計算機システムの基礎(第10回　配布)

ソフトウエア

・オペレーティングシステム
・アルゴリズム

予告：　期末試験は7/24に実施予定。
　　中間演習　と同様な方法を検討中

　　8/1はプログラミング演習　ですが、
　　自力でできる人はOndemadでOK

担当：　福井大学　大学院工学研究科
　　情報・メディア工学専攻
　　森　眞一郎　(moris@u-fukui.ac.jp)

スライドは、一部のページを除き下記URLでカラー版が入手可能です
http://sylph.fuis.u-fukui.ac.jp/~moris/lecture/CompIntro/

Web　教材を使った学習

- JREC-IN　研究人材のための e-Learning教材

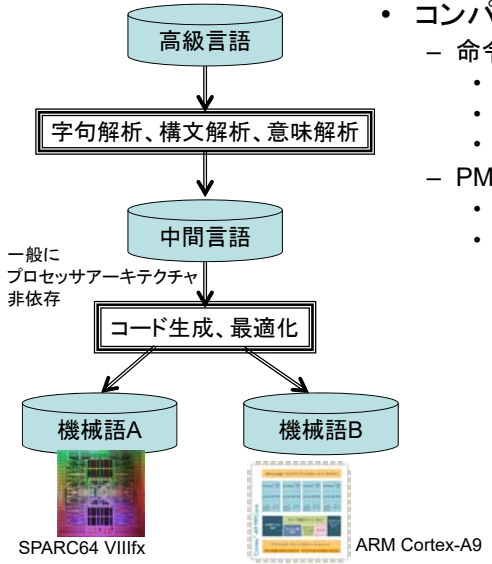
https://hakase-compass.jst.go.jp/e-learning

「情報通信」

・オペレーティングシステムコース　(2022.7.1復活確認)
1. オペレーティングシステムの役割(7分)
2. OSの関係する各種の接点(7分)
3. プログラムとのインタフェース(10分)
4. OSの構成(12分)

・データ構造とアルゴリズムコース　(2020.07.10 時点で復活確認)
1. アルゴリズムと計算時間(10分)
2. 配列の探索[Search](10分)
7. 整列[Sequential Sort](10分)
8. 整列[Heap Sort and Quick Sort](10)
11. 各種のアルゴリズム(10分)

コンパイラとアーキテクチャの関係



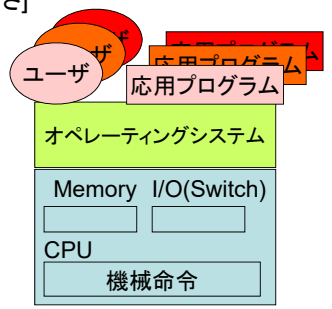
- コンパイラが意識するのは
  - 命令セットアーキテクチャ
    - 命令セットのシンプルさ
    - 命令セットの機能レベル
    - レジスタの数
  - PMSアーキテクチャ
    - キャッシュの構造、容量
    - キャッシュ(一貫性)制御のポリシ

例　A ＝ B × 6
ISA1(掛算命令あり)
　　MUL　A、B、6

ISA2(掛算命令なし)
　　LSHIFT　B2、B、1
　　LSHIFT　B4、B2、1
　　ADD　　A、B4、B2

OSとアーキテクチャの関係

- OS(オペレーティングシステム)の代表的な機能
  - ハードウェア資源の管理
    - 資源保護[信頼性]
    - 抽象化による操作性の向上[使いやすさ]
  - 実行制御
    - マルチプログラミング[性能]
  - ユーザインタフェース[使いやすさ]



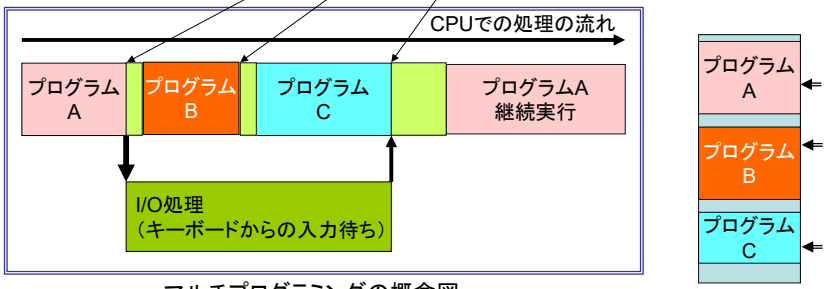
例　OSが提供する　scanf, printf　の機能
　　キーボードからのデータ読み込み
　　TTYコンソール(標準出力)への文字出力

