21 |

ある値 $t$ を示され、$a_x = t$ となるときの、$x$ を求めることを考えよう。(たとえば、13 と言われたら、それは $a_7$ だよ、というふうに答えるプログラムを考える)。

このとき、単純に考えるのは、前から順番に探していく方法だ。コードを書いてみると、以下のようになる。

```
i ← 1
x ← 0
n ← 配列 a の要素数
while i ≤ n do
  if a_i = t then
    x ← i
  endif
  i ← i + 1
done
return x
```

計算量のオーダーは、$n$ 回繰り返すループが 1 つあるので、$\mathrm{O}(n)$ である。

だが、これでは要素の数が増えた時に時間がかかる。ここで、二分探索という方法を考える。

上の配列を例にして考えてみよう。ここでは、$a_x = 13$ となるときの、$x$ を求めることを考えよう。

まず最初に、全 10 個のうち、$a_x = 13$ が前半にあるか後半にあるかを調べる。ここで、$a_6 = 12$ だから、$a_x$ は後半にあることがわかる。

つぎに、後半 5 つのうちで、$a_x$ が前半にあるか後半にあるかを調べる。$a_8 = 15$ だから、$a_x$ は、後半 5 つのうちの前半 2 つの中にあることがわかる。

つぎに、後半 5 つのうちの前半 2 つのうちで、$a_x$ が前半にあるか後半にあるかを調べる。$a_7 = 13$ だから、$a_x$ は、後半、つまり $x = 7$ のところにあるとわかる。

これをコードで書いてみよう。$t$ が、上の例での 13 にあたる。

```
j ← 配列 a の要素数
k ← 1
while j ≥ 2 do
  c ← (j ÷ 2 の整数部分)
  if a_{k+c} > t then
    j ← c
  else
    j ← j - c
    k ← k + c
  endif
done
if a_k = t then
  return k
endif
```

$j$ はそのとき考えている個数、$k$ はその考えているのが何番目から始まるのかを示している。また、$c$ は、そのとき考えている個数の前半部分の個数である。

1 回目のループだったら、$j = 10$ で、$k = 1$ だ。前半部分はは $c = j \div 2 = 5$ 個だから、$k + c$ 番目、すなわち後半部分の 1 番最初の要素 $a_{k+c}$ と、$t$ の大小を比較する。そのとき、$a_{k+c}$ の方が大きかったら、$a_x$ は前半にあるから、$j$ 個で考えていたのを $c$ 個にして、もう一回考える。逆に、$t$ の方が大きかったら、$a_x$ は後半にあるから、$j$ 個で考えていたのを $j - c$ 個にして、後半部分を考えるから $k$ は $k + c$ として、次に後半部分を考える。

こんな感じ。

これの計算量のオーダーを考えよう。たとえば 10 個のとき、ループする回数は、1 回目（$j = 10$）、2 回目（$j = 5$）、3 回目（$j = 2$）の 3 回だ。すなわち、配列の個数を何回 2 で割ったら 2 以下になるかがわかればよい。これを考えれば、$m$ をループの回数として、

$$j \div 2^m = 2$$

の式が成り立つ。これを $m$ について解くと、

$$m = \log_2 j - 1$$

となる。すなわち、計算量のオーダーは $\mathrm{O}(\log n)$[*55] となる。

# 28　ブール代数

AND とか OR とかを、数式みたいに表したもの。一種の表現方法だ。$+$、$\cdot$、$\oplus$、$\odot$ を演算子として、論理関数を表す。あとの 2 つは計算にはあまり使われず、簡略化するときのみに使う。過去に出題例あり。

$+$ が AND、$\cdot$ が OR、$\oplus$ が XOR [*56]、$\odot$ が EQ [*57] を表している。また、$\overline{x}$ で、$x$ の否定となる。

一応表にしてみると、以下の様な感じ。

| $x$ | $y$ | $x + y$ | $x \cdot y$ | $x \oplus y$ | $x \odot y$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |

なんでブール代数表現があるのかというと、それが四則演算みたいに、交換則や結合則が成り立って、等価な論理関数をたくさん書けるからだ。

計算法則をすこし書き並べておく。覚える必要はないが、確かにそうだよね、ということを感じてもらいたい。

**冪等則**　$x + x = x$、$x \cdot x = x$
**交換則**　$x + y = y + x$、$x \cdot y = y \cdot x$
**結合則**　$(x \cdot y) \cdot z = x \cdot (y \cdot z)$、$(x + y) + z = x + (y + z)$
**吸収則**　$x + (x \cdot y) = x$、$x \cdot (x + y) = x$
**分配則**　$x \cdot (y + z) = x \cdot y + x \cdot z$、$x + y \cdot z = (x + y) \cdot (x + z)$
**ド・モルガンの法則**　$\overline{x + y} = \overline{x} \cdot \overline{y}$、$\overline{x \cdot y} = \overline{x} + \overline{y}$

[*55] $n$ は配列 $a$ の要素数とする。
[*56] eXclusive OR、排他的論理和。MIL 記法だと  になる。2 つの値が異なったら 1、同じだったら 0 を返す。
[*57] EQuivalence、等価。MIL 記法だと  になる。2 つの値が同じだったら 1、異なったら 0 を返す。XOR の否定。

たとえば、次のような式変形ができる。

$$x\cdot\overline{y}+\overline{x}\cdot y=(\underbrace{x\cdot\overline{x}}_{\text{常に }0}+x\cdot\overline{y})+(y\cdot\overline{x}+\underbrace{y\cdot\overline{y}}_{\text{常に }0})=x\cdot(\overline{x}+\overline{y})+y\cdot(\overline{x}+\overline{y})=(x+y)(\overline{x}+\overline{y})$$

$$x\cdot y+\overline{x}\cdot\overline{y}=(x\cdot y+\underbrace{x\cdot\overline{x}}_{\text{常に }0})+(\overline{x}\cdot\overline{y}+\underbrace{y\cdot\overline{y}}_{\text{常に }0})=x\cdot(\overline{x}+y)+\overline{y}\cdot(\overline{x}+y)=(x+\overline{y})\cdot(\overline{x}+y)$$

ちなみに、上が $x\oplus y$、下が $x\odot y$ を表したものになっている。

慣れれば、AND とか OR とかをいちいち書いているよりは使いやすいかもしれない。

# 29　ハフマン符号化

ハフマン符号化は、もっとも効率的な符号化の方法。数学的帰納法で証明できるらしいが省略する。

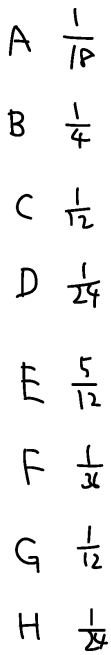
A〜H の 8 種類の文字が、以下の確率で出現するとしよう。

| 文字 | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 出現確率 | $\frac{1}{18}$ | $\frac{1}{4}$ | $\frac{1}{12}$ | $\frac{1}{24}$ | $\frac{5}{12}$ | $\frac{1}{36}$ | $\frac{1}{12}$ | $\frac{1}{24}$ |

このとき、もっとも効率のいい符号の作り方を、順をおって説明していく。同じように書けるようになれば、もう君も一人前のハフマン符号化 er だ。

## 確率と一緒に書き並べる

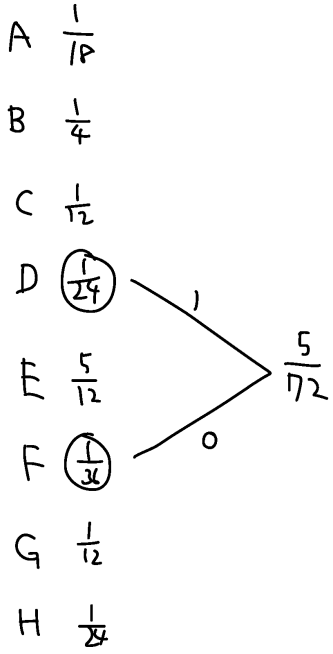
下図のように、とりあえず縦に書き並べる。




## 一番小さい 2 つの確率を選んで木を作る

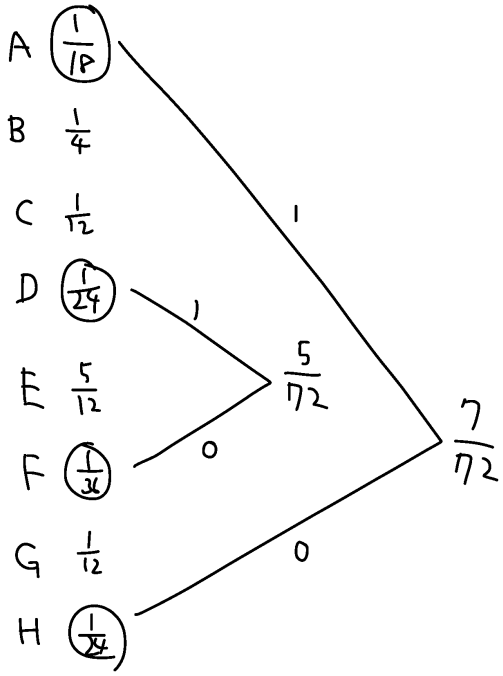
最も小さい 2 つの確率を探して、そこから線を下図のように結ぶ。何種類か引けそうだが（今回は $\frac{1}{24}$ が 2 つあるので、2 通りある）、どっちを引いてもかまわない。

