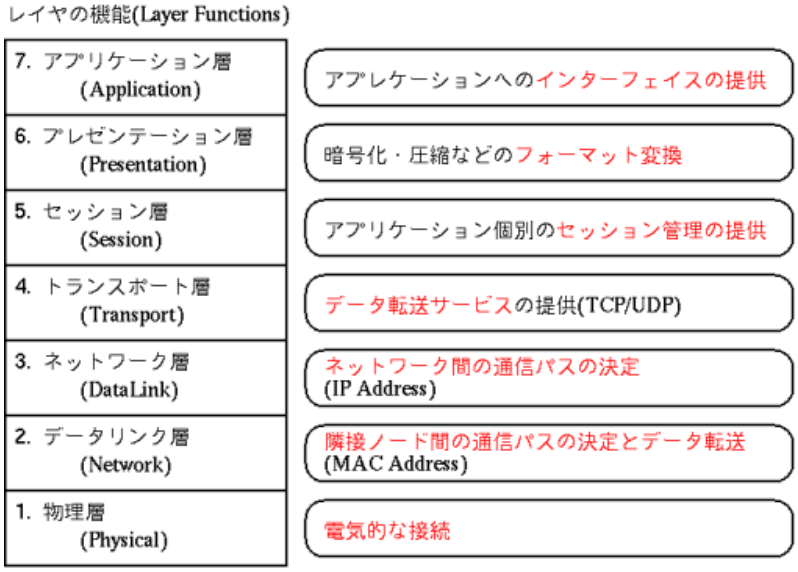
院試　まとめ

*過去問によく出ていたものや、個人的に重要だと思ったものを勝手にまとめました。加筆・修正などは随時自由に行ってください。画像などは著作物のため、学習以外の目的での使用は禁止です。

・OSI7 階層参照モデル



・階層化することのメリットについて

　階層化することにより各階層の作業を独立させることができ、作業の単純化が図れるほか、他の階層に依存せず機能の追加が行える。これは TCP/IP4 層モデルでも同じ。

・各層の役割について（上図に要点はまとまっている）

1) 物理層：
　デジタルデータと電気信号の相互変換やコネクタの形状、ケーブルの特性の規定など物理的な部分を担う。
2) データリンク層：
　同一ネットワーク内の相手との通信方式、通信路を規定するほか、通信路内を流れるデータのエラー検出も行う。
3) ネットワーク層：
　相互に接続された複数のネットワーク（インターネット）を介した相手との通信路の選択や、経路中で宛先として用いるアドレスの管理を行う
4) トランスポート層：

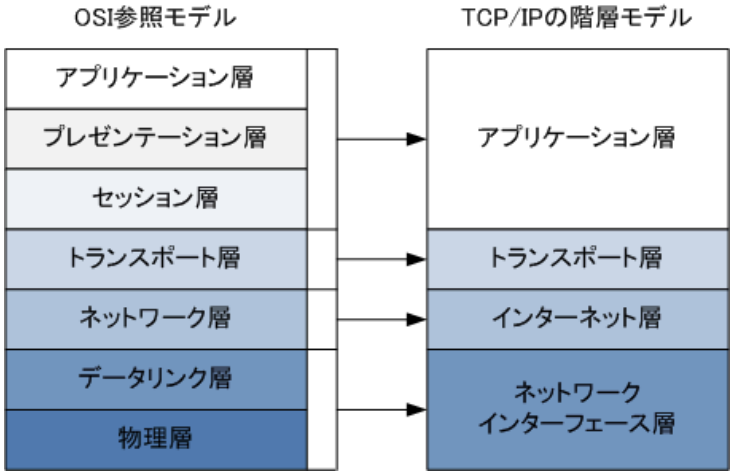
通信されるデータの内容の制御を行う。具体的にはデータを効率よく送るために圧縮するほか、データの誤り訂正や再送要求の制御を行う。

5) セッション層：
プログラム間でのデータの通信を行うための手順を規定し、仮想的な経路の確立や解放を行う。

6) プレゼンテーション層：
データの表現形式について規定し、異なる文字コードを用いているデバイス間の通信でもおなじ文字で表示されるよう補正するほか、データの暗号化と復号もこの層の役割である。

