データ構造とアルゴリズム　第１回

- 担当：増澤 利光　masuzawa@ist.osaka-u.ac.jp
  泉 泰介　t-izumi@ist.osaka-u.ac.jp
- ＴＡ：谷内 優斗（M1）y-taniuchi@ist.osaka-u.ac.jp
  - ミニレポート採点，質問対応
- 成績：試験（中間／期末）７割，ミニレポート ３割
- オンライン教材：CLEの「コンテンツ／オンライン教材」
- 講義資料等の配布：CLEの「コンテンツ／講義資料」
- ミニレポート：CLEの「コンテンツ／ミニレポート」
  - ミニレポート課題提示／レポート提出，質問
  - 締切は次回の講義の前日

講義内容

- 内容：データ構造とアルゴリズムの基礎
- テキスト
  - 浅野，和田，増澤「アルゴリズム論 」オーム社(2003)
    - ８章まで（９章は「情報解析Ｂ」で学習）
- 効率よく問題を解く（計算する）手法を学ぶ
  - データ構造：計算機内でのデータの表現方法
  - アルゴリズム：データの処理方法（手順）
  - プログラミングA/Bと重複あり（二分探索，ソートなど）
- 何の役に立つのか
  - ソフトウェア開発
  - 問題の難しさの理解：この時間で解ける／解けない

データ構造とアルゴリズム　内容

1. アルゴリズムの重要性
2. 探索問題
3. 基本的なデータ構造
4. 動的探索問題とデータ構造
5. データの整列

＊ 中間テスト

6. グラフのアルゴリズム
7. 文字列のアルゴリズム
8. アルゴリズム設計手法

＊ 期末テスト

データ構造とアルゴリズム　第１回

1. アルゴリズムの重要性
2. 探索問題
3. 基本的なデータ構造
4. 動的探索問題とデータ構造
5. データの整列
6. グラフのアルゴリズム
7. 文字列のアルゴリズム
8. アルゴリズム設計手法

第１章　アルゴリズムの重要性

1.1　アルゴリズムとは

1.2　アルゴリズムの記述

1.3　アルゴリズムの効率

1.4　アルゴリズムの最適性

今日の学習目標

- アルゴリズムとは何か説明できる
- アルゴリズムの重要性を説明できる
- アルゴリズムの記述を理解でき，アルゴリズムを記述できる
- 漸近的計算量とその重要性を説明できる
  - O記法，Ω記法
- アルゴリズムの最適性とは何か説明できる

1.1　アルゴリズムとは

- ソフトウェアができるまで
  1. 問題発見
  2. 問題分析
  3. どうやって解くか ⇒ アルゴリズム設計
  4. プログラム作成（プログラミング）
  5. ソフトウェア（プログラム）完成

1.1　アルゴリズムとは

- アルゴリズム（algorithm，算法）
  - 計算機で問題を解くための手順
  - 曖昧性があってはいけない
  - 正しくなければならない
  - プログラム ＝ アルゴリズムの計算機向け表現

アルゴリズムの計算量

- 優れたアルゴリズムとは
  - 理解しやすい
    - プログラム化しやすい
  - 計算時間が短い　　→ **時間計算量**
  - 使用メモリが小さい → **領域計算量**（空間計算量）
  - 　・・・

アルゴリズムの計算量

- 計算量：問題のサイズの関数
  - 同じサイズの問題でも，入力によって異なる
    - 例：ソーティング（データを大きさで並び替える）
    - 挿入法（アルゴリズム）：
      入力データがソート済みに近いほど高速
- **最大**時間計算量：**最悪**の場合の時間計算量
  - 実時間処理では重要（処理の遅れが許されない）
  - 評価は比較的容易
- **平均**時間計算量：**平均**の場合の時間計算量
  - バッチ処理では重要
  - 評価は一般に難しい
  - 確率分布の仮定の妥当性？