線を結んだところに、2 つの確率の和（下図であれば、$\frac{1}{24} + \frac{1}{36} = \frac{1}{72}$）を書く。また、引いた線の上に 0 および 1 と書く（どっちがどっちでもいい）。足した 2 つの確率を丸で囲んでおくと後でラク。



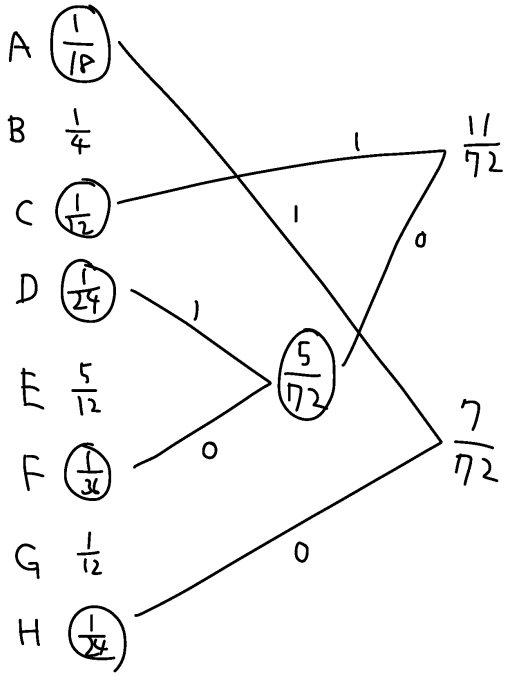

丸で囲まれていない一番小さい 2 つの確率を選んで同じように木を作る

$\frac{1}{18}$ と $\frac{1}{24}$ が一番小さいので、下図のように同じように書く。



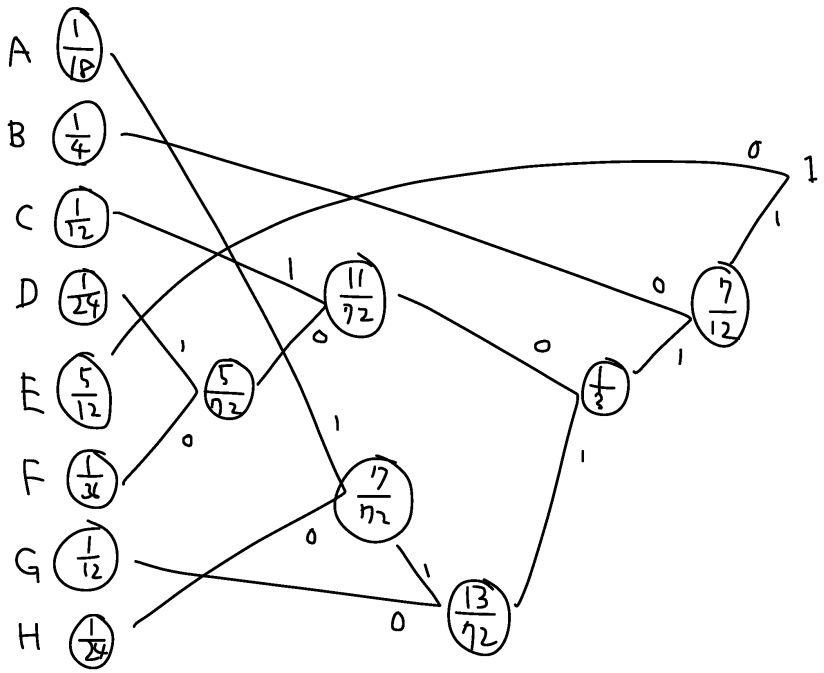
### くり返す

今度は、丸で囲まれていない確率のうち、$\frac{1}{12}$ と、さっき足して出てきた $\frac{5}{72}$ が一番小さいので、この 2 つから線を引っぱる。




### 1 になるまでくり返す

確率が 1 になるところまで続ける。続けると下図のようになる。



確率を足していって出てきた 1 以外の確率に、全部丸がついていれば OK。こうしてできた図を、ハフマン木という。

### 枝の根元の方から読む

「1」と書いてあるところから順番に読んでいって、それぞれの要素の符号を決める。下図の例であれば、A は 11111、B は 10、C は 1101……といったふうに。この時、A〜H の文字の書いてある側からではなく、確率を足していって 1 と書いた側から読むことに注意しよう。

### まとめる

こうして完成した符号がこちら。

| 文字 | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 符号 | 11111 | 10 | 1101 | 11001 | 0 | 11000 | 1110 | 11110 |

ちゃんと曖昧でない符号になっていることも確認してみよう。

# 第 IX 部
# 練習問題

## 30　練習問題の使い方

この練習問題は、あくまでも暇な人がその暇を持て余すためのものです。ですが、せっかくなのでやっておくと役に立つかもしれません。

過去問から抜粋しています。過去問は解答がありませんから、そもそも問題が完全ではないかもしれないこと、解答が不十分かもしれないことをあらかじめここで述べておきます。

問題は部ごとにわかれています。部末問題代わりに使うのもアリだと思います。

解答は、第 X 部（p.69〜）に載っています。

## 31　コンピュータと計算の知恵

### 31.1　プログラミング

$p$、$q$ を異なる自然数とする。このとき、与えられた自然数 $d$ について、$d$ 以下の自然数 $k$ のうちで、

$$k = mp + nq \qquad (m, n \geq 0, \quad m, n \in \mathbb{Z})$$

のように表すことができるものを、小さい順にすべて列挙し、最後にその個数を表示したい。そのために次のようなプログラムを作った。ここで、`int(x)` は $x$ を超えない最大の整数を表す関数である。なお、すでに変数 $p$、$q$、$d$ には値が代入されているものとする。

```
u ← 0
k ← 1
while k ≤ d do
  if k - int(k/p) * p = 0 then
    [ア]
    u ← u + 1
  else
    m ← 0
    while m ≤ int(k/p) do
      r ← k - m * p
      if [イ] then
        [ア]
        u ← u + 1
        m ← int(k/p)
      endif
      [ウ]
    done
  endif
  k ← k + 1
done
```

総数は u であると表示

このとき、以下の設問に答えよ。

**(1)** 上記のプログラムの［ア］、［ウ］に当てはまるものを、それぞれ次の⓪～⑦の中から 1 つずつ選べ。

⓪ k ← k + 1　　① r ← r + 1　　② u ← u + 1
③ m ← m + 1　　④ k を表示　　⑤ r を表示
⑥ u を表示　　⑦ m を表示

**(2)** 上記のプログラムの［イ］に当てはまるものを、次の⓪～⑤の中から 1 つ選べ。

⓪ $\mathtt{r - int(\frac{r}{m}) * m} \neq 0$　　① $\mathtt{r - int(\frac{r}{m}) * m} = 0$
② $\mathtt{r - int(\frac{r}{p}) * p} \neq 0$　　③ $\mathtt{r - int(\frac{r}{p}) * p} = 0$
④ $\mathtt{r - int(\frac{r}{q}) * q} \neq 0$　　⑤ $\mathtt{r - int(\frac{r}{q}) * q} = 0$

**(3)** 上記のプログラムを実行し、変数 $p$、$q$、$d$ にそれぞれ 3、7、15 を入力した時、整数の列

3　［エ］　7　9　［オ］　12　13　14　15

に続いて、

総数は 9

が出力される。このとき、`if k - int(`$\frac{\mathtt{k}}{\mathtt{p}}$`) * p = 0 then` の下の［ア］は［カ］回行なわれたことになる。

また、変数 $p$、$q$、$r$ にそれぞれ 3、7、100 を入力したとき、整数の列に続いて、

総数は［キ］

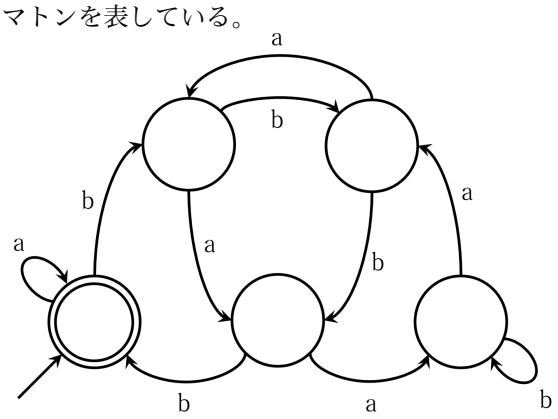
が出力される。

このとき［エ］～［キ］に当てはまる数を答えよ。

→解答は p.69

## 31.2　有限状態機械（オートマトン）

**(1)** 以下の図はあるオートマトンを表している。



以下はこのオートマトンの動作を説明するチェック表である。(a)〜(e)を埋めて、チェック表を完成させなさい。

| 入力 | 終了状態 |
|---|---|
| 無し | Yes |
| b | No |
| aa | Yes |
| baaa | (a) |
| bab | (b) |
| baab | (c) |
| baba | (d) |
| bbaab | (e) |