7) アプリケーション層：
アプリケーションごとのデータ形式や処理の基準について規定する。具体的には電子メールやファイル転送、web についてのプロトコルがこの層で規定される。

・TCP/IP4 階層モデル



TCP/IP モデルと OSI モデルの対応は上図のようになっている。

1) ネットワークインターフェース層
実際のネットワークハードウェアが通信を実現するための層。各種イーサネットや無線 LAN はここに属する。またモデムや光回線で特定の相手と接続し、TCP/IP で通信するための PPP はこの層のプロトコルである。

2) インターネット層
OSI モデルのネットワーク層に相当し、複数のネットワークを相互に接続した環境での危機のデータ伝送を実現する。IP はこの層のプロトコルである。

3) トランスポート層

OSI モデルのトランスポート層に相当し、通信を行うプログラム間のデータ伝送を実現し、必要に応じてエラー検出や誤り訂正、双方向の通信路の確立などをおこなう。単にデータ伝送を行うだけの UDP や、信頼性のある双方向の通信を実現する TCP はこの層のプロトコルである。

4) アプリケーション層
OSI モデルのセッション層からアプリケーション層までに相当し、個々のプログラム間でどのような形式、手順でデータをやり取りするかを定める。文字コードや画像の形式、暗号化などのデータ表現形式もこの層で扱う。HTTP のような Web や電子メール等のアプリケーションプロトコルはこの層に属する。

・公開鍵暗号方式

ここでは素因数分解を効率的に行うアルゴリズムがないことを利用した、最も典型的な RSA 暗号についてのみ述べるが、離散対数問題に基づいた DH 鍵交換法など、異なる手法は存在する。

・RSA 暗号

1) 初めに大きい素数p, qを生成しその積pqをnとする。
2) 次に$(p-1)(q-1)$と互いに素な整数k_1を求める。
3) $k_1k_2 \equiv 1 \pmod{(p-1)(q-1)}$となる$k_2$を求める
4) そして公開鍵を(n, k_1)とし、秘密鍵をk_2とする。
5) 平文 m を$m^{k_1} \pmod n$を計算して暗号文 C とする。

この時k_2がわかっていれば$C^{k_2} \pmod n$を計算することで m に戻せるが、C と(n, k_1)から m を求めるには n を素因数分解して p,q に戻し、k_2を求める必要があり、これは現実的な時間では不可能である。なお、下線部の証明にはフェルマーの小定理が用いられる。

*補足：フェルマーの小定理とその証明

p が素数、a が任意の自然数の時、

$$a^p \equiv a \bmod p$$

特に、a が p と互いに素な自然数の時

$$a^{p-1} \equiv 1 \bmod p$$

下のほうの証明）1~p-1 の数値は p で割っても余りとしてそのまま残る。この時 1~p-1 を全て a 倍したものの余りは並べ替えると 1~p-1 になる（a と p は互いに素なため、a 倍しても p の倍数にはならないから）。よって以下の等式が成

