情報頻出分野徹底メタシケプリ

共通問題

第2章

**・シャノンの標本化定理**

標本間隔t0で標本化したとき、正しく復元できる精度は標本間隔(1/2)t0

○CD：周波数44.1kHzで記録→上限22.05kHzで復元

　　　 44.1kHzの間隔で標本化、16ビットの精度で量子化

第3章

**・情報量**

**情報量 ＝ log2(事前の場合の数 ÷ 事後の場合の数) ＝－log2(確率)**(単位:ビット)

※情報の加法性を満たす

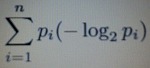
場合の数が大きく減るほど、また珍しい情報であるほど、情報量も大きくなる

　二者択一（場合の数２→１）の時に情報量が1.000になる

などの性質をもつ

情報量の平均(期待値)を平均情報量と呼ぶ

すなわち事象Ai (i=1,2,･･･,n) が起こる確率がそれぞれpi(i=1,2,･･･,n)　であるときの平均情報量は

である

**・共通鍵暗号と公開鍵暗号**

○共通鍵暗号

　暗号化と復号化に同じ鍵を用いる

しかし第3者に暗号鍵が漏れた場合暗号の内容がわかってしまう

○公開鍵暗号

　　暗号化の鍵は誰でも入手可能にしておく … 公開鍵

　　復号化の鍵は自分だけが持っておく … 秘密鍵

こうすると誰でも公開鍵を使って暗号化したメッセージを送ることができるが、そのメッセージを読めるのは秘密鍵を持つ受信者だけである

第3者にメッセージを盗み見られても内容はわからない

**・情報通信**

○プロトコル

　お互いの意図を理解するための約束事をプロトコルという

○サーバーとクライアント

ネットワークどうしを接続する中継機器をルータと呼ぶ。

通信において情報を要求する側をクライアント、要求に応じて情報を提供する側をサーバーと呼ぶ。サーバーとクライアントのやりとりは以下のように行われる。

1. クライアントが相手サーバーのIPアドレスを調べる
2. クライアントはメッセージをパケットと呼ばれる細かい単位に分割し、パケット毎に相手のIPアドレスに向けて送信する
3. 各パケットはまず同一ネットワーク内のルータに届けられる。ルータはそれと同じネットワーク内の別のルータにパケットを届ける
4. 順次ルータを回ってサーバーに辿り着いたパケットは正しい順序に並び替えられ元のメッセージが復元される
5. サーバーがメッセージを受け取る
6. サーバーからクライアントへと同じ手順で情報が送られる

パケット交換方式では1パケット送信時間以上は回線を占有しないので回線上にさまざまな通信のパケットが混在でき、回線の利用効率が高い。一方で、回線が混雑しているときは通信速度が落ちるため通信速度の保証が難しい。

○階層プロトコル

インターネットではTCP/IPと呼ばれるプロトコル群が用いられている。このプロトコル群は階層的な構造をしている。

階層モデルは4階層から成っている。ネットワークインターフェース層(イーサネット)、インターネット層(IP)、トランスポート層(TCP)、アプリケーション層(主にHTTP)である。

ルータ間のネットワーク内通信においてはイーサネットが用いられる。クライアントとサーバーとの間のネットワーク通信では1つ下の層のIPと呼ばれるプロトコルが用いられる。この層ではIPパケットを相手のマシンに送るためにどのルータに送るのかが決定される。分割されたパケットを元通りに組み立てるためにさらに1つ下の層のTCPが用いられる。さらに下のHTTPはアプリケーション間の通信に関するプロトコルである。これらのプロトコルは担当の階層以外の通信に関しては関与しない。このようなプロトコルを階層プロトコルという。階層プロトコルにはカプセル化が用いられる。先頭のヘッダと末尾のトレーラで下位のプロトコルを挟み込むことで、上位のプロトコルの通信時には下位のプロトコルは関係なく通信を行うことができるのだ。

第4章

**・階層モデルと木構造**

樹形図のように枝分かれのある構造を木構造という。(樹形図が木構造に含まれるかはｼﾗﾈ)このように階層的な構造を階層モデルという。

○階層的ファイルシステム

コンピュータのファイルシステムも階層的になっていることが多い。メリットは以下。

1､フォルダの中にサブフォルダを作って細かい仕分けができる

2､フォルダを分けておけば、それぞれのサブフォルダが混ざることはない

3､フォルダをたどることで、すべてのフォルダとファイルを重複なく処理できる

4､サブフォルダ以下の内容を、そのサブフォルダで代表させることができる

例えば、階層構造の中でサブフォルダを移動すれば、サブフォルダ以下をまとめて移動させることが出来る

住所やドメイン名なども木構造をしている。階層的ファイルシステムとメリットはほぼ同じである。(各区がそれぞれの下の住所を独立に管理できる･･･など)

第5章

**・逐次式処理のプログラミング**

処理によって変化させる値を変数、変数の値を設定していく操作を代入という。数学の変数や代入とは異なる。(数学の変数はパラメタと呼ぶべきだ　by教科書)

(プログラム中だと　x=x+1　とかなるしね　数学と一緒に考えるとイミフ　by俺)