また、以下はこのオートマトンの動作を説明したものである。(f)を埋めて、説明を完成させなさい。

a と b だけからなる入力を、「a」が 0、「b」が 1 に対応した 2 進数だと読みかえる。このとき、入力が [ (f) ] の時に終了状態となるオートマトン。

**(2)** 暗証番号「abc」または「acaba」が入力されたときに解錠する（終了状態になる）電子錠をオートマトンとしてモデル化することを試みる。ただし、途中間違った暗証番号を押しても後からどちらかの暗証番号を入力すれば解錠するものとする。また一度解錠した後は何を入力し続けても解錠状態となる。入力は a、b、c の 3 つだけからなるものとする。

以下はこのオートマトンの動作を説明するチェック表である。(g)〜(j)を埋めて、チェック表を完成させなさい。

| 入力 | 終了状態 |
|---|---|
| 無し | No |
| acabc | Yes |
| acaba | Yes |
| abaca | (g) |
| ababa | (h) |
| aabc | (i) |
| acababc | (j) |

また、この条件を満たすオートマトンを作成して図にしなさい。

→解答は p.70

## 31.3　論理回路と論理関数

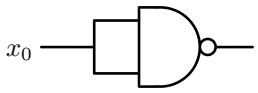
下の表は、NOT、AND、OR、NAND、XOR の各演算を表した真理値表である。

| $x_0$ | $x_1$ | NOT($x_0$) | AND($x_0, x_1$) | OR($x_0, x_1$) | NAND($x_0, x_1$) | XOR($x_0, x_1$) |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | A | E | 0 |
| 0 | 1 | 1 | 0 | B | F | 1 |
| 1 | 0 | 0 | 0 | C | G | 1 |
| 1 | 1 | 0 | 1 | D | H | 0 |

NAND は、NAND$(x_0, x_1)$ = NOT(AND$(x_0, x_1)$) となる演算である。

(1) 表中の空欄 A〜H を埋めよ。

(2) NOT$(x_0)$ = NAND$(x_0, x_0)$ であるため、MIL 記法を用いると NOT は次のように 1 つの NAND によって表現できる。MIL 記法を用いて AND を 2 つの NAND により表現せよ。




(3) MIL 記法を用いて OR を 3 つの NAND により表現せよ。

(4) MIL 記法を用いて XOR をなるべく少ない（最小は 4 つ）NAND により表現せよ。

→解答は p.72

# 32　情報とデジタルデータ

## 32.1　情報量とは & 情報と符号

古代の遺跡から、$\{0, 1\}$ の 2 つの記号からなる謎のデータの一部で長さ 20 の記号列が発見された。

```
10000111000100010000
```

この謎のデータが $\{0, 1\}$ の 2 つの記号が決まった確率で生成されるデータであり、確率は観測されたデータの中の出現頻度であると仮定して平均情報量を求めてみる。以下の表の(a)〜(e)を埋めなさい（値を簡単に計算できない場合は、式の形で表現すること）。

| 記号 | 出現回数 | 出現確率 | 情報量 | 平均情報量 |
|---|---|---|---|---|
| 0 | 14 | 0.7 | (a) | 0.7× (a) |
| 1 | (b) | (c) | (d) | (c) × (d) |
| 計 | 20 | 1.0 | | (e) |

その後の研究でこの謎のデータは単なる 2 進数の列ではなく、A→00、B→01、C→10、D→11、のようにして A、B、C、D の 4 つの文字を符号化した列だったことがわかった。発見された記号列は、

```
CABDABABAA
```

を表現したいものであるということになる。そこで、$\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ の 4 つの記号が決まった確率で生成されるデータであり、確率は観測されたデータの中の出現頻度であると仮定して、平均情報量を計算しなおしてみた。以下の表および文中の(f)〜(n)を埋めなさい（値を簡単に計算できない場合には、式の形で表現すること）。なお、(n)には「と等しい」、「より大きい」、「より小さい」のうちのどれかが入るものとする。

| 記号 | 出現回数 | 出現確率 | 情報量 | 平均情報量 |
|---|---|---|---|---|
| A | 5 | 0.5 | (f) | 0.5× (f) |
| B | 3 | 0.3 | (g) | 0.3× (g) |
| C | 1 | (h) | (i) | (h) × (i) |
| D | (j) | 0.1 | (k) | 0.1× (k) |
| 計 | (l) | 1.0 | | (m) |

(e)と(m)を比較すると、2 進数の表現で 2 文字分なので、(m)が(e)のちょうど 2 倍になることを期待した。実際に計算してみたところ、(m)は(e)の 2 倍 (n) ことがわかった。

→解答は p.74

## 32.2　アナログ情報のデジタル化

音楽 CD（コンパクトディスク）は、2ch（ステレオ）で約 74 分の音を記録することができる。これに関して以下の各問にすべて答えなさい。

**(1)** 音楽 CD における量子化と標本化について、以下の用語・数値を必ず 1 回以上用いて 2〜3 行程度で記述しなさい。

量子化　標本化　44.1kHz　16bit

**(2)** (1)で答えた量子化・標本化の方法で、2ch（ステレオ）で 1 秒分の音を記録すると何ビット必要になるか答えなさい。

**(3)** (1)で答えた量子化・標本化の方法で、74 分の音を記録するためのデータ量は何バイト必要か答えなさい。

→解答は p.75

## 32.3　文字の符号化

次の文章中の(ア)、(イ)に、もっとも適する用語・数字を解答群から選択せよ。また、文章中の下線部は用語の誤りが存在する。訂正する用語が最少となるように誤り部分を抜き出し、正しい語句と置き換えて修正せよ。

数々の仕様の中から規格とされる仕様を確立する作業は標本化と呼ばれる。この作業によって定められた仕様は、製品開発やユーザの使い勝手に大きく影響する。現在、日本語文字コードは (ア) やシフト JIS、EUC-JP など複数のコード体系が混在しており、時々 (イ) を引き起こしてしまう。

【解答群】

128　256　65535　65536　アルゴリズム　アプリケーションプログラム　量子化　記号化
文字化け　復号　CUI　GUI　JAS　JIS

→解答は p.76

## 33　データモデル

次の(1)、(2)の問に答えよ。解答するための仮定が不足する場合には、自分で補い、解答にどのような仮定を補ったかを明記せよ。

(**1**) 以下の(a)〜(e)のような [対象] のデータモデルとして、階層モデル（木構造）を用いるのが適切でないものをすべて挙げよ。なお、[問題例] は各対象を用いてどのような問題解決を行うかの例である。

(**2**) (1)で適切でないものとして答えた各対象について、階層モデルが適切でない理由を 2 行以内で書け。

(a)　[対象]　大学の授業とそれらの間にある「一方を先に履修しておかなければいけない」という関係。
[問題例]　卒業までに履修しておきたい複数の授業があったときに、それ以外に履修しなければいけない授業の集合を求める。

(b)　[対象]　会社における社員と、それらの間にある直属の上司と部下という関係。
[問題例]　2 人の社員がいたときに、2 人の共通の上司で最も身分の低い者を見つける。

(c)　[対象]　インターネットにおけるコンピュータやルータとそれらの間の接続関係。
[問題例]　2 つのコンピュータが通信する際に、経由するネットワーク（ルータ）の数を最小にするような経路を見つける。

(d)　[対象]　「ビデオで予約録画を行う」課題を「開始時刻の設定」などの下位の課題に分割し、さらに下位の課題を具体的な操作になるまで分割したもの。
[問題例]　課題を完了するのに必要な具体的な操作の回数を数える。

(e)　[対象]　人間の親子関係。
[問題例]　（夫婦でない）2 人の人間が何親等であるかを調べる。ただし、親等は 2 人の間にある親子関係の個数の最小のものとする。

→解答は p.76

## 34　インターネットの仕組み

以下の各小問に解答せよ。まず、あたえられている例文に含まれる誤りの箇所を簡潔に指摘せよ。次に、用語から 2 つ以上用いて、主題に関して説明する文章を作成せよ。説明の文章の量は 50〜150 字を目安とする。