り立つ。

$$a^{p-1}(\mathrm{p}-1)! \equiv (\mathrm{p}-1)! \bmod \mathrm{p}$$

したがって、両辺を(p-1)!で割ると

$$a^{p-1} \equiv 1 \bmod \mathrm{p}$$

が導けた。

・ソフトウェア開発の代表モデル
・ウォーターフォールモデル

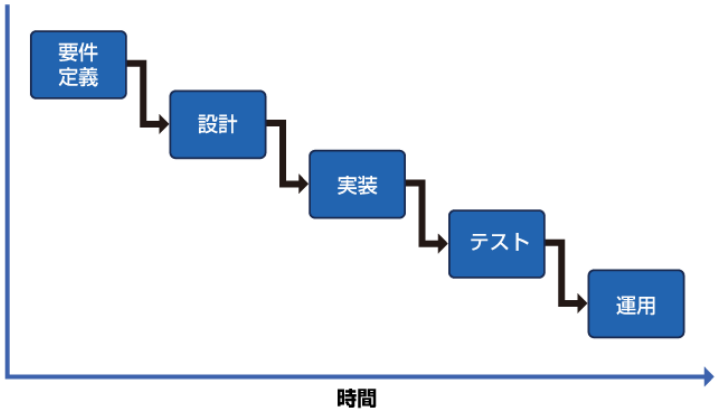


図 1：ウォーターフォール・モデル

開発工程を上図のように数個の作業工程にトップダウンに分割する。その後、これらの工程を上から順に一度で終わらせる計画を立て進捗管理をする。原則として前工程が終了しないと次の工程に進まないことで前工程の品質を確保し、手戻りを最小限にする。このモデルの利点は各工程の終わりにドキュメントが多く作成されるため各工程の進捗管理を厳格にできることである。一方で、前工程の成果物に誤りを含まないことを前提としているため、誤りが含まれるとテスト時にそれが発覚し、大きな手戻りが生じてしまうという欠点もある。そのため、大規模で管理の必要な開発に向いた手法であるといえる。

・アジャイルソフトウェア開発
（試験では今のところ出ていないが、ウォーターフォールと対比されることが多いので補足としてまとめておく。）
アジャイル型の開発手法の多くはイテレーションと呼ばれる短い期間単位を採用することでリスクの最小化を図る。この開発手法では対象をいくつかの小さな機能に分割し、一つのイテレーションで一つの機能を開発する。そしてこの反復サイクルを継続して行うことで一つずつ機能を追加していく。一つ一つの反復は小規模なウォーターフォールモデルととらえられる。特徴として、ドキ

ュメントの作成よりもプロジェクト関係者が必要に応じて即座に意思疎通を行えることを重視する。ここで。プロジェクト関係者には少なくともプログラマと顧客が含まれる。また、アジャイル開発手法では実際に動くソフトウェアこそが最重要なプロジェクト進行尺度であるとする。この二つの性質からドキュメントの量が他の手法と比較して少なくなるという特徴があり、批判材料の一つになっている。

・探索について

探索アルゴリズムでは初めに、解くべき問題を状態と状態変化に分ける。また最初から与えられる状態を初期状態、目的とする状態を最終状態と呼び、初期状態から最終状態に至るまでの状態および状態変化の並びが解となる。

1）知識なし

知識を用いない探索アルゴリズムは問題の性質によらず適用可能なため、汎用的に実装することができる一方、探索空間が大きく、処理にある程度のメモリ量や計算時間が必要になる。

・幅優先探索

木構造やグラフの探索に用いられるアルゴリズム。根ノードから開始し、現在のノードに隣接したすべてのノードを探索する。そして見つかったものについても同様の操作を施す。これにより目的のノードを見つけるか、新しいノードが見つからなくなったところで探索を終了する。アルゴリズムとしては

1）根ノードをキューに加える

2）ノードをキューから取り出し以下の処理を行う

・ノードが目的のものであれば探索を終了

・それ以外ならば子ノードの中で未探索のものを全てキューに追加

3）キューが空ならばグラフ内のすべてのノードの処理が終わっているため、探索を終了し、目的のノードがなかったことを述べる。

4）2)に戻る

・深さ優先アルゴリズム

木構造やグラフの探索に用いられるアルゴリズムで、後戻りする必要が生じる深さまで可能な限り探索を続ける。具体的には最初のノードから始め、隣接するノードのなかで選んでいないものを一つ選ぶことを繰り返し、目的のノードが見つかれば探索を終了、見つからないまま未探索のものがなくなるか、隣接するノードがなくなればひとつ前のノードに戻り隣接ノードを選び直すことを

繰り返して探索する。再帰なしで表現する場合、新しく見つかったノードをスタックにためることで実装できる。
・深さ制限探索
深さ優先探索と同じだが、探索する深さに上限を設けることで無限の経路や感情の経路にとらわれないようにした手法。制限された深さの範囲内での最適経路を求められる。

・反復深化深さ優先探索
この手法では深さ優先探索の制限を徐々に増大させ、最終的に目標状態の深さになるまで反復する。各反復においては深さ優先探索の順序でノードを調べるが、各ノードを初めて調べる順序は幅優先探索と同じになる。

