基本情報技術 I

10. 流れ図, ソート

菊池浩明

講義目標

- 教科書
  - 2-2 アルゴリズムの記述
    - 2. 流れ図, 3. 流れ図の理解
  - 2-3 基本アルゴリズム
    - 1. 整列処理

アルゴリズムとは

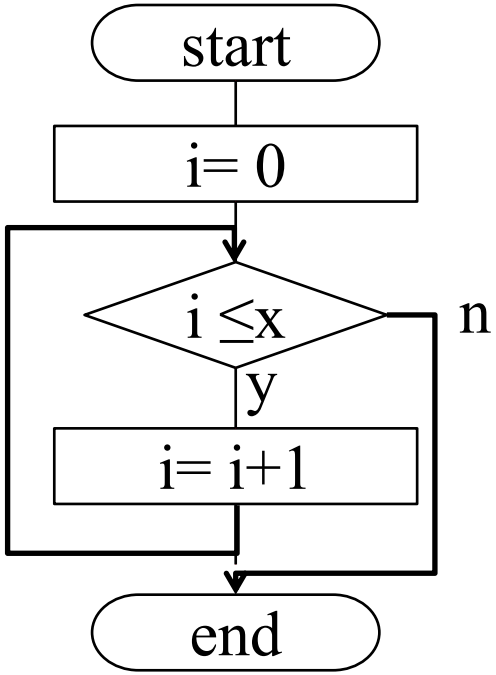
- 定義
  - 19世紀ペルシャの数学者 “Al-Khwarizmi”
  - 「入力と出力が明確に定義された計算手続き」
  - 確定的（例外．確率的アルゴリズム）
  - 言語非依存

フワーリズミー

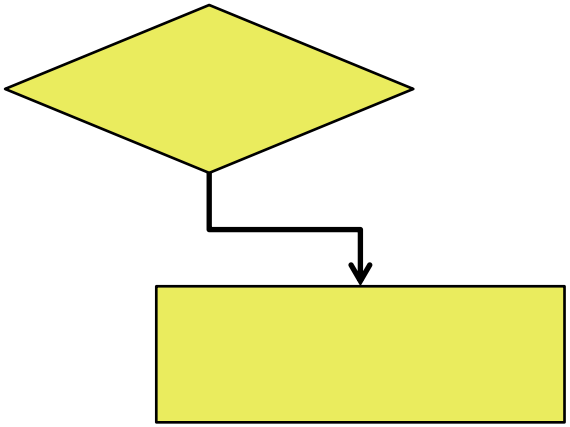
1. 流れ図★頻出

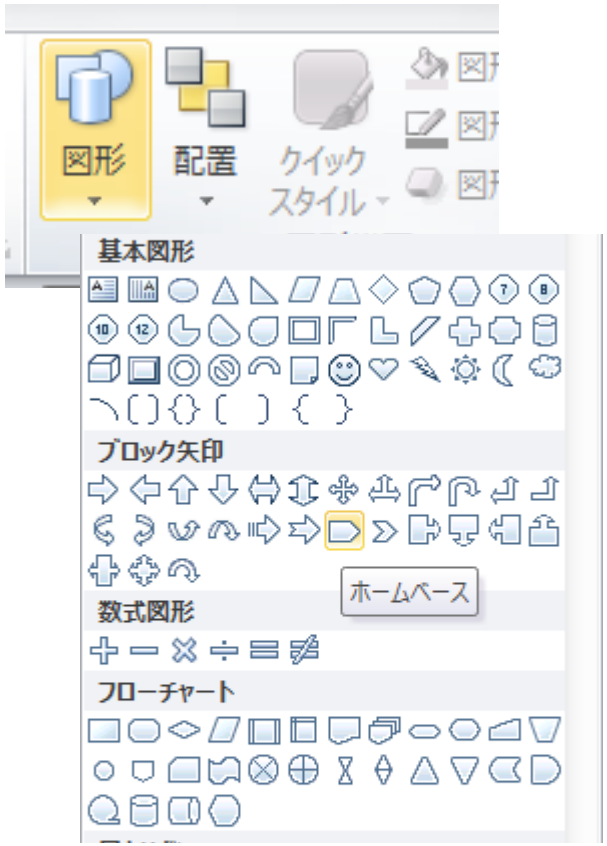
- フローチャート (flowchart)
  - アルゴリズムの図式での記述法
  - JIX X 0121-1986で標準化

```
int func(int x){
  int i = 0;
  while(i <= x){
    i = i+1;
  }
  return(i)
}
```