(1)　[主題]　インターネットを通じて電子メールが届く仕組み
[例文]　送信者が電子メールを書くときに用いたコンピュータと、受信者がメールを読むときに用いたコンピュータが、通信ケーブルで直接に結ばれている必要がある。
[用語]　DNS、パケット、ルータ

(2)　[主題]　公開鍵暗号方式
[例文]　公開鍵暗号方式でメッセージを送る際は、送信者と受信者がお互いに鍵を交換し、公開された鍵を用いて暗号化と復号化を行う。
[用語]　公開鍵、共通鍵、秘密鍵、平文

(3)　[主題]　インターネットと通信の所要時間
[例文]　自宅のインターネット設備を整えた記念に、1 メガバイトのファイルをインターネットからダウンロードした際には $a$ 秒かかった。したがって今後 10 メガバイトのファイルをダウン

ロードする際には $a \times 10$ 秒より長かかることはないと結論づけられる。

[用語]　サーバ、DNS、パケット、ルータ

⑷　[主題]　ドメイン名と地域や組織

[例文]　インターネット上で使われるホスト名やドメイン名とは、完全に仮想的な名前であり、接続者の国、地域、所属組織などとの対応関係は一切ない。

[用語]　IP アドレス、DNS、ドメイン名

→解答は p.76

# 35　情報と社会

大きく分けて、主題論述型と自由記述型の 2 つがある。主題論述型の方のみ解答を載せた。

## 35.1　主題論述型

事件や主題と絡めて説明させる問題。いくつか載せておく。

### 35.1.1　Winny にまつわる問題

デジタル情報の所有権をどのように管理すればよいかについては、様々な問題が存在する。たとえば、ファイル共有ソフト Winny に関しては、まず 2003 年にその利用者 2 名が逮捕され、2004 年に開発者が逮捕された。開発者に関しては、一審で有罪判決、二審で無罪判決、その後 2011 年に最高裁判決で無罪が確定した。

⑴ Winny 利用者 2 名の逮捕理由について、以下の用語を用いて簡潔に記述せよ。各用語の初出部分に、下線を引くこと。

著作権者　ネットワーク　ハードディスク

⑵ Winny 開発者を起訴した側の主張を 3 行程度で述べよ。

⑶ 同じく、開発者の支援者の主張を 3 行程度で述べよ。

→解答は p.77

### 35.1.2　Google 図書館にまつわる問題

図書館では書籍を無料で見ることができる。より広く、書籍をインターネットで自由に見られるようにする事が「Google ブック検索」などで行われている。しかし、これについては様々な議論がある。以下の⑴〜⑸の概念についてそれぞれ 2〜3 行で、概念の意味と、その概念から見たときにどのような議論があり得るかを述べよ。

⑴ 著作権

⑵ 制約からの解放

⑶ 社会の権威構造

⑷ 無形性（無体性）

⑸ 複製可能性

→解答は p.77

### 35.1.3　不正アクセス禁止法

以下の(a)〜(d)の各項目における行為が、不正アクセス防止法（正式名称「不正アクセス行為の禁止等に関する法律」）における不正アクセス行為に該当するかどうかを、その理由と併せて答えよ。

(a)　違法なわいせつ画像を掲載しているホームページにアクセスする。
(b)　他人を中傷する文章をインターネット上で公開し、アクセスを可能にする。
(c)　他人のパスワードを使って、権限なしにインターネット経由でコンピュータにアクセスする。
(d)　認証機構をもっていない共有サーバにアクセスし、海賊版ソフトウェアをダウンロードする。

→解答は p.78

### 35.1.4　その他語句論述

(1) 情報技術に関連した法律を 1 つ選び、どのように情報技術と関係するか概要をのべよ。
(2) 情報リテラシーとは何を指すか。情報における批判的思考をふくめて答えよ。

→解答は p.78

## 35.2　半自由記述型

出された問に対して、それに合うように自由に述べる問題。情報リテラシーや常識を持って答えることが重要。答えは一意に決まらないが、あまりに極端だったり非常識だったりする考えはやめておいたほうがいい。解答は省略させてもらった。

### 35.2.1　情報格差

(1) 情報技術は国家間、地域間、個人間の格差を縮小するものであると主張する立場に立つものとして、その主張を 8 行以内で書きなさい。
(2) 上の主張に対して反論する立場をとるとして、その反論を 8 行以内で書きなさい。

2007 年 玉井哲雄 個別問題

### 35.2.2　メールのマナー

電子メールを発信する際、情報倫理あるいはマナーの観点から注意すべき点を少なくとも 3 点あげ、その理由を説明せよ。

2006 年 玉井哲雄 個別問題

# 36　情報システムの裏側

次の文章はインターネットの WWW を介して利用できるチケット予約システムにおける複数のコンピュータの動作を順に説明したものである。空欄 A1 から A4 に入るべき言葉を A 群から、空欄 B1 から B4 にはいるべき言葉を B 群からそれぞれ選び、記号で答えよ。

利用者が Web クライアントを使ってチケット情報ページの URL を入力する。[ A1 ] が [ B1 ] する。[ A2 ] が [ B2 ] する。公演情報参照プログラムが公演情報データベースにデータを要求する。[ A3 ] が [ B3 ] する。[ A4 ] が [ B4 ] する。Web クライアントがジャンル別、地域別の購入可能な公演の情報ページを表示する。

■A 群

⒜　公演情報データベース
⒝　Web サーバ
⒞　Web クライアント
⒟　公演情報参照プログラム

■B 群

⒜　公演情報参照プログラムを実行
⒤　公演情報ページを作成し、Web サーバを介して Web クライアントに提供
⒰　Web サーバにサービスを要求
⒠　公演情報参照プログラムに公演情報データを提供

→解答は p.79

## 37　ユーザインタフェース

ディスプレイに線分や円などの複数種類の図形を描くための 2 種類のプログラム A と B があるとする。プログラム A で用いることのできる入力デバイスはキーボードのみであるのに対し、プログラム B で用いることのできる入力デバイスはマウスのみである。それぞれのプログラムを用いてユーザに線分を描かせるには、どのような手順で行なわせるのが適当と考えられるかを答えよ。また、その際のユーザの入力の手間を 2 つのプログラムの間で比較して論ぜよ。

→解答は p.79

# 第 X 部
# 練習問題解答

## 38　コンピュータと計算の知恵

### 38.1　プログラミング

解答例

(1) ア ④　　ウ ③　　(2) イ ⑤　　(3) エ 6　　オ 10　　カ 5　　キ 94

2009 年度のセンター試験の過去問を編集したもの。センター試験のプログラミングの問題は、いつもぐちゃぐちゃしているので、解釈に手間取る。本番の試験でも解釈がめんどくさいコードが出題されることが多々あるので、練習になっただろう。

コードは、以下の様な仕組みになっている。

1. k を 1 から d まで順番に動かして、それが $mp + nq$ として表わされるかを調べる。
2. まず $n = 0$ のときに成り立つかどうか調べる。

4 行目 `if k - int(`$\frac{\texttt{k}}{\texttt{p}}$`) * p = 0 then` は、「$k \div p$ の余りが 0 であれば」の if 文。$k \div p$ の余りが 0 であれば、その商が $m$ になるから、結果的に $m = k \div p$、$n = 0$ で表される場合に、ということである。そのとき、その $k$ を表示して、$u$ を 1 増やせばいいことがわかるから、(**1**)の ア は ④ `k` を表示 が正解。

3. $n = 0$ で成り立たかなったら、$m$ を 1 つずつ増やしていき、該当する $n$ があるかどうかを調べる。

$m$ の値の範囲は、0 から `int(`$\frac{\texttt{k}}{\texttt{p}}$`)` まで。`r ← k - m * p` によって、$r$ に $nq$ となってほしい値を入れる。そのとき、さっきと同様に $r \div q$ の余りが 0 であれば、整数 $n$ を用いて、$r = nq$ を成立させることができる。だから、(**2**)の イ は「$r \div q$ の余りが 0」を表す ⑤ `r - int(`$\frac{\texttt{r}}{\texttt{q}}$`) * q = 0` が正解。

そこでまた $k$ を表示し $u$ を 1 増やすわけだが、その下に `m ← int(`$\frac{\texttt{k}}{\texttt{p}}$`)` という文がある。これはただ while 文をすぐに抜けださせるためのものだから、大きな意味はない。

(**1**)の ウ は、その while 文で、$m$ を 0 から `int(`$\frac{\texttt{k}}{\texttt{p}}$`)` まで順番に試すために $m$ を 1 つずつ増やしていく文が入るはずだから、 ③ `m ← m + 1` が正解。

