アルゴリズムとデータ構造

- 第8回講義トピック：整列 (ソート、sorting)
  - 選択ソート
  - 挿入ソート
  - バブルソート
  - シェルソート

整列 (sorting) とは

- 整列、ソート:

  データ列をある規則(キーの順序)にしたがって並べ替えること

  - 内部整列:コンピュータの主記憶に読み込まれたデータに対して整列を行う。
  - 外部整列:外部ディスクからデータを読みながら、整列を行う。(この部分は本講義の対象外)

整列方法の性能

- 性能をはかるパラメータ:
  - 実行時間コスト
    - 比較の回数、交換の回数など基本操作を考える。
    - O記法で記述する。
  - 作業領域コスト
    - メモリ使用量を考える。
  - 安定性
    - ソーティングによって要素の位置と順序関係が変化するが、同じキーを持つ項目の相対的な順序が保たれる場合、安定性があるという。

選択ソート (selection sort)

- 考え方
  - まず、全要素から最小の要素を見つけ、最初の位置にある要素と交換する。
  - 次に残りの要素から、2番目に小さい要素を見つけ、2番目の位置にある要素と交換する。
  - 同じ操作を残り要素数1つになるまで繰り返す。

選択ソートのプログラム

```
selection(int a[], int N) {
  int i, j, min, t;
  for (i = 1; i < N; i++) {
    min = i;
    for (j = i+1; j <= N; j++)
      if (a[j] < a[min]) min = j;
    t = a[min]; a[min] = a[i]; a[i] = t;
  }
}
```

選択ソートの例

- 以下の文字列を選択ソートによって整列される様子

  Ａ Ｓ Ｏ Ｒ Ｔ Ｉ Ｎ Ｇ Ｅ Ｘ Ａ Ｍ Ｐ Ｌ Ｅ

挿入ソート (insertion sort)

- 考え方
  - データの前半部分がすでに整列されているとして、次の要素を取り上げ、整列されているデータ列に挿入する。（データ挿入により、それ以降のデータは1つずつシフトする必要があるので、要素交換の回数が増える。）
  - 同様の操作を全要素が整列するまで繰り返す。

挿入ソートのプログラム

- このアルゴリズムはvが配列の最小値の時whileで配列の左端に出てしまうので、a[0]に番兵（sentinel，番人、標識）、この場合はつまり配列の最小値よりも小さい値を置く必要がある。
- もし、配列の最小値が決められない場合はwhile文の中でindexのチェックをする必要があり、アルゴリズムが若干遅くなる。

```
insertion(int a[], int N) {
  int i, j, v;
  for (i = 2; i <= N; i++) {
    v = a[i]; j = i;
    while (a[j-1] > v) {
      a[j] = a[j-1]; j--;
    }
    a[j] = v;
  }
}
```

挿入ソートの例

- 以下の文字列を挿入ソートによって整列される様子

  Ａ Ｓ Ｏ Ｒ Ｔ Ｉ Ｎ Ｇ Ｅ Ｘ Ａ Ｍ Ｐ Ｌ Ｅ

バブルソート (bubble sort)

- 考え方
  - データの先頭から末尾まで、順番に隣り合う要素を比較し、順序が逆ならば、要素を交換する。
  - 一巡の比較交換で一番大きい要素が必ず一番右に来るので、次の巡回操作は最後から1つ手前で止めることができる。
  - 以上の操作を配列全体が整列されるまで繰り返す。

バブルソートのプログラム

```
bubble(int a[], int N) {
  int i, j, t;
  for (i = N; i >= 1; i--)
    for (j = 2; j <= i; j++)
      if (a[j-1] > a[j]){
        t = a[j-1];
        a[j-1] = a[j];
        a[j] = t;
      }
}
```

バブルソートの例

- 以下の文字列がバブルソートによって整列される様子

  ＡＳＯＲＴＩＮＧＥＸＡＭＰＬＥ

シェルソート (shell sort)

- 挿入ソートの問題点
  - 隣の要素としか比較しないので、整列されるまで時間が掛かり、交換の回数も増える。
- シェルソートの考え方
  - まず、一定間隔hで並んでいる要素の部分列（hだけの数の部分列がある）について整列を行う。hだけ離れた要素と交換するので、交換のスピードがあげる。この整列をh－整列という。
  - hの値を徐々に小さくしていき、最後にh=1の通常の挿入ソートを行えば、配列全体が整列される。
    この最後の通常挿入ソートはデータがすでに整列に近い状態になっているので、最初から挿入ソートを行うことより遥かに速くできる。

シェルソートのプログラム

```
shellsort(int a[], int N) {
  int i, j, h, v;
  for (h = 1; h <= N/9; h = 3*h+1) ;
  for ( ; h > 0; h /= 3)
    for (i = h+1; i <= N; i += 1) {
      v = a[i]; j = i;
      while (j > h && a[j-h] > v) {
        a[j] = a[j-h]; j -= h;
      }
      a[j] = v;
    }
}
```

＊hの取り方は決まってないが、例えば3*h+1で計算される数列：1，4，13，40，121，364，1093，…の最大値（N/9を超えた値）から1へ逆にたどっていく方法

シェルソートの例

- 以下の文字列がシェルソートによって整列される様子

  Ａ Ｓ Ｏ Ｒ Ｔ Ｉ Ｎ Ｇ Ｅ Ｘ Ａ Ｍ Ｐ Ｌ Ｅ