2）知識あり
・最良優先探索
幅優先探索を評価関数に従って次に探索する最も望ましいノードを探索するように拡張した探索アルゴリズム。優先度付きキューを用いて実装するのが一般的である。

・A*アルゴリズム
最良優先探索の一種で、ヒューリスティック関数 h(n)を用いて効率よく探索を行うことを目的としたアルゴリズムである。ここで h(n)は各頂点 n から目的ノードまでの距離の妥当な推定値を返す関数で、探索により内容は異なる。A*の探索効率は h の適切さに大きく左右されるが、h にかかわらず必ず解を発見することは保証されている。これを完全性という。

・ルータの役割
インターネットに接続されたデバイス間で通信を行う際、その中継を担う。まず IP パケットの中から宛先となる受信側の IP アドレスを取り出し、自身のルーティングテーブルを参照して、その中から最も IP アドレスのビット列が一致しているものを選ぶ最長一致によりネクストホップとなる隣接ルータを決定し、その IP パケットを伝送する。これを繰り返し目的のデバイスにデータを送る経路を見つけることがルータの役割である。

・コンピュータの数値表現
・1 の補数・2 の補数
コンピュータには 1 の補数、2 の補数と呼ばれる数値表現がある。

1 の補数は負の数を二進数で表すとき、正の値のビット列を反転させたものとなる。

一方 2 の補数は 1 の補数表現の最下位ビットに 1 足したもの、あるいは与えられた上限のビット列より 1 大きいビット列から正のビット列を引いたものとなる。また、n ビット用いて表現できる場合、1 の補数は$-2^{n-1}+1$~$2^{n-1}-1$までの数値を、2 の補数は-2^{n-1}~$2^{n-1}-1$までの数値を表すことができる。1 の補数は 0 を表すビット列が 2 つ生じてしまうため、2 の補数に比べて表現範囲が狭くなる。補数表現を用いるメリットは、減算を加算と同じ式で表すことができるためである。

ex) $-5_{(10)}$を 1 の補数、2 の補数の順に 4 桁のビット列であらわすことを考えると、

$$5_{(10)} = 0101_{(2)}$$

より、1 の補数では

$$-5_{(10)} = 1010_{(2)}$$

2 の補数では

$$-5_{(2)} = 1010_{(2)} + 0001_{(2)} = 10000_{(2)} - 0101 = 1011_{(2)}$$

となる。

・固定小数点方式

小数を含んで数値をビットで表現するとき、小数点の位置を固定する表現である。固定する都合上表現できる数値範囲は狭くなるが、一方で仕組みは単純であり計算や実装は単純である。

・浮動小数点方式

仮数・基数・指数という三つの数値により、元の数値を(仮数)*(基数)^(指数)という形で表す。小数点の位置が固定されていないため、固定小数点方式に比べ表現できる数値の範囲は広いが、仕組みが複雑になるため実装、計算が難しくなるほか、元の数値を近似的に表すことになるため、情報落ちが起こる。

IEEE754 方式が一般に使われるが中でも単精度(32bit)について補足しておく。単精度では、符号 1bit、指数 8bit、仮数 23bit で表現する(基数は言うまでもなく 2)。この時、

・符号は正で 0、負は 1
・指数は負の場合でも対応できるようにするためバイアス地と呼ばれる数値(ここでは 127)を足す
・仮数は指数部を調節して 1.~~の形になるようにし 1 を自明として省略する

・キャッシュメモリのデータ格納構造

1) ダイレクトマップ方式

コンピュータ内部のメインメモリのアドレスに一定の演算を施すことで直接メインメモリの格納位置を算出する方式。メインメモリのアドレスから

2) フルアソシアティブ方式

メインメモリのアドレスとは無関係にキャッシュメモリの空いている領域にデータを格納する方式。そのため格納位置の衝突が起こらずキャッシュヒット率が高くなる一方で、読出し時の探索コストが大きくなるほか、装置の構成が複雑になるため実装が難しいという欠点がある。

・スラッシング

主記憶装置の容量が少ないにもかかわらず、アプリケーションが巨大なメモリ領域を確保して作業を始めると、仮想記憶システムにおいてページの置き換えが頻発し、コンピュータシステム全体の性能が極端に低下する。この現象をスラッシングという。