4. 最後に総数として $u$ の値を表示する。

これでプログラムの完成である。

(**3**)はこれを実行させた時の話。$p = 3$、$q = 7$、$d = 15$ の時は、

1. $3 = 1 \times 3 + 0 \times 7$
2. $6 = 2 \times 3 + 0 \times 7$
3. $7 = 0 \times 3 + 1 \times 7$
4. $9 = 3 \times 3 + 0 \times 7$
5. $10 = 1 \times 3 + 1 \times 7$
6. $12 = 4 \times 3 + 0 \times 7$
7. $13 = 2 \times 3 + 1 \times 7$
8. $14 = 0 \times 3 + 2 \times 7$

9. $15 = 5 \times 3 + 0 \times 7$

の 9 つが $k$ として当てはまるから、［エ］は **6**、［オ］は **10** が正解。

またさっき解説したように、`if k - int(`$\frac{\texttt{k}}{\texttt{p}}$`) * p = 0 then` の下の［ア］は、$n = 0$ で $k$ が表される時に行なわれるから、［カ］は $k = 3, 6, 9, 12, 15$ の **5** 回行なわれる。

$p = 3$、$q = 7$、$d = 100$ の時は、12 以上はすべて $mp + nq$ で表現できることから、［キ］は $(100 - 12 + 1) + 5 = \mathbf{94}$ が正解。

## 38.2　有限状態機械（オートマトン）

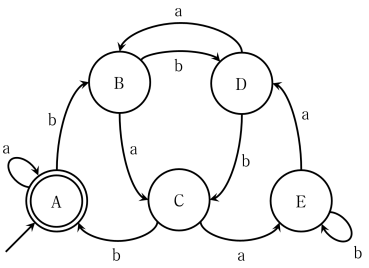
解答例

(1) (a) No　(b) Yes　(c) No　(d) Yes　(e) Yes　(f) 5 の倍数
(2) (g) No　(h) No　(i) Yes　(j) Yes　図は解説参照

2008 年度の田中哲朗の個別問題の過去問を一部改変したもの。

(**1**) 入力された 2 進数が 5 の倍数であるかどうかを判定するオートマトン。(a)〜(e)に関しては、そんなことを考えるよりも、与えられたオートマトンをなぞっていくほうが早い。落ち着いてオートマトンをなぞろう。

(f)に関しては、少々経験面も必要だが、チェック表をながめて「もしや!?」と思えれば最高だ。一応仕組みを解説しておく。




$n$ を自然数とする。今までに入力された 2 進数が $5n$ で表されるとする。ここでオートマトンの状態は A だ。そこに a（0）を入力すると、入力された 2 進数は全体で $10n$ になり、これも 5 の倍数。b（1）を入力すると、入力された 2 進数は全体で $10n + 1$ になり、これは 5 で割って 1 余る数。ここでオートマトンの状態は B だ。

今までに入力された 2 進数が $5n + 1$ で表されるとする。ここでオートマトンの状態は、上より B だ。そこに a（0）を入力すると、入力された 2 進数は全体で $10n + 2$ になり、5 で割って 2 余る数になっている。ここでオートマトンの状態は C だ。b（1）を入力すると、入力された 2 進数は全体で $10n + 3$ になり、これは 5 で割って 3 余る数。ここでオートマトンの状態は D だ。

今までに入力された 2 進数が $5n + 2$ で表されるとする。ここでオートマトンの状態は、上より C だ。そこに a（0）を入力すると、入力された 2 進数は全体で $10n + 4$ になり、5 で割って 4 余る数になっている。ここでオートマトンの状態は E だ。b（1）を入力すると、入力された 2 進数は全体で $10n + 5$ になり、これは 5 で割りきれる数。ここでオートマトンの状態はちゃんと A になっている。

同じように $5n + 3$、$5n + 4$ のときも考えると、A、B、C、D、E それぞれの状態が、今まで入力された 2 進数が 5 で割り切れる、5 で割って 1 余る、5 で割って 2 余る、5 で割って 3 余る、5 で割って 4 余る状態を表しているとわかる。

論理的に考えると長くなるが、これを応用すれば、なんの倍数でも作ることができる。

試しに 6 の倍数かどうかを判定するオートマトンを考えてみよう。

- 余りが 0 → 入力が 0 なら余りが 0 に、入力が 1 なら余りが 1 に
- 余りが 1 → 入力が 0 なら余りが 2 に、入力が 1 なら余りが 3 に
- 余りが 2 → 入力が 0 なら余りが 4 に、入力が 1 なら余りが 5 に
- 余りが 3 → 入力が 0 なら余りが 0 に、入力が 1 なら余りが 1 に

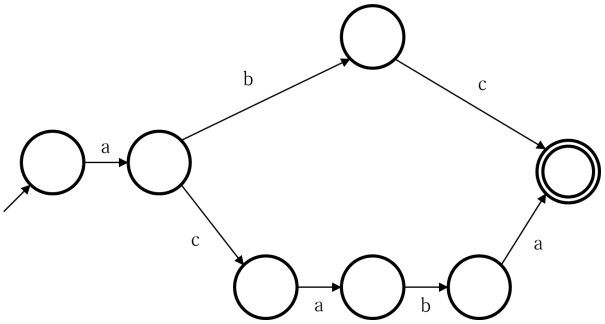
- 余りが 4 → 入力が 0 なら余りが 2 に、入力が 1 なら余りが 3 に（余りが 1 のときと同じ）
- 余りが 5 → 入力が 0 なら余りが 4 に、入力が 1 なら余りが 5 に（余りが 2 のときと同じ）

というふうに順番に考えていく。余りが 1 のときと 4 のときでは、遷移する方向が同じだから、2 つをまとめてしまっていい。余りが 2 のときと 5 のときも同様。すると、余りが 0 のとき、1 および 4 のとき、2 および 5 のとき、3 のとき、の 4 つの状態があればいい。あとは自分で書いてみてね。解答は下[*58]。

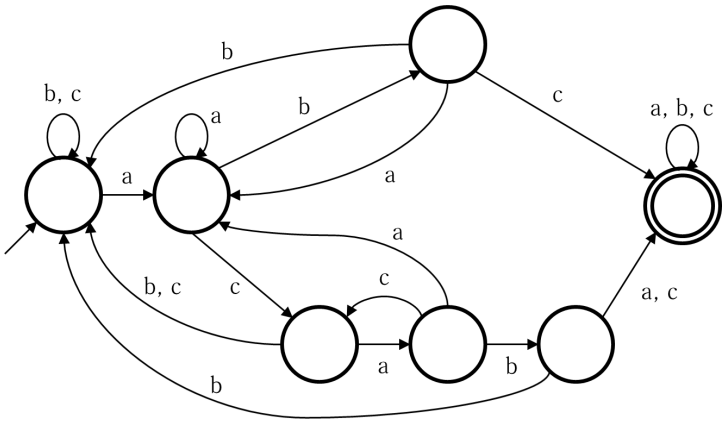
(2) 暗証番号をオートマトンで表す問題。(g)〜(j)に関しては、「abc」と「acaba」が連続して入っていれば Yes、入っていなければ No とすべし。

オートマトンの図を書くのは、結構厄介。一応、おすすめというか、著者がオートマトンを作る際に行った手順で説明する。

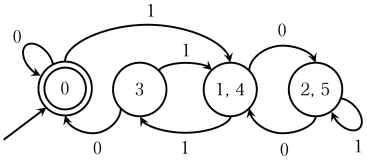
■まず大枠だけ決める　ここで、「abc」と「acaba」の 2 通りの順路があるから、下図のように、とりあえずそれだけ適当に再現する。




■それぞれの状態から遷移する方向を考える　未定義な状態ができないように、うまく配置していく。ある状態から a、b、c を入力したら、それぞれどの状態に遷移するかを丁寧に把握していくのがコツ。以下の図が解答。




[*58] 解答例。状態の数字は、それまでに入力された 2 進数の余りを表している。



## 38.3　論理回路と論理関数

解答例

(1) A 0　B 1　C 1　D 1　E 1　F 1　G 1　H 0　(2)〜(4) 解説参照

2009 年度の共通問題の過去問を編集したもの。珍しく共通問題で論理回路の問題が出た年である。

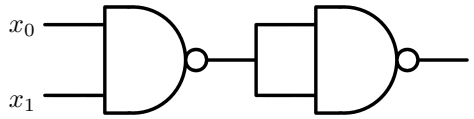
(**1**) これは簡単……なはず。OR は「両方 0 なら 0、それ以外は 1」。NAND は AND の NOT である、と問題文中に書いてあるので、それに従って表を埋めれば、以下のようになるはず。

| $x_0$ | $x_1$ | NOT($x_0$) | AND($x_0$, $x_1$) | OR($x_0$, $x_1$) | NAND($x_0$, $x_1$) | XOR($x_0$, $x_1$) |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | **0** | **1** | 0 |
| 0 | 1 | 1 | 0 | **1** | **1** | 1 |
| 1 | 0 | 0 | 0 | **1** | **1** | 1 |
| 1 | 1 | 0 | 1 | **1** | **0** | 0 |