○条件付き処理 … ～ならば…という処理を行う、という操作

　　　　　　　　　　if 条件

　　　　　　　　　　　then 条件が成立した場合に行う処理

　　　　　　　　　　　else そうでない場合に行う処理

　　　　　　　　　　endif

　　　　　　　　　という形式で表すことにする（elseは省略が可能）

しかし添え字付き変数に関して条件付き処理を繰り返すと冗長なプログラムになってしまう。同じ条件付き処理を繰り返すときは反復処理を用いるとすっきりする。

○反復処理 … ある条件が成立している間、指定された処理を繰り返す操作

　　　　　　　while 条件 do

　　　　　　　　繰り返し実行する処理

　　　　　　　done

　　　　　　　　という形式で表すことにする

ここで添え字をつけた値を配列という(数列みたいなもんかな？)

第6章

・アルゴリズム

　ある正の実数xが与えられたときに、２乗するとxに近くなる正の実数yを精度δで求める。つまり、|－y|＜δとなるようなyを１つ求める。

教科書の例を元に2つのアルゴリズムを紹介する

○反復法

　　　y ← 0

　　　while (y＋δ)2 ＜ x do

　　　　y ← y＋δ

　　　done

　　　return y

　　　※「return y」は「yを解として計算を終了」の意

　　　　 「x←y」　は「xという変数にyを代入」の意

範囲を一定量ずつ狭めている。精度に逆比例した計算回数が必要になる。

**○二分法**

　　　a ← 0

　　　b ← x

　　　while b－a ＞ δ do

　　　　c ←

　　　　　if c2 ＞ x

then b ← c

else a ← c

endif

　　　done

　　　return a

その名の通り範囲を半分ずつに狭めている。計算回数は反復法より効率的なことが多い。

独自問題

**・オートマトン**

共通問題の第6章、有限状態機械と被っている

繰り返される文字列や終了後の遷移などに注意?(よくﾜｶﾝﾈ)

・情報量　パス

**・ED21**

何故か共通問題の過去問にもあった。命令は一応覚える(多分覚えなくてもだいじょぶ)

LOAD n: ｎ番地の内容をアキュムレータにコピーする

ADD n: ｎ番地の値をアキュムレータの値に加える

STORE n: アキュムレータの値をｎ番地に格納する

STOP n: プログラムを終了する

MUL n: アキュムレータの値にｎ番地の値を掛ける

SUB n: アキュムレータの値からn 番地の値を引く

J n : 無条件でｎ番地にジャンプ

JM n : アキュムレータの値が負のときにｎ番地にジャンプ

JZ n : アキュムレータの値がゼロのときにｎ番地にジャンプ

命令は各番地(0から)に収容され基本的に上から処理される。

**・論理回路**

共通問題にも出てた。個人的には最強の敵、ラスボス(ぉ　4年中3年出てやがった。ｳｾﾞｰ

真理値表を書くor完備性のある組み合わせである回路を作る　ができればおｋ

MIL記法：論理素子を線と四角(もどき)で表す方法

以下、各論理関数について真理値表とMIL記法と多少の補足を

○AND(論理積)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| x1・x2 | 0 | 0 | 0 | 1 |

○OR(論理和)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| x1・x2 | 0 | 1 | 1 | 1 |



○NOT

　真理値表はいいでしょう　　　先端の○がビット反転を表す

**※｛AND，OR，NOT｝の組み合わせは完備性を持つ**

○XOR(排他的論理和)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| x1・x2 | 0 | 1 | 1 | 0 |



○EQ(等値)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| x1・x2 | 1 | 0 | 0 | 1 |

　XORの否定

○NAND(否定論理積)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| x1・x2 | 1 | 1 | 1 | 0 |

　ANDの否定　　**それ単独で完備性を持つ**

○NOR(否定論理和)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| x1・x2 | 1 | 0 | 0 | 0 |

　ORの否定　　**それ単独で完備性を持つ**

○NANDの完備性

NANDは単独で完備性を持つ。すなわち、上の7つの回路は全て、NANDのみを組み合わせることで表される。

1､NOT

NOT(x)＝NAND(x,x)

2､AND

AND(x,y)＝NAND(NAND(x,y)､NAND(x,y))

3､OR

OR(x,y)＝NAND(NAND(x,x)､NAND(y,y))

4､XOR

XOR(x,y)＝NAND(NAND(x,NAND(x,y)),NAND(NAND(x,y),y))

5､NOR

OR＋NOT

6､EQ

XOR＋NOT

○1ビット半加算器

2進数での1ケタ＋1ケタを繰り上がり込みで計算する回路　真理値表は以下

|  |  |  |  |
| --- | --- | --- | --- |
| *x* | *y* | *s* | *c*out |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |

真理値表よりs=XOR(x,y)cout=AND(x,y)となることがわかる

○1ビット全加算器

下のケタからの繰り上がりも考えて1ケタ＋1ケタを計算する回路

半加算器2つとORを使用

○nビット全加算器

nケタ＋nケタの計算を行う。1ビット全加算器をnコつなぐ

このように簡単な回路を組み合わせて複雑な回路を階層的に構成する方法を、モジュール化という。