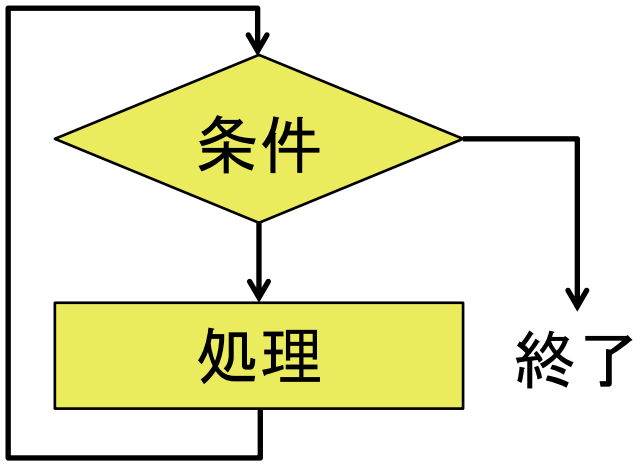
Power Pointでフローチャート



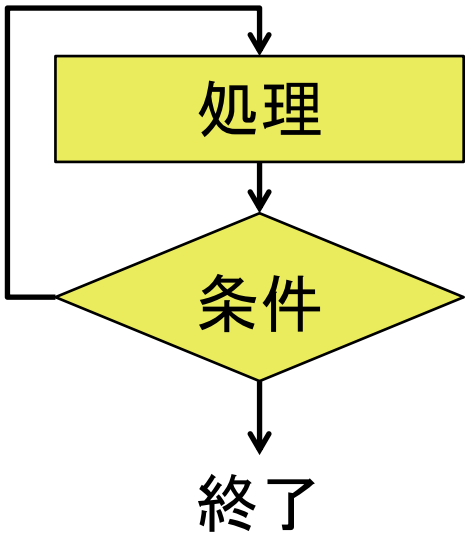


代表的な制御構造

- 事前判定型
  - **while**(条件) {
    処理
    }

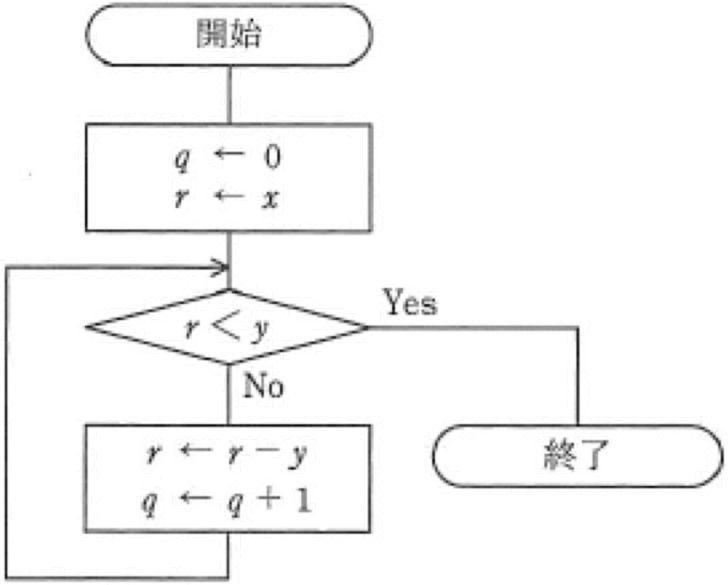


- 事後判定型
  - **do**{
    処理
    }**while**(条件);



例1

- xとyを自然数とする.
  - 次の流れ図で示される手続きを実行した時，qとrの値を求めよ.
  - (x = 10, y = 3の時，qとrはどうなるか?)



フローチャートの問題

- 傾向
  - 決まったパターンはない.
  - 手続き通り実行すると何が生じるかを問う.
  - 特殊なフローチャート記法を覚える必要はない.
  - アルゴリズムの基本知識を要する(論理演算, 2進数など)ことが多い.

2. 整列★頻出

- 問題
  - 次のデータを小さい順に並び替えよ.

    5, 2, 4, 6, 1, 3, 2, 7

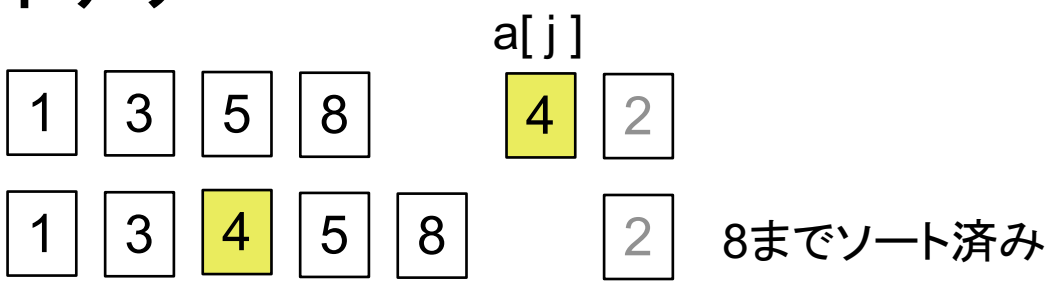
  - 小さい順 (1,2,…)　昇順
  - 大きい順 (7,6,..)　降順