(**2**) AND を NAND2 つで表現する問題。

$$\begin{aligned}\mathrm{AND}(x_0, x_1) &= \mathrm{NOT}(\mathrm{NOT}(\mathrm{AND}(x_0, x_1))) \\ &= \mathrm{NOT}(\mathrm{NAND}(x_0, x_1))\end{aligned}$$

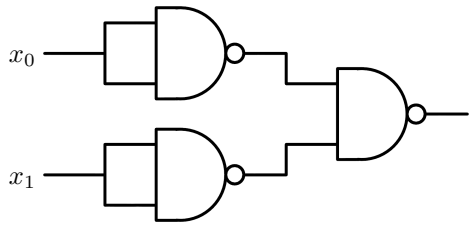
となるから、これを利用すれば以下のような図が書けるはず。




(**3**) OR を NAND3 つで表現する問題。

$$\begin{aligned}\mathrm{OR}(x_0, x_1) &= \mathrm{OR}(\mathrm{NOT}(\mathrm{NOT}(x_0)), \mathrm{NOT}(\mathrm{NOT}(x_1))) \\ &= \mathrm{NOT}(\mathrm{AND}(\mathrm{NOT}(x_0)), \mathrm{NOT}(x_1))) \\ &= \mathrm{NAND}(\mathrm{NOT}(x_0), \mathrm{NOT}(x_1))\end{aligned}$$

となる [*59] から、これを利用すれば、以下のような図が書ける。




(**4**) XOR を NAND で表現する問題。

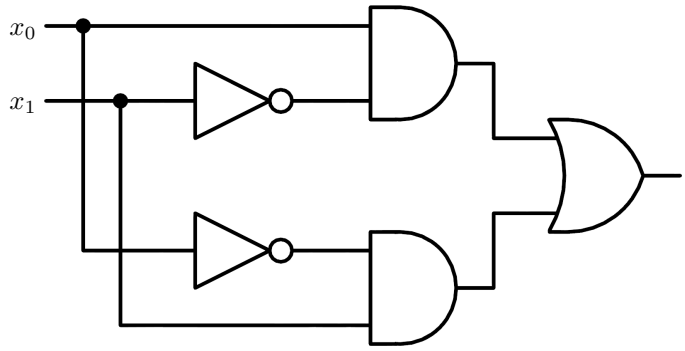
これは非常に厄介。特に最小の 4 つで表現するのはかなり困難。ぶっちゃけできなくてもいい。とりあえず、5 つで表現する方法をまず説明する。

[*59] 1 行目から 2 行目の間で、ド・モルガンの法則を使っている。$\mathrm{OR}(\mathrm{NOT}(x_0), \mathrm{NOT}(x_1)) = \mathrm{NAND}(x_0, x_1)$ という法則。「A でないかつ B でない」は「A または B でない」と同値である、ってヤツ。OR と AND を相互に行き来するにはこれしかないので、覚えておこう。

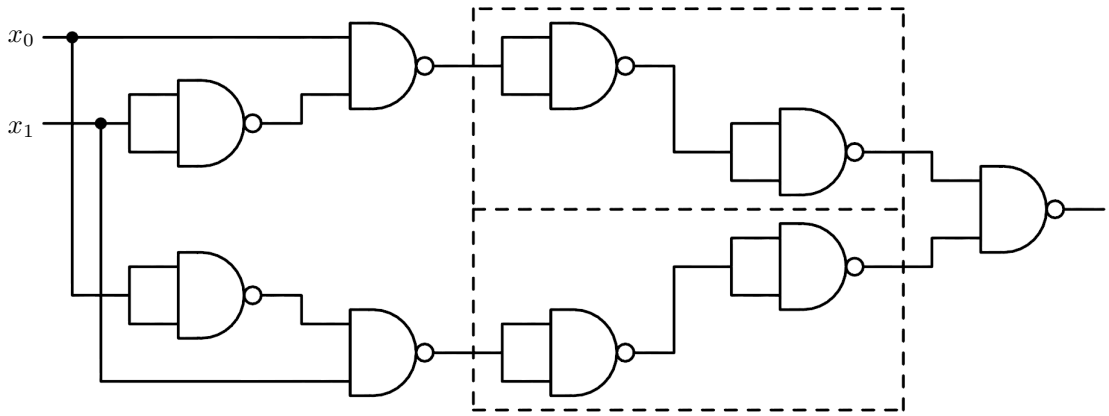
MIL 記法で書く時のことを思い出してほしい。OR と AND だけで書けるような方法を学んだはずだ。これを使えば、

$$\mathrm{XOR}(x_0, x_1) = \mathrm{OR}(\mathrm{AND}(x_1, \mathrm{NOT}(x_0)), \mathrm{AND}(\mathrm{NOT}(x_1), x_0))$$

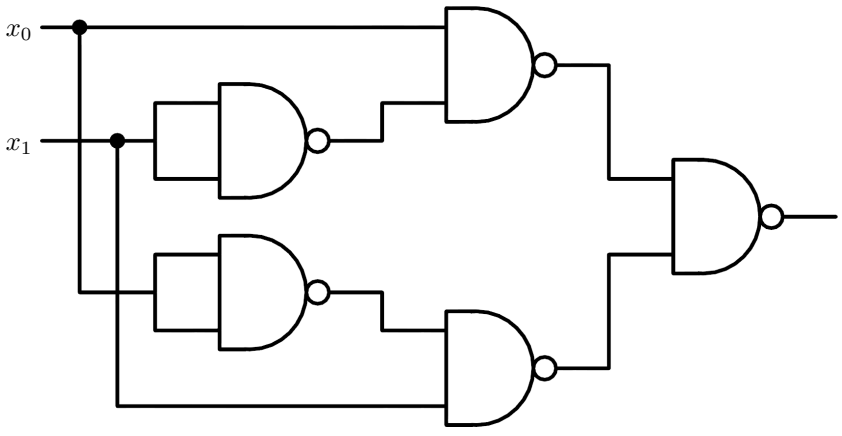
と書き直せるはずだ。これをそのまま図にすると、




となる。この図中の AND、OR、NOT を、今まで導いてきた図で当てはめると、




と表せる。ここで、点線で囲まれた部分が NOT-NOT になっていて、意味のない部分になっているので、この部分を削れば、

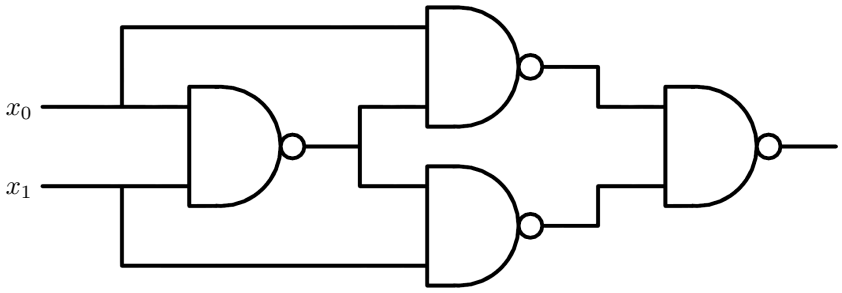



というふうに整理することができる。

さて、4 個のをどうやって表現するか。面倒なので、ブール代数で書いておきます。余裕のない人は理解しなくてもまったく構いません。

$$\begin{aligned}
\mathrm{XOR}(x_0, x_1) &= (x_0 + x_1) \cdot (\overline{x_0} + \overline{x_1}) \\
&= (x_0 + x_1) \cdot (\overline{x_0 \cdot x_1}) \\
&= x_0 \cdot (\overline{x_0 \cdot x_1}) + (\overline{x_0 \cdot x_1}) \cdot x_1 \\
&= \overline{\overline{x_0 \cdot (\overline{x_0 \cdot x_1})} + \overline{(\overline{x_0 \cdot x_1}) \cdot x_1}} \\
&= \overline{\left(\overline{x_0 \cdot (\overline{x_0 \cdot x_1})}\right) \cdot \left(\overline{(\overline{x_0 \cdot x_1}) \cdot x_1}\right)} \\
&= \mathrm{NAND}(\mathrm{NAND}(x_0, \mathrm{NAND}(x_0, x_1)), \mathrm{NAND}(\mathrm{NAND}(x_0, x_1), x_1))
\end{aligned}$$

と書けるので、これを図にすると、以下のように NAND4 つで XOR を表現することができる。




すごいね! こんなの試験時間中に思いつくかよ!