第１章　アルゴリズムの重要性

1.1　アルゴリズムとは

1.2　アルゴリズムの記述

1.3　アルゴリズムの効率

1.4　アルゴリズムの最適性

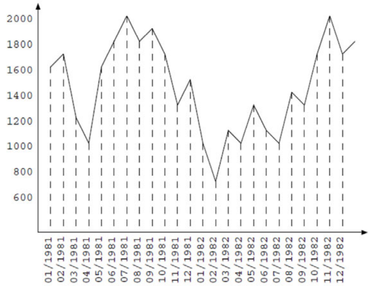
１.２　アルゴリズムの記述

- テキストではＣ風の書き方
  - 分かりやすさのために，マクロ的な書き方もする
    - 例：株式投資における最大売却益

```
 入力：毎月の株価sp[0], ..., sp[n-1]
mxp = 0; // 利益の最大値 mxp を 0 に初期化
for i=0 to n-2
    for j=i+1 to n-1 {
        d = sp[j]-sp[i]; //売却益 = 売値 - 買値
        if d > mxp then mxp = d;
         // 今まで以上の売却益であればmxpの値を更新する
    }
mxpを答として返す;
```

最大売却益問題

- 問題：最大売却益問題
- 入力：$sp[0], sp[1], \ldots, sp[n-1]$（$SP[i]$：年月 i の株価）
- 出力：**売却益が最大**となる買う年月と売る年月を求める
  - **売却益**： $sp[j] - sp[i]$
    年月 i に買って，年月 j に売る場合（$i < j$）



最大売却益アルゴリズム1.1

- アルゴリズム1.1：(A) 方式

```
mxp = 0; // 利益の最大値 mxp を 0 に初期化
for (i=0 to n-2)
    for (j=i+1 to n-1) {
        d = sp[j] -sp[i]; //売却益 = 売値 - 買値
        if (d > mxp) mxp = d;
         // 今まで以上の売却益であればmxpの値を更新する
    }
mxpを答として返す;
```

全 $i, j\ (0 \le i < j \le n-1)$ の組をチェック

計算時間：減算　$n(n-1)/2$ 回

比較　$n(n-1)/2$ 回

最大売却益アルゴリズム1.2

- アルゴリズム1.2：(A) 方式

```
mxp = 0; // 利益の最大値 mxp を 0 に初期化
for (i=0 to n-2) {
    mxsp = sp[i]; //株価の最高値 mxsp を sp[i] に初期化
    for (j=i+1 to n-1)
        if (sp[j] > mxsp) mxsp = sp[j];
    d = mxsp - sp[i]; //売却益=買値とそれ以降の最高値との差
    if (d > mxp) mxp = d;
    // 今まで以上の売却益であればmxpの値を更新する
    }
mxpを答として返す;
```

年月 i 以後の最高値を求める

計算時間：減算　$n-1$ 回（$n(n-1)/2$ から改善）

比較　$n(n-1)/2 + n - 1$ 回

最大売却益アルゴリズム1.3

- アルゴリズム1.3：(B) 方式

```
mxp = 0; // 利益の最大値 mxp を 0 に初期化
for (j=1 to n-1) {
    mnsp = sp[j]; // 株価の最安値 mnsp を sp[j] に初期化
    for (i=0 to j-1)
        if (sp[i] < mnsp) mnsp = sp[i];
    d = sp[j] - mnsp; //売却益=売値とそれ以前の最安値との差
    if (d > mxp) mxp = d;
    // 今まで以上の売却益であればmxpの値を更新する
}
mxpを答として返す;
```