OSとアーキテクチャの関係

- OSの代表的な機能
  - ハードウェア資源の管理
  - ユーザインタフェース
  - 実行制御
    - マルチプログラミング

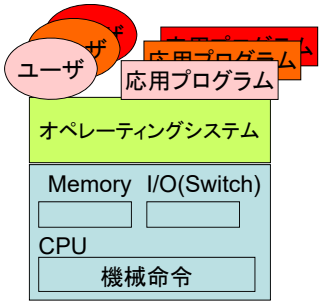
コンテキスト（文脈）切り替え、割り込み処理

コンテキストとは
プログラムカウンタ、レジスタ等
個々のプログラムの実行に必要不可欠な実行時情報



マルチプログラミングの概念図

OSとアーキテクチャの関係



- OSが意識するのは
  - 命令セットアーキテクチャ
    - 割り込みアーキテクチャ
      - 割り込みレベル
      - レジスタ構成
        - » コンテクスト関係
    - 資源保護メカニズム
      - 特権命令/実行モード
        - » スーパバイザモード vs ユーザモード
        - » 排他制御命令
      - 仮想化機構
        - » メモリ管理支援機構、アドレス変換支援機構
  - PMSアーキテクチャ
    - 主記憶の容量
    - 入出力装置の構成、制御方法

Web 教材を使った学習

- JREC-IN 研究人材のための e-Learning教材

https://hakase-compass.jst.go.jp/e-learning

「情報通信」

・オペレーティングシステムコース (2022.7.1復活確認)

1. オペレーティングシステムの役割（7分）
2. OSの関係する各種の接点（7分）
3. プログラムとのインタフェース（10分）
4. OSの構成（12分）

・データ構造とアルゴリズムコース (2020.07.10 時点で復活確認)

1. アルゴリズムと計算時間(10分)
2. 配列の探索[Search](10分)
7. 整列[Sequential Sort](10分)
8. 整列[Heap Sort and Quick Sort](10)
11. 各種のアルゴリズム(10分)

計算量の話

表 1.2.3 計算量と計算時間の関係（特記するものを除いて単位は秒）

計算量 \ n	10	20	50	100	200	500	1000	10000
$O(1)$	1	1	1	1	1	1	1	1
$O(\log n)$	0.5	0.65	0.85	1	1.15	1.35	1.5	2
$O(n)$	0.1	0.2	0.5	1	2	5	10	100
$O(n \log n)$	0.05	0.13	0.42	1	2.3	6.75	15	200
$O(n^2)$	0.01	0.04	0.25	1	4	25	100	2.78 時間
$O(n^3)$	0.001	0.008	0.125	1	8	125	1000	11.6 日
$O(2^n)$	0.001	1.02	34.8 年	—	—	—	—	—

O(n!) 0.001 21年

アルゴリズム(Algorithm, 算法)

問題を解くための手順を定めたもの。
この手順は、どのような操作をどのような順序で行うかを曖昧な点の残らないようにきちんと定めたもの。

アルゴリズムの条件
1. どんな入力データに対しても、間違った答えを与えないこと。
2. どのような入力データに対しても有限の時間で答えを与えること
（注）問題の条件に反する入力データに対しては、上の2つを保障する必要はない。

アルゴリズムの評価尺度
1. 求解に要する時間（時間計算量）
2. 求解に要するメモリ領域（空間計算量）

計算量：
入力データのサイズがnであったときに計算の手間がどのくらいになるかを nの関数として表したもの

オーダ記法 O(n) ：
ある正定数 c, n_0が存在し、$n>n_0$を満たす全てのnに対して、
$T(n) \leqq cf(n)$となるとき$T(n)= O(f(n))$ と表記する。
（例）$T(n)=0.1n^3+1000n^2 =O(n^3)$

代表的なアルゴリズム設計戦略

1. 分割統治法(Divide and Conquer Algorithm)
例：ソーティング、離散フーリエ変換

2. 動的計画法(Dynamic Programming Algorithm: DP)
例：最短経路問題（全対全）、多角形の三角形分割、遺伝子のアラインメント

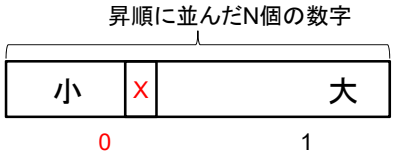
3. 貪欲法, 欲張り法(Greedy Algorithm)
例：最小木、最短経路問題(1対1)

4. バックトラック法(Backtracking Algorithm),
分岐限定法(Branch and Bound Algorithm)
例：ナップサック問題、巡回セールスマン問題