# 39　情報とデジタルデータ

## 39.1　情報量とは & 情報と符号

解答例

(a) $-\log_2 0.7$　(b) 6　(c) 0.3　(d) $-\log_2 0.3$　(e) $-0.7\log_2 0.7 - 0.3\log_2 0.3$
(f) $-\log_2 0.5$　(g) $-\log_2 0.3$　(h) 0.1　(i) $-\log_2 0.1$　(j) 1　(k) $-\log_2 0.1$　(l) 10
(m) $-0.5\log_2 0.5 - 0.3\log_2 0.3 - 0.1\log_2 0.1 - 0.1\log_2 0.1$　(n) より小さい

2009 年度の田中哲朗の個別問題の過去問を編集したもの。
情報量が $-\log_2$(その情報の示す確率) で表されることを思い出せば、至って簡単なはずである。
表を埋めれば以下のとおり。

| 記号 | 出現回数 | 出現確率 | 情報量 | 平均情報量 |
|---|---|---|---|---|
| 0 | 14 | 0.7 | $-\log_{\mathbf{2}}\mathbf{0.7}$ | $0.7 \times (-\log_{\mathbf{2}}\mathbf{0.7})$ |
| 1 | 6 | 0.3 | $-\log_{\mathbf{2}}\mathbf{0.3}$ | $0.3 \times (-\log_{\mathbf{2}}\mathbf{0.3})$ |
| 計 | 20 | 1.0 | | $-\mathbf{0.7}\log_{\mathbf{2}}\mathbf{0.7} - \mathbf{0.3}\log_{\mathbf{2}}\mathbf{0.3}$ |

| 記号 | 出現回数 | 出現確率 | 情報量 | 平均情報量 |
|---|---|---|---|---|
| A | 5 | 0.5 | $-\log_{\mathbf{2}}\mathbf{0.5}$ | $0.5 \times (-\log_{\mathbf{2}}\mathbf{0.5})$ |
| B | 3 | 0.3 | $-\log_{\mathbf{2}}\mathbf{0.3}$ | $0.3 \times (-\log_{\mathbf{2}}\mathbf{0.3})$ |
| C | 1 | **0.1** | $-\log_{\mathbf{2}}\mathbf{0.1}$ | $\mathbf{0.1} \times (-\log_{\mathbf{2}}\mathbf{0.1})$ |
| D | **1** | 0.1 | $-\log_{\mathbf{2}}\mathbf{0.1}$ | $0.1 \times (-\log_{\mathbf{2}}\mathbf{0.1})$ |
| 計 | **10** | 1.0 | | (省略) |

上の省略した(m)の部分には、当然 $-\mathbf{0.5}\log_{\mathbf{2}}\mathbf{0.5} - \mathbf{0.3}\log_{\mathbf{2}}\mathbf{0.3} - \mathbf{2} \times \mathbf{0.1}\log_{\mathbf{2}}\mathbf{0.1}$ が入る。

さて、(n)に関して調べよう。

$-0.7\log_2 0.7 - 0.3\log_2 0.3$ の 2 倍と、$-0.5\log_2 0.5 - 0.3\log_2 0.3 - 2\times 0.1\log_2 0.1$ の大小を比べる。

$$\begin{aligned}
2\times(-0.7\log_2 0.7 - 0.3\log_2 0.3) &= -\frac{1}{10}\times 2\log_2 (0.7)^7\times(0.3)^3 \\
&= -\frac{1}{10}\times\log_2\left(\frac{7^7\times 3^3}{10^{10}}\right)^2 \\
&= -\frac{1}{10}\times\log_2\frac{7^{14}\times 3^6}{10^{20}} \\
&= -\frac{1}{10}\times\log_2\frac{7^{14}\times 3^6}{10^{20}} \\
&= -\frac{1}{10}\times\log_2\frac{49442.4620106921}{10^{10}} \\
-0.5\log_2 0.5 - 0.3\log_2 0.3 - 2\times 0.1\log_2 0.1 &= -\frac{1}{10}\times\log_2 (0.5)^5\times(0.3)^3\times(0.3)^2 \\
&= -\frac{1}{10}\times\log_2\frac{5^5\times 3^3}{10^{10}} \\
&= -\frac{1}{10}\times\log_2\frac{84375}{10^{10}}
\end{aligned}$$

(e) $-0.7\log_2 0.7 - 0.3\log_2 0.3$ の方が大きいから、(m)は(e)の 2 倍より小さいことがわかる。

簡単な問題だが、計算が重いという罠。まあ $7^2 \simeq 50$ とかにしても十分答えは出るだろうから、うまく手を抜いて計算してください。

## 39.2　アナログ情報のデジタル化

解答例

(1) 音楽 CD は、人間の可聴域 20Hz〜20000Hz よりも大きい周波数がナイキスト周波数になるように、20000Hz の約 2 倍である 44.1kHz、つまり 44100 秒に 1 回のペースで、音圧を標本化している。また、人間の聴覚の分解能を十分カバーできるように、深さ 16 ビット、すなわち $2^{16} = 65536$ 段階で量子化している。

(2) 1411200bit ($1.41\times 10^6$bit)

(3) 783216000byte ($7.83\times 10^8$byte)

2009 年度の共通問題の過去問を編集したもの。量子化と標本化が、どっちがどっちかこんがらがらないように注意しよう。

(**1**) 解答の通り。

(**2**) 16[bit] × 44100[回/s] = 1411200[bit/s]

(**3**) 1411200[bit/s] × 74[min] × 60[s/min] × $\frac{1}{8}$[byte/bit] = 783216000[byte]

783216000byte はだいたい 750MB。「あれ?」と思った人もいるかもしれない。そう、CD の容量は 650MB と書かれている（前にこのシケプリでもちらっと述べた)。本来の容量より 100MB 分多く音が収録できているようにみえる。

実は、本来の CD の容量は 750MB 入るのだが、デジタルデータを収録する際は、途中で誤りが起きたら困るので、誤り訂正用の 100MB が確保されている。そのため、デジタルデータを入れるときには 650MB 分しか入れられない。だが、音楽データを入れる際には別にそんな少しくらいの誤りは気にしないし、それを気にしているくらいだったらもっと音楽を入れられたほうがいいということで、めいっぱい 750MB まで入るようになっている。

### 39.3　文字の符号化

解答例

(ア) JIS　(イ) 文字化け　　誤り：標本化 → 標準化（統一化）

2010 年度の共通問題の過去問を抜粋して編集したもの。特に解説はいらないだろう。わからなかったら、本編の該当ページを参照すること。

## 40　データモデル

解答例

(1) (a) (c) (e)　(2) 解説参照

2008 年の共通問題の過去問から。ここで紹介するのはあくまでも一例で、補った仮定によっては答えが変わってくる場合がある。さて、順番に見ていこう。

(**a**) 科目 B と科目 C を履修するために、科目 A が必要で、科目 D を履修するために科目 B と科目 C が必要だとすれば、枝分かれ構造が収束してしまっているので、これは階層モデルとは言えない。よって、階層モデルは適切でない。

(**b**) 1 人の人に 2 人以上の上司がいるとは、問題文からは考えられないので、階層モデルが適切である。

(**c**) インターネットにおけるコンピュータやルータは、それぞれ複数のコンピュータやルータと繋がっており、クモの巣状になっていることから、階層モデルは適切でなく、ネットワークモデルが適切である。

(**d**) 問題文より、操作を下位の段階段階に分けて行っているので、これは木構造であると伺えるから、階層モデルが適切である。

(**e**) 自分の親が、お互いいとこ同士であったとすると、その二人の祖父母は一致する、すなわち自分の曾祖父母が一致するから、自分の曾祖父母への最短経路が 2 つ存在するため、これは階層モデルとしては成り立たない。

こんな感じ。

このように、あるノード（要素）から別のノード（要素）に辿る方法が、最短経路で 2 つ存在するときには、階層モデルが使えない。迷った時には、実際に木の形のモデル図を書いてためしてみるべし。

## 41　インターネットの仕組み

解答例

(1)　誤り　お互いのコンピュータは直接ケーブルでつながれている必要はない。
　説明　送信者が電子メールというメッセージを送る際、そのメッセージはパケットに分割されて、複数のルータを通ってインターネット上を移動して、相手のメールサーバまで届く。そこに受信者がアクセスしてメールを受信することで、メールが届く。(111 文字)

(2)　誤り　鍵は双方が公開するのではなく、受信者のみが自分の持っている秘密鍵とペアの公開鍵を公開する。
　説明　公開鍵暗号方式では、受信者があらかじめ公開鍵と秘密鍵を作っておき、公開鍵のみを一般に公開する。この 2 つの鍵はペアになっており、公開鍵から秘密鍵を推測することは

できない。送信者は公開鍵を用いて平文を暗号化し、それを受信者が秘密鍵で復号することで、他人に内容を見られないように通信を行うことができる。(148 文字)

⑶ 誤り $a \times 10$ 秒より長くかかることも十分考えられる。

説明 自分のコンピュータと Web サーバとは、ルータにより繋がれた多数のネットワークを通り、メッセージをパケットに分割することで、通信している。通信時間は、サーバの位置のみならず、パケットのサイズなどでも決まってくるため、ファイルサイズと通信時間は一概に比例するとは言えない。(134 文字)