年月 j 以前の最安値を求める

計算時間：減算　$n-1$ 回

比較　$n(n-1)/2 + n - 1$ 回

第１章　アルゴリズムの重要性

1.1　アルゴリズムとは

1.2　アルゴリズムの記述

1.3　アルゴリズムの効率

1.4　アルゴリズムの最適性

1.3　アルゴリズムの効率

- 最大売却益アルゴリズム
- アルゴリズム１.１
  - 計算時間：減算　$n(n-1)/2$ 回
  - 比較　$n(n-1)/2$ 回
- アルゴリズム１.２，１.３
  - 計算時間：減算　$n-1$ 回
  - 比較　$n(n-1)/2 + n-1$ 回
- どれも，概ね n^2 に比例する計算時間
  - $O(n^2)$ と表す
    - オーダ n^2 または ビッグオー n^2 と読む

漸近的計算量

- O記法（オーダ記法）
  - 漸近的計算量の表し方
  - 主要項以外の項を無視
  - 主要項の係数を無視

例：$\boldsymbol{an} + b \quad (a > 0) \quad \rightarrow \boldsymbol{O(n)}$

$\boldsymbol{cn^2} + dn + e \quad (c > 0) \quad \rightarrow \boldsymbol{O(n^2)}$

主要項

O記法（オーダ記法）　p.29

$O(f(n))$　　オーダ $f(n)$，ビッグオー $f(n)$

ある正定数 n_0, c が存在し，

$n \geq n_0$ を満たすすべての n について，

$g(n) \leq c \cdot f(n)$ を満たす関数 $g(n)$ の集合

スモールオー $o(f(n))$ もある

- $g(n) \in O(f(n))$ を $g(n) = O(f(n))$ と書くことも多い
- n が十分大きい場合の $g(n)$ の上界
- $an + b$ を表すのに正しいのはどれ？

$O(\log n)$✖　$O(\sqrt{n})$✖　$O(n)$

$O(5n)$　$O(n^2)$　$O(n+\sqrt{n})$

なるべく小さく簡単な関数で表す　$\boldsymbol{O(n)}$

O記法に関する演算

- $f_1(n) = O(g_1(n)),\quad f_2(n) = O(g_2(n))$
- $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$

$$f_1(n) \le c_1 \cdot g_1(n) \qquad (n \ge n_1)$$
$$f_2(n) \le c_2 \cdot g_2(n) \qquad (n \ge n_2)$$
$$\therefore f_1(n) + f_2(n)$$
$$\le c_1 \cdot g_1(n) + c_2 \cdot g_2(n) \qquad (n \ge \max(n_1, n_2))$$
$$\le c(g_1(n) + g_2(n)) \qquad (c = \max(c_1, c_2))$$

- $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

Ω記法（オメガ記法）　p.29

$\Omega(f(n))$　オメガ $f(n)$，ビッグオメガ $f(n)$

ある正定数 n_0, c が存在し，

$n \ge n_0$ を満たすすべての n について，

$g(n) \ge c \cdot f(n)$ を満たす関数 $g(n)$ の集合

スモールオメガ $\omega(f(n))$ もある

- $g(n) \in \Omega(f(n))$ を $g(n) = \Omega(f(n))$ と書くことも多い
- n が十分大きい場合の $g(n)$ の下界（かかい）
- $an + b$ を表すのに正しいのはどれ？

$\Omega(\log n)$　$\Omega(\sqrt{n})$　$\Omega(n)$

$\Omega(5n)$　$\Omega(n^2)$ ✖　$\Omega(n + \sqrt{n})$

なるべく大きな簡単な関数で表す　$\Omega(n)$

Θ記法（シータ記法）　p.29

$f(n) \in \Theta(g(n))$　シータ $g(n)$

$f(n) \in O(g(n))$ かつ $f(n) \in \Omega(g(n))$

スモールシータはない

- $g(n) \in \Theta(f(n))$ を $g(n) = \Theta(f(n))$ と書くことも多い
- n が十分大きい場合の $g(n)$ の上下界
- $an + b$ を表すのに正しいのはどれ？