・計算量とランダウの記号(O 記法)

アルゴリズムの計算量とは、計算機がそのアルゴリズムの実行に要する計算資源の量のことであり、アルゴリズムのスケーラビリティを示す。

時間計算量はアルゴリズムが問題のインスタンスを解くのに要するステップ数を意味し、入力データの長さの関数としてあらわされる。

空間計算量はアルゴリズムが問題を解く際に必要とする記憶容量を示すものである。

ランダウの記号 f(x)=O(g(x))は変数 x を極限に飛ばした時の f(x)のふるまい(漸近的挙動)を異なる関数 g(x)を目安に記述する目的で用いる。この時 g(x)は $\lim_{n\to\infty} \frac{f(x)}{g(x)}$が存在し、その値が有限で f(x)よりも扱いやすいものが選ばれる。上記の計算量に O 記法を用いることでアルゴリズムの比較評価がしやすくなる。

・ゲーム理論

・ミニマックス法

ゲーム理論において打つ手を決定する際に用いられるルールの一つ。チェスやオセロのような完全情報ゲームを計算器で扱う際に用いられる。大枠としては、想定されるような最大の損害を最小化するように手を選択する。具体的には、現在の局面を評価する静的評価関数で求まる静的評価値だけで次の手を決定することは困難なため、相手の打つ手を先読みする。その際に用いる方策が

ミニマックス法である。内容は以下のようになっている。

1) 相手の打つ手を予測する際、その次に出現する局面の評価値が最小になる手、つまり自分が最も不利になり、相手が最も有利になる手を打ってくると仮定し、評価値を用いる。
2) 自分の打つ手を選択するときは、次に現れる局面の中で最も評価値が高くなるものを選択すればよく、その値を局面の評価値として用いる。

・完全情報ゲーム

各プレイヤーが自分の手番において、これまで各プレイヤーの行った選択についてすべての情報を知ることができるゲーム。また打つ手によりどのような局面が出現するかを場合分けすることでゲームの展開を樹形図にできる。この機をゲーム木と呼び、理論的には最終局面に至るすべての局面を予想することが可能である。

(*おまけ：ナッシュ均衡)

ナッシュ均衡とは、非協力ゲームの解の一種で、どのプレーヤーも自分の戦略を変更することによってより高い利得を得ることができない均衡状態となった解のことである。

・プロセッサの動作

1) フェッチ動作：メモリから命令コードを取りだす
2) デコード：機械語で表現された命令コードを解釈し、プロセッサで用いるマイクロコードに変換する
3) オペランド：数式を構成する要素のうち演算の対象となる値や変数、定数などのこと。プログラミングの分野ではこれに加え、プログラム中の個々の命令・処理の対象となるデータやデータの所在情報なども示す。
4) 実行：デコードとオペランドリードによって得たデータに基づき命令を実行する
5) ライトバック：演算結果を必要に応じてレジスタに書き戻す。

・フォンノイマンボトルネック

ノイマン型の計算機では記憶装置に命令が格納されており、プロセッサが命令を実行するためには必ずバスを通して記憶装置にアクセスしなければならず、アクセス回数が多すぎたりアクセス速度が遅かったりすると計算機の処理全体のボトルネックとなる。

・OS のカーネルについて
・カーネルの役割
　コンピュータのリソースを管理し、他のプログラムがそれらを用いて動作できるようにする。
1) プロセス管理
　プロセスの生成・実行・消滅を管理するほか、プロセス間通信や排他制御もプロセス管理の役割である。また、プロセスへのリソース割り当ても制御する。
2) メモリ管理
　計算機のメモリを管理するもので、単純化するとプログラムの要求に応じてメモリの一部を割り当てる方法と、そのメモリが不要になったときに再利用するために解放する方法を提供する。
3) デバイス管理
　実際に何らかの処理を行うためには、OS は計算機に接続された周辺機器にアクセスする必要があり、周辺機器は開発元が書いたデバイスドライバを通して制御される。デバイスドライバは OS がハードウェアデバイスとやり取りするためのプログラムで、OS に対して何らかのハードウェアを制御・通信するための情報を提供する。
4) システムコール
　オペレーティングシステムの機能を呼び出すために使用される機構のこと。