⑷ 誤り 接続者の国、地域、所属とドメイン名は大きく関わっている。

説明 ホスト名の一部であるドメイン名は `lumpofsugar.co.jp` などとなっているが、`.jp` により国が、`.co` により企業であることなどがわかる。その国や企業の管理する DNS に、ホスト名に対応した IP アドレスを問い合わせていくことによって、DNS の分散的・階層的な管理が可能となっている。(143 文字)

2011 年の共通問題の過去問から。解答は説明を読めばだいたい分かるだろう。本番の試験ではここまで細かく説明する必要はないかも。「?」と感じたところがあったら、本編の方を Check しよう。なお「平文」は普通「ひらぶん」と読み、「へいぶん」はマイナー。

# 42 情報と社会

## 42.1 主題論述型

### 42.1.1 Winny にまつわる問題

解答例

⑴ 著作権者 の許可なく、ハードディスク 内に保存されている著作物を ネットワーク を通じて自由にダウンロードできる状態にしたため。

⑵ Winny はファイル共有ソフトである。Winny は、ファイルの共有がスムーズである反面、著作物が違法に出まわりやすくなっている。Winny 開発者はこのことを理解しながらもこのソフトウェアを開発し、インターネット上で広く公開したため、これは著作権侵害の幇助と言える。

⑶ Winny 開発者はあくまでも、著作権法に反さない枠内での、有用なファイルの共有を目的として作ったはずである。著作権を侵害したファイルの共有の責任は、あくまでもアップロードした者にあるのであり、Winny の開発だけで開発者を罪に問うことはできない。

2012 年共通問題より。

### 42.1.2 Google 図書館にまつわる問題

解答例

⑴ 著作権とは、全ての著作物の作者が持つ、その著作物を管理する権利、すなわち他人に許可なく複製や再配布をされない権利である。書籍が無料でインターネット公開されてしまうと、著作権者が得る経済的利益が少なくなる可能性の他、データとして簡単にコピーされてしまうなど、著作権が侵害される恐れがある。

⑵ 制約からの解放とは、著作物が紙などの物理的媒体に縛られずに、世界中に短い時間かつ低コストで伝達されることである。これにより、書籍の内容をオンラインで閲覧できるようになるため、利用者は書店や図書館へ出向く必要がなくなるが、書店の売り上げや図書館の利用者が減少する可

能性がある。
⑶ 社会の権威構造とは、社会の体制や権力を支える構造のことである。書籍の内容がインターネット上に無料で公開されることにより、出版業界の持つ権益の崩壊や、検閲機能の低下による文化や宗教への影響など、社会の権威構造を揺るがす事態が想定される。
⑷ 無形性とは、電子データの形として存在する著作物は物理的な媒体を持っていないということである。これによって人々の「著作物を所有している」という感覚が希薄になり、著作物の不正ダウンロード・アップロードなどの不正利用に対して罪悪感を感じにくくなってしまっている。
⑸ 複製可能性とは、データ複製の容易さを表す概念である。デジタルデータとしての著作物からは、オリジナルと「完全に一致する」データを容易に複製できることができるため、本物と同じ品質の不正コピーを閲覧して満足してしまう人が出てくる危険性があり、著作権の侵害が懸念される。

2009 年共通問題より。時間の関係で、only my information からそのまま引用させてもらったが、それぞれの概念に対して「光の面」と「影の面」を両方提示するのが最もよい答えだろう。

### 42.1.3 不正アクセス禁止法

解答例

⑴ 不正にわいせつな画像にアクセスしたわけではなく、公開されているものにアクセスしているので、**該当しない**。
⑵ 情報倫理や個人情報の面からすれば問題ではあるが、不正なアクセスではないので、**該当しない**。
⑶ 本来自分がアクセスできないところを、許可無くアクセスしているため、これは**不正アクセスに該当する**。
⑷ 公開されているデータにアクセスしているため、著作権的には問題だが、不正アクセスには**該当しない**。

2008 年共通問題より。

不正アクセスというのは、本来自分がアクセスしちゃダメなところにアクセスすること。他人のパスワードを盗み見て、それを使ってパソコンの中身見たらそれも不正アクセスよ!

わいせつ画像ね。児童ポルノ法が改正されるだのうんだのという議論が一時期巻き起こっていたけど、まあ、わいせつ画像を見るのは自由ですよね。だけど、わいせつな行為をするのはアカン。

### 42.1.4 その他語句論述

解答例

⑴ 著作権法
情報技術の発達によって、著作物のあり方、人々の著作物に対する考え方は大きく変化した。著作物がデータとして扱われるようになったことが原因で、人々の著作物を所有しているという感覚は薄くなりつつあり、著作物のデータを不正に入手、譲渡することへの抵抗が少なくなってきている。著作物のデータを不正に入手、譲渡することはもちろん著作権法に反しているが、インターネットの匿名性や、違反している人間があまりに多いことから、実際はほとんど取り締まれていない。この事実に対して、徹底して法で取り締まるべきだという考えと、現実に合わせて法を変えていくべきだという考えの 2 つの考えがある。
⑵ 情報リテラシーとは、ただ単に情報機器を扱える能力を指すのではない。与えられる情報をただただ受動的に受け取るのではなく、多くの情報の中からどの情報が正しいのかを論理的に判断し、

情報の取捨選択をする能力のことを指す。また情報リテラシーには、客観性に基づいて情報を主体的に活用、編集し、発信していく能力も含まれる。

2007 年共通問題より。これも時間の関係により「only my information」から引用。(1)では、他に個人情報保護法、不正アクセス禁止法なども書けそうだが、著作権が一番ラクそう。

### 42.2　半自由記述型

解答略。

## 43　情報システムの裏側

解答例

A1 (c)　A2 (b)　A3 (a)　A4 (d)　B1 (う)　B2 (a)　B3 (d)　B4 (b)

2009 年共通問題より。合ってなかったら、当てはめて読んで納得しよう。

## 44　ユーザインタフェース

解答例

■線分を書く方法

プログラム A　始点と終点の座標、もしくは始点の座標とその点からの線分の向きと長さを入力する。

プログラム B　始点で左クリックボタンを押し、ボタンを押した状態で終点までポインタを動かし(そのままドラッグして)、終点でポインタを離す。

■ユーザの入力の手間の比較　プログラム A は精密に座標を指定して描けるので、正確な線分を引くことができるが、逆にその精密な座標を入力するのに手間がかかる。一方、プログラム B は直感的に線分を引くことができ、精密さを求めないならプログラム A よりは手間が少なく行える。しかし、精密な線分をひこうとすると微妙な傾きや長さのズレがでてしまい、何度もやり直すハメになるため、この場合はプログラム A の方が手間が少ない。

2008 年共通問題より。プログラム A が CUI で、プログラム B が GUI であることがわかっただろうか。プログラム B は、クリックして始点を選択、カーソルを動かしてもう一度クリックして終点を選択、などでもよい。

ちなみに TEX で直接図を書くときもプログラム A みたいなことをしなきゃいけないのでタイヘン……。

# あとがき

ここまで全部読んだ人、お疲れさま。
一部だけ読んでここまで辿り着いた人。時間があったらもうちょっと読もう。
何も読まずにあとがきだけ読もうとしてる奴。垢消せ。

内容はどうだっただろうか。なるべくわかりやすくしたつもりではあるが、わかりにくかったら、問い合わせてもらえるとうれしい。今後のシケプリ作りの参考にさせていただく。

東京大学に憧れ、進振りに希望を抱き、未来を楽しみに待っていたあの頃。今となっては、ただ毎日をルーチンワークのようにこなし、授業を適宜サボり、可と不可の間をさまよいながら期末試験をくぐり抜ける毎日だ。入ってから気付く東大の闇。
人生はそんなものである。光だと思っていたものが、まったくの闇だったりする。
でも逆に、闇だと思っていたものが、あとから見たら明るく見えるかもしれない。

君たちはこれから時代を作る者たちだ。荒れ狂う波にもまれながら、未来を作っていくのだ。残念ながら、その中で「情報」は大きな役割を持つ。もしいつか、何かに困ったとき、このシケプリが役に立ったら、嬉しいものである。

さあ。もうすぐ夏休み。残る期末試験を乗り越えて、飛び越えて、9 月に控える期末試験たちの対策を練りながら、充実した 8 月を過ごしてほしい。

そして輝ける未来を歩いて行ってほしい。
君たちの活躍を心から祈っている。

# 索引

# Informatic Synthesis -2013 Summer-

2013 年 7 月 29 日

編者　　イショティハドゥス
執筆者　イショティハドゥス、ミヤダイ
挿絵　　イショティハドゥス



本プリントに関するお問い合わせは、Twitter アカウント @Ishotihadus までお願いします。

編者公式サイト：https://sites.google.com/site/ishotihadus/