$\Theta(\log n)$ ✖　$\Theta(\sqrt{n})$ ✖　$\Theta(n)$

$\Theta(5n)$　$\Theta(n^2)$ ✖　$\Theta(n + \sqrt{n})$

なるべく簡単な関数で表す　$\Theta(n)$

他の漸近的計算量の記法（参考）

$o(f(n))$　スモールオー $f(n)$

任意の正定数 c に対し，ある正定数 n_0 が存在し，

$n \ge n_0$ を満たすすべての n について，

$g(n) < c \cdot f(n)$ を満たす関数 $g(n)$ の集合

- O記法との違いに注意
  - 真に大きい上界
- $an + b$ を表すのに正しいのはどれ？

$o(\log n)$ ✖　$o(\sqrt{n})$ ✖　$o(n)$ ✖

$o(5n)$ ✖　$o(n^2)$　$o(n + \sqrt{n})$ ✖

他の漸近的計算量の記法（参考）

$\omega(f(n))$　　スモールオメガ $f(n)$

任意の正定数 c に対し，ある正定数 n_0 が存在し，

$n \geq n_0$ を満たすすべての n について，

$g(n) > c \cdot f(n)$ を満たす**関数 $g(n)$ の集合**

- Ω記法との違いに注意
  - **真に小さい**下界
- $an + b$ を表すのに正しいのはどれ？

$\omega(\log n)$　　$\omega(\sqrt{n})$　　$\omega(n)$ ✖

$\omega(5n)$ ✖　　$\omega(n^2)$ ✖　　$\omega(n + \sqrt{n})$ ✖

最大売却益アルゴリズム１.４

- アルゴリズム１.４：（Ｂ）方式　改良版

```
mxp = 0;// 利益の最大値 mxp を 0 に初期化
msf = sp[0]; // これまでの最安値 msf を sp[0] に初期化
for (j=1 to n-1) {
    d = sp[j] - msf; // 売却益
    if (d > mxp) mxp = d;
    // 今まで以上の売却益であればmxpの値を更新する
    if (sp[j] < msf) msf = sp[j];
    // これまでの最安値の更新
}
mxpを答として返す;
```

計算時間：減算　$n - 1$ 回

　　　　　比較　$2(n - 1)$ 回

$O(n)$

漸近的計算量の重要性

- アルゴリズムの**オーダーの改善** ⇒ **劇的な高速化**
  - 例：$\Theta(n^2)$ から $\Theta(n)$ に改善
    - 係数が同じなら，

      $n = 100$ で 100 倍高速化

      $n = 1000$ で 1000 倍高速化
    - 計算機が 100 倍高速化したとき，

      同じ時間で解ける問題のサイズ

      $\Theta(n^2)$ なら 10 倍

      $\Theta(n)$ なら 100 倍

第１章　アルゴリズムの重要性

1.1　アルゴリズムとは

1.2　アルゴリズムの記述

1.3　アルゴリズムの効率

1.4　アルゴリズムの最適性

最大売却益アルゴリズム１.４

- アルゴリズム１.４：（Ｂ）方式　改良版
  - 計算時間　$O(n)$
- より高速なアルゴリズムを考案できるか？　NO!
  - 株価を１つでも見落とすと正解を得られない可能性
  - n 個の株価をすべて見ないといけない
  - 絶対に n に比例した時間はかかる
- 最大売却益問題の計算時間の下界は $\Omega(n)$
- 最大売却益アルゴリズム１.４の計算時間は最適

第１章　アルゴリズムの重要性

1.1　アルゴリズムとは
1.2　アルゴリズムの記述
1.3　アルゴリズムの効率
1.4　アルゴリズムの最適性

今日の学習目標（振返り）

- アルゴリズムとは何か説明できる
- アルゴリズムの重要性を説明できる
- アルゴリズムの記述を理解でき，アルゴリズムを記述できる
- 漸近的計算量とその重要性を説明できる
  - O記法，Ω記法
- アルゴリズムの最適性とは何か説明できる