・CPU の特権モードと非特権モード
　CPU が実行できる操作には通常制約があるがカーネルモードにおいて CPU はそのアーキテクチャのすべての操作が可能である。

・シャノンの情報源符号化定理
　情報源 S は任意の一意復号可能な r 元符号で符号化する場合、その平均符号長 L は

$$\frac{H(S)}{\log_2 L} \leq L$$

を満たす。また任意の正数$\varepsilon > 0$について平均符号長 L が

$$\mathrm{L} < \frac{H(S)}{\log_2 L} + \varepsilon$$

となる r 元瞬時符号を作ることができる。

・エントロピー

確率変数 X がとりうる値が$x_1, x_2, \ldots, x_M$とし、X がそれぞれの値を取る確率が$p_1, p_2, \ldots, p_M$であるとき X の平均情報量$\mathrm{H(X)} = -\sum_{i=1}^{M} p_i \log_2 p_i$を X のエントロピーと呼ぶ。

また確立 p で生起する事象が起きたことを知ったときに得られる情報量 I(p)を事故情報量と呼び、

$$I(p) = -\log_a p \ (a\text{は}a \neq 0 \text{ の定数})$$

と定義する。a=2 の時の単位はビットである。

・ハフマン符号とハフマン木

Alphabet	Probability	Tree of code	Code word
A	0.5		1
B	0.2		01
C	0.1		0011
D	0.08		0010
E	0.05		0001
F	0.04		00001
G	0.02		000001
H	0.01		000000

ハフマン記号はコンパクト符号である。コンパクト符号とは一記号ずつ符号化する際その平均符号長が最小となるような符号である。

・ムーアの法則

「半導体の集積率は十八か月で二倍になる」という半導体業界の経験則。つまり、半導体内のトランジスタ数が十八か月で二倍になるということであるから、CPU の性能が十八か月ごとに二倍になると同時に、一トランジスタ当たりのコストが半分になることを表している。

・マルコフ過程

マルコフ性を持つ確率過程のこと。つまり、未来の値が現在の値のみで決定され、過去の挙動とは無関係であるという性質を持つ確率過程のことである。

・不偏標本分散

$\frac{1}{n-1}\sum_{k=1}^{n}(x_k-\bar{x})^2$を不偏標本分散という。ここで$\bar{x}$は$\frac{1}{n}\sum_{k=1}^{n}x_k$である。不偏標本分散の期待値は母集団の分散を一致する。

・尖度

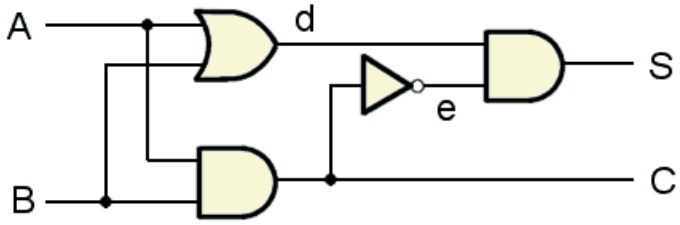
確率変数の確率密度関数や頻度分布の鋭さを表す数値。四次のモーメントである。値から 3 を引く定義があるが、これは正規分布の尖度を 0 とするためのものである。

・プロセッサの構成要素と機能

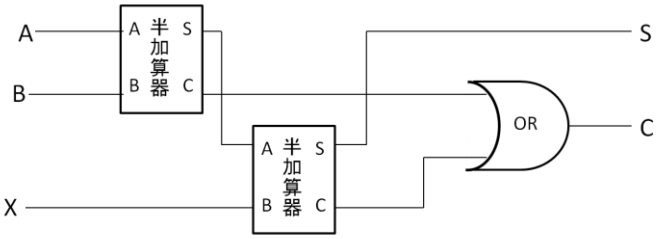
1) バス：CPU などの中心機器と周辺機器や制御部をつなぎ、命令や結果の伝送を行う。
2) 制御装置：実行される命令のアドレスをもとに主記憶装置に格納されたプログラムから命令を取り出して解読し、命令の実行に必要な信号を入力装置・出力装置・演算装置などの各装置に送り制御する。
3) レジスタ：計算結果を一時的に保存したり、RAM・ROM などのメイン江守を読み書きする際のアドレスを保持したり、プロセッサや周辺機器の動作状態の保持・変更を行う。
4) 演算装置：主記憶装置からデータを取り込んで、演算命令を制御装置から受け取り、演算結果を主記憶装置に返す。

