プロコン練習会：ソート（整列）

ソート（整列）とは？

- データの持つ値を昇順，もしくは降順に並び替えること

- 例：$\mathrm{A} = \{3, 6, 9, 2, 1, 5\} \rightarrow \{1, 2, 3, 5, 6, 9\}$

ソート（整列）とは？

- データの持つ値を昇順，もしくは降順に並び替えること

- 例：$A = \{3, 6, 9, 2, 1, 5\} \rightarrow \{1, 2, 3, 5, 6, 9\}$

- このような前処理を行うことで1回の何かしらの操作を高速で行えるようになる．（値の検索など）

挿入ソート

1	2	3	6	5	4

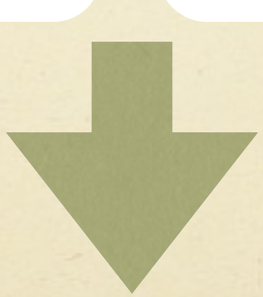
挿入ソート

1	2	3	6	5	4

1	2	3	5	6	4

挿入ソート

1	2	3	5	6	4



1	2	3	4	5	6

挿入ソート

- どんなアルゴリズムで並び替えているのか？

挿入ソート

- どんなアルゴリズムで並び替えているのか？

・途中まではうまくソートされていた.

・ソートされていない値が現れたらその値を適切な位置に挿入していく.

基本的にやることはこれだけ！！

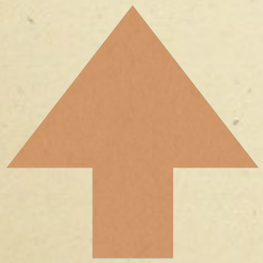
挿入ソート

1	2	3	6	5	4

ソート済み

挿入ソート

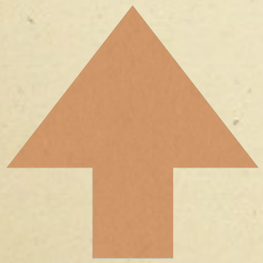
1	2	3	6	5	4



ソート済み

挿入ソート

1	2	3	6	5	4



ソートされていない

挿入ソート

1	2	3	6	5	4

5が入って欲しい位置

ソートされていない

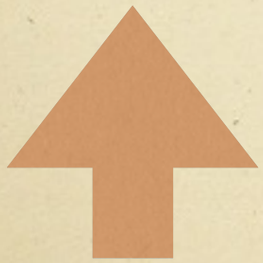
挿入ソート

挿入！！

ソートされていない

挿入ソート

1	2	3	5	6	4



ソート済み

計算量

- 時間計算量：$O(n^2)$

  n：カードの枚数

- カード一枚を並び替えるのに最悪でn回の比較を行うので$n*n = n^2$になる.

コード

```
1  #include<iostream>
2  using namespace std;
3
4  void insertionSort(int* A, int N){//N個の要素を含む配列A
5    for (int i = 1; i < N; i++) {
6      int v = A[i];
7      int j = i - 1;
8      while(j >= 0 && A[j] > v){
9        A[j + 1] = A[j];j--;
10       A[j + 1] = v;
11     }
12   }
13 }
```

バブルソート

3	4	1	6	5	2

バブルソート

3	4	1	6	5	2

比較

バブルソート

3	4	1	6	2	5

交換

バブルソート

3	4	1	6	2	5

比較

バブルソート

3	4	1	2	6	5

交換

バブルソート

3	4	1	2	6	5

比較

バブルソート

3	4	1	2	6	5

比較

バブルソート

3	1	4	2	6	5

交換

バブルソート

3	1	4	2	6	5

比較

バブルソート

1	3	4	2	6	5

交換

バブルソート

1	3	4	2	6	5

この部分に対して同じ操作をする

バブルソート

- 1回の操作で赤枠の中の最小の値が一番左に来る.
- よって，n回の操作でソートが完了する
- 時間計算量：$O(n^2)$

コード

```
1  #include<iostream>
2  using namespace std;
3
4  void bubleSort(int* A, int N){
5    bool flag = 1;
6    while(flag){
7      flag = 0;
8      for (int i = 0; i < N; i++) {
9        for (int j = N - 1; j >= i; j--) {
10         if(A[j] < A[j - 1]){
11           flag = 1;
12           int tmp = A[j];
13           A[j] = A[j - 1];
14           A[j - 1] = tmp;
15         }
16       }
17     }
18   }
19 }
```

選択ソート

1	3	4	2	6	5

選択ソート

1	3	4	2	6	5

最小値

選択ソート

1	3	4	2	6	5

移動

選択ソート

1	3	4	2	6	5

最小値

選択ソート

1	2	4	3	6	5

移動

選択ソート

- まだソートされていない範囲から最小の値を選んで一番左に持って来る.
- 時間計算量：$O(n^2)$
- 最小の値を選ぶのに$O(n)$，ソートされるまでにこれをn回行うので$O(n^2)$となる.

コード

```
1  #include<iostream>
2  using namespace std;
3
4  void selectionSort(int* A, int N){
5    for (int i = 0; i < N; i++) {
6      int minj = A[i];
7      int pos = i;
8      for (int j = i; j < N; j++) {
9        if(minj > A[j]){
0          minj = A[j];
1          pos = j;
2        }
3      }
4      int tmp = A[i];
5      A[i] = minj;
6      A[pos] = tmp;
7    }
8  }
```

安定ソート

- 安定ソートとはソートのアルゴリズムの中で同等のデータの順序がソート前とソート後で変わらないソートのことである.

- これまで紹介したソートは全て安定なソートである.

プロコンで使うソート

- 今回紹介はしていないがプロコンでよく使うソートはクイックソートと呼ばれるO(n*log n)のソートアルゴリズムである.

- 気になる人は「クイックソート　アルゴリズム」でググると色々出てくると思います.

使い方

- C++

```cpp
#include<iostream>
#include<algorithm>
using namespace std;

int main(int argc, char *argv[]){
  int a[n];
  sort(a, a + n);
  return 0;
}
```

他のソート

- 基数ソート：O(n*log n)
- シェルソート：$O(n^{1.25})$
- マージソート：O(n*log n)
- バケットソート：O(n)