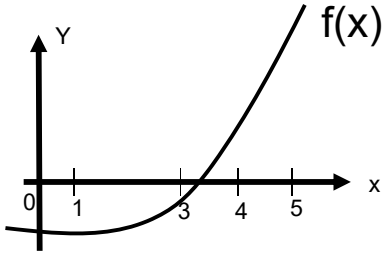
0と1を抽象的に考える （右 or 左）

2分探索

列中の数字を『探す』



f(x)=0となるxを『探す』

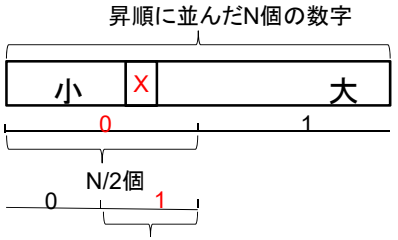


001 --- 101(4)
011 --- 101(2)
011 --- 100(1)
11.0 --- 11.1(0.5)
11.01---11.10(0.25)

0と1を抽象的に考える （右 or 左）

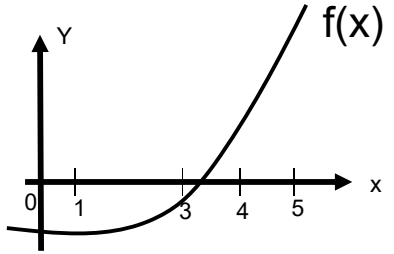
2分探索

列中の数字を『探す』



$log_2 N$ 回で必ず答えが出る

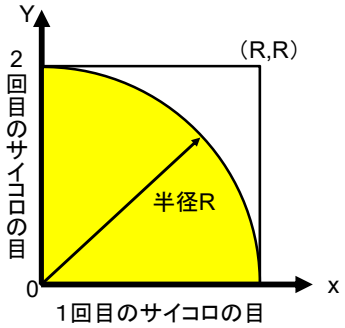
f(x)=0となるxを『探す』

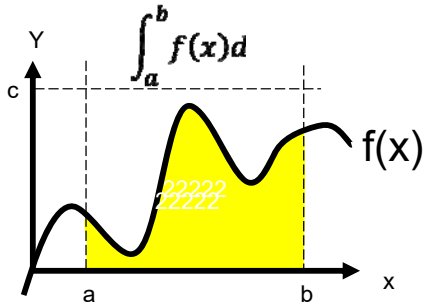


001 --- 101(4)
011 --- 101(2)
011 --- 100(1)
11.0 --- 11.1(0.5)
11.01---11.10(0.25)

0と1を抽象的に考える（中　と　外）

モンテカルロ法





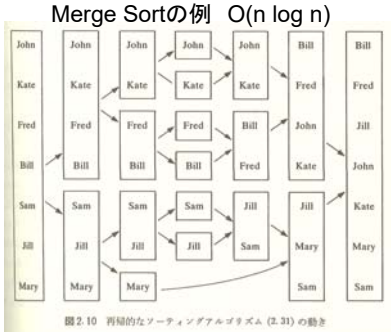
0から始まるサイコロを考えてください。
皆さんが知っている6面体であれば
Rは5になります。

[考えてみよう！]
f(x)の最小値が　負の数だったらどうしよう？

分割統治法(Divide and Conquer Algorithm)

与えられた問題を、まずいくつかの小さな問題に分割して、次にその分割された問題の解を用いて最初に与えられた問題の解を求める問題。

Merge Sortの例　O(n log n)



順序回路の等価性判定の例
＝FFマッチング＋組合せ回路の等価性判定

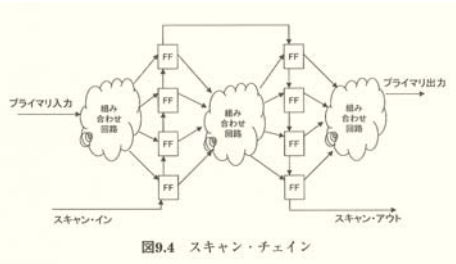


図9.4　スキャン・チェイン

動的計画法
(Dynamic Programming Algorithm: DP)

与えられた問題を、まずいくつかの小さな部分問題に分割して、
次にその部分問題をある種の適当な順番で解いて、その解を表にして表し、最後に、その表を引く(search)ことで最初の問題を解く方法

大きな問題を解く過程で、同じ小問題が何度も出てくる場合に、小問題をといた結果を保存しておいて、再利用することで計算量を削減する方法

[例]　Fibonacci数列

$F_n=F_{n-1}+F_{n-2}$ (n≧2)
ただし
F1=1
F0=0

再帰で計算すると
T(n)=T(n-1)+T(n-2) +1(n≧2)
T(n)=0 (n=0,1)
⇒　(0.5*(1+√5))^(n-2)　….　指数時間

表を作ると
T(n) = n　　O(n)=n

動的計画法
(Dynamic Programming Algorithm: DP)

例題 8.1
文字列 X = abcbdab, Y = bdcaba の最長共通部分列を動的計画法で求める過程を示せ (表を示せ).