・半加算器・全加算器

半加算器

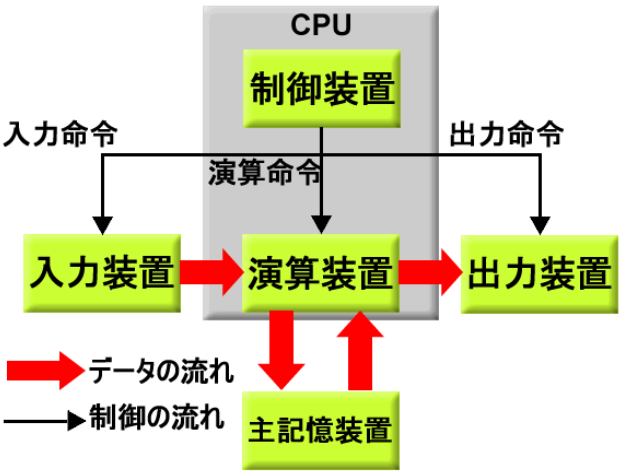


全加算器



A	B	X	S	C
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

・ノイマン型コンピュータの概念図



・複雑ネットワーク

複雑ネットワークとは実世界の構造を頂点と辺で構成されるグラフとしてとらえたもので、このグラフは一般に非常に大規模なものとなる。また、このグラフは次の三つ性質を持つことが知られている。

1) スケールフリー性

一部の頂点は多数の頂点と辺でつながりを持つ一方で、大多数の頂点はごく一部の頂点としかつながりを持たず、次数は低いという性質。

2) スモールワールド性

任意の二つの頂点間の距離が、船体のネットワークの複雑さにもかかわらず小さい値となっていることを表す性質。

3) クラスタ性

複雑ネットワークを構成する頂点では、二つの頂点が辺でつながっているとき、隣接ノードに共通のものを多く持つという性質がある（三つのノードで三角形を形成する形になっているということ）。また、次数 k の頂点 v のクラスタ係数は v の隣接転換の変数を$\frac{k(k-1)}{2}$で割ったものである。

・判別分析とクラスタ分析

1) 判別分析

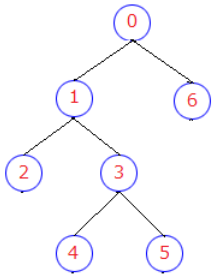
既に分類の分かっているデータが存在するとき、未知データがどの属性に分類されるかを判別するための関数を求めることを目的とする分析手法。ゆえに事前に分類の分かるデータが与えられていない場合に適用することはできない。

2) クラスタ分析
分類の与えられていないデータが存在するとき、似た性質のものを集め、クラスタと呼ばれる集合を形成するようにデータをいくつかのクラスタに分ける分析手法。各クラスタの持つ性質を知ることが目的である。

・強化学習について
1) 方策：どのような行動をとるかを記述したルール
2) 状態価値関数：状態について数値で評価するための関数で、その状態にあることで将来手に入る報酬の期待値を算出する。

強化学習とは状態価値関数の値に基づき一連の行動をとることで得られる報酬が最大になるような方策を学習することを目的とする手法である。

・木構造の探索順序（ソート関連で頻出）
今回は以下の木を考える。



二分探索木の探索手法には一般に以下の三つがある。
1) 行きがけ順：根、左、右の順
サンプルコードは

```
void ahead(Tree t){
    if(t==NULL) return;
    printf("%d¥n", t->value);
    ahead(t->left);
    ahead(t->right);
}
```

出力は 0,1,2,3,4,5,6 の順になる。
2) 通りがけ順：左、根、右の順
サンプルコードは

```
void through(Tree t){
    if(t==NULL) return;
    through(t->left);
    printf("%d¥n", t->value);
```

```
        through(t->right);
    }
```
出力は 2,1,4,3,5,0,6 の順になる。

3) 帰りがけ順：左、右、根の順

サンプルコードは

```
    void back(Tree t){
        if(t==NULL) return;
        back(t->left);
        back(t->right);
        printf("%d¥n", t->value);
    }
```
出力は 2,4,5,3,1,6,0 となる。

・グラムシュミットの正規直交化（3 次元）

$\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$を正規直交化する。

1) $\boldsymbol{u}_1 = \frac{\boldsymbol{a}_1}{|\boldsymbol{a}_1|}$

2) $\boldsymbol{v}_2 = \boldsymbol{a}_2 - (\boldsymbol{a}_2\boldsymbol{u}_1)\boldsymbol{u}_1$

3) $\boldsymbol{u}_2 = \frac{\boldsymbol{v}_2}{|\boldsymbol{v}_2|}$

4) $\boldsymbol{v}_3 = \boldsymbol{a}_3 - (\boldsymbol{a}_3\boldsymbol{u}_1)\boldsymbol{u}_1 - (\boldsymbol{a}_3\boldsymbol{u}_2)\boldsymbol{u}_2$

5) $\boldsymbol{u}_3 = \frac{\boldsymbol{v}_3}{|\boldsymbol{v}_3|}$

以上により求まったu_1, u_2, u_3が正規直交基底となる。

・情報セキュリティの三要素(CIA)

セキュリティシステムでは以下の三要素が要求される。

1) 秘匿性：データの暴露がないこと
2) 完全性：データを改変されることがないこと
3) 可用性：データが利用不能にされないこと

・構造化プログラミングについて

・GOTO 文：機械語の分岐命令に対応した制御文

・スパゲッティプログラム：GOTO 文が不規則に並んでおり、全体像の把握しにくいプログラム

・構造化定理：プログラムをフローチャートで表現した際、どのようなフローチャートも三

つの基本構造

1) 順次

2) 選択

3) 反復

の組み合わせにより等価な構造化フローチャートに変換できる。構造化フローチャートとは、図中に線の考査を含まず、出入り口が一つのものを指す。

*実際には後判定反復及び return 文を含めてもよい。

・プログラムの正当性

アサーションとポテンシャル関数を付ける。これらにより各部で満たす条件を表記。

1) 事前条件：スタート時点での条件

2) ループ不変条件とポテンシャル関数 -> ループ時の条件

3) 事後条件：終了時点で成り立つ条件

・アサーション：プログラム変数の間に成立する関係を表す命題。

制御がその位置に達したとき恒真となる。特定の位置で定義される。

・ポテンシャル関数：プログラム変数の値を用いて整数値を返す関数。値は常に非負。

・ループ不変条件：ループ部分で変化せず、常に満たされる条件。

プログラムの正当性とは、事前条件を満たす任意の入力に対し次の二つの条件を満たすこと。

1) 部分正当性：もし実行が停止すればその時点で事後条件が成り立つ。

2) 停止性：必ず実行が停止する。

・部分正当性の証明

プログラムの各ループの少なくとも一か所にアサーションを書く。

アサーションA_iを書いた位置 i からアサーションA_jを書いた位置 j に至るすべての経路ごとに i でA_iが真のとき、経路に沿ってプログラムを実行して j でA_jが真となることを示す。

*ただし隣接するアサーションか自陣に帰ってくるアサーションまでについて示すだけでよい

・停止性の証明

非負性：関数値が常に非負

減少性：その位置に実行が到達するたびに関数値が減少する

この 2 つを数学的帰納法などを用いて示す。

・XOR を AND,OR,NOT のみで示す

NAND のみで AND,OR,NOT をあらわすような問題が多かったが、これらは簡単で、NOT は与えられている場合が多いので割愛。

ただしXORは知らないと無理だと思うのでここに載せておく。

a ^ b = ~(a & b)&(a | b)

MIL記法を用いてNANDのみで記したXOR回路は以下のようになる。