【解答】

表 8.1

		Y	–	b	d	c	a	b	a
	–		0	0	0	0	0	0	0
	a		0	0	0	0	1	1	1
	b		0	1	1	1	1	2	2
	c		0	1	1	2	2	2	2
X	b		0	1	1	2	2	3	3
	d		0	1	1	2	2	3	3
	a		0	1	1	2	2	3	4
	b		0	1	1	2	2	3	(4)

```
Longest Common Substring (X, Y) {
  LCS ← X.length × Y.length の大きさの表
  for (i ← 0...X.length)
    for (j ← 0...Y.length)
      d ← もし X[i] = Y[j] ならば 1, それ以外なら 0
      LCS[i, j] ← max( LCS[i − 1, j − 1] + d,
                       LCS[i − 1, j],
                       LCS[i, j − 1]);
}
```

右下の 4 が最長共通部分列の長さである．そこへ到る青い矢印を逆にたどると (表の灰色部分) 最長共通部分列 bcba が得られる．■

参考：
ヒトゲノム文字列　~30億文字
遺伝子配列　2～2.5万個

「赤の印紙を売るか？」
に「青い空」が含まれる？

五十嵐健夫著、「データ構造とアルゴリズム」，数理工学社 より引用

貪欲法(Greedy Algorithm)

その場、その場において、最も最適なものを、将来そのために
不利益になるかもしれないなどとは少しも考えずに選んでいって
解をもとめようという手法。
（一般に、得られた解が最適解であるとは限らない。）

[例] 1次元詰め込み問題: （2次元に拡張すると配置問題）

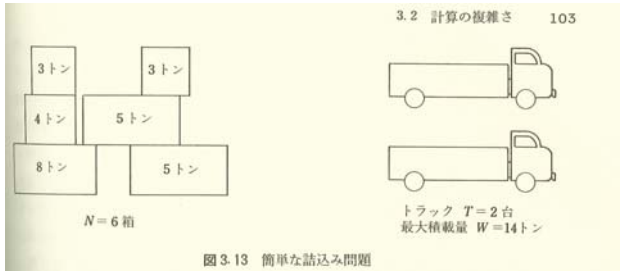


図3.13 簡単な詰込み問題

Greedy
8+5
5+4+3
あまり3

最適解
8+3+3
5+5+4

L.ゴールドシュレーガー、A.リスター著、「計算機科学入門」(第2版), 近代科学社より引用

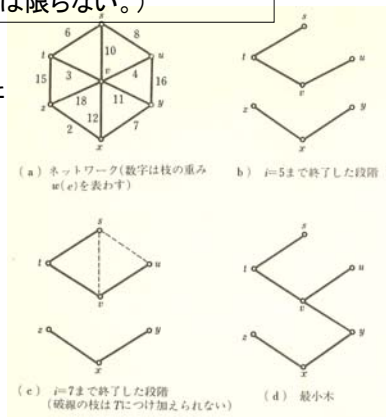
貪欲法(Greedy Algorithm)

その場、その場において、最も最適なものを、将来そのために
不利益になるかもしれないなどとは少しも考えずに選んでいって
解をもとめようという手法。
（一般に、得られた解が最適解であるとは限らない。）

[例] 最小木問題: （LSI内の総配線長最小化）
辺に重みが定義されている無向連結グラフにおいて
グラフの連結性を保ったまま辺をとりのぞいて、残った
辺の重みの和が最小となるものを見つける問題

G=(V,E), n=|V(G)|, m=|E(G)|
[Kruskalのアルゴリズム]
1)辺の重みの小さい順にソートして
c(e1)≦c(e2)≦…≦c(em)とする
2)T:= (V(G),0) とする
3) For i=1 to m do
If T+ei が閉路を含まない then T:=T+ei とする

[参考]B.コルテ、J.フィーゲン著、「組合せ最適化 --理論とアルゴリズム--」, シュプリンガー・フェアラーク東京



[引用]
浅野、今井著、計算とアルゴリズム、オーム社

(注)最小木問題 と 最短経路問題は違います！！