InsertSort(A) 基本挿入法

```
1.  void sort(int a[ ]){
2.          for(int j = 1; j < a.length; ++j){
3.              int k = a[j];
4.              int i = j-1;
5.              while(i >= 0 && a[i] > k){
6.                  a[i+1] = a[i];
7.                  i = i - 1;
8.              }
9.              a[i+1] = k;
10.         }
11.     }
```

基本挿入法のしくみ

- アイデア



- 実行例
  - 5, 2, 4, 6, 1
  - 2, 5, 4, 6, 1
  - 2, 4, 5, 6, 1
  - 2, 4, 5, 6, 1
  - 1, 2, 4, 5, 6

例2

- 次のデータを昇順のInsertSortにかける.

  8, 7, 6, 5, 4, 3, 2, 1

  1. 外側のループ(2行目から10行目)を一回実行した結果を示せ.
  2. ソートが完了するまでに，交換（6行目)は何回実行されるか?

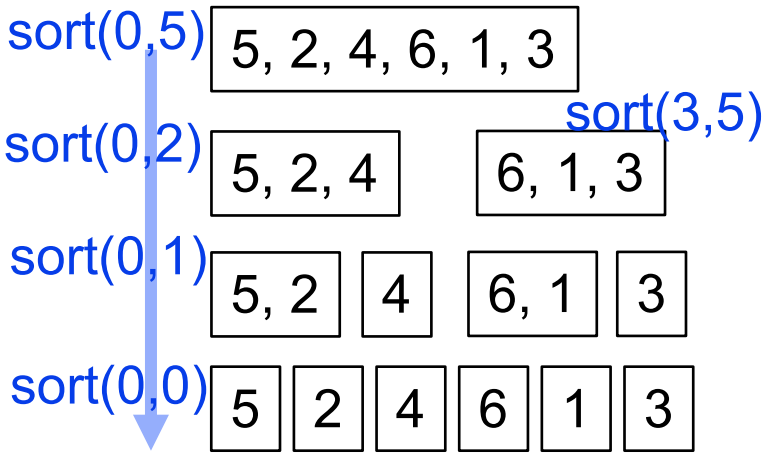
MergeSort(A) マージソート

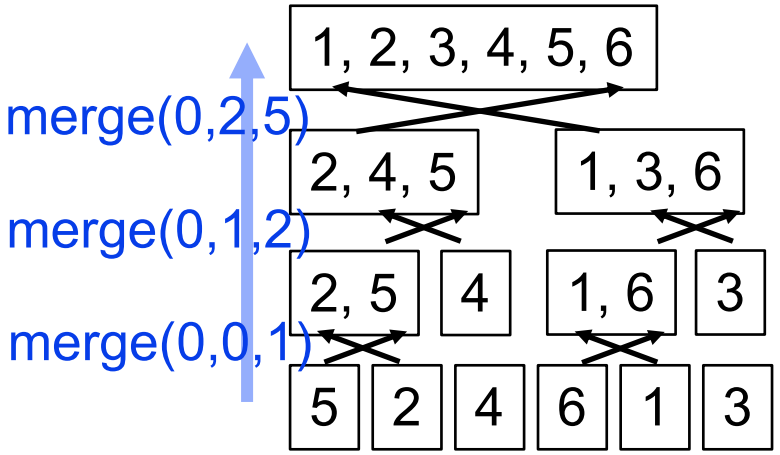
```
1. void sort(int a[], int p, int r){
2.                 if(p == r) return;
3.                 int q = floor((p+r)/2);
4.                 sort(a, p, q);
5.                 sort(a, q+1, r);
6.                 merge(a, p, q, r);
7.         }
8. void merge(int a[], int p, int q, int r){
9.         // a[p..q] と a[q+1..r] の二つの整列済み数列をマージする }
```

マージソートのしくみ

- アイデア
  - 小さな束ほどソートが易しい（分割統治法）
  - A: 2 4 5 + 1 3 6 （二つのソート済み列）

- 実行例





例3

- 次のデータを昇順のMergeSortにかける.

  8, 7, 6, 5, 4, 3, 2, 1

  1. 最初に終了条件(2行目 p == r)が充足するまでに，何回再帰が行われるか?
  2. 1024個のデータを整列するには，何回の再帰が必要か?

問題

- InsertSortとMergeSortのどちらが優れているか？

  - よいアルゴリズムの条件

  1. 正しい
  2. シンプルで読み易く書き易い
  3. 早い（処理時間，メモリ消費が少ない）

InsertSort(A) 最良の時

```
public static void sort(int a[]){   a = {1, 3, 7, 8, 10}
        for(int j = 1; j < a.length; ++j){
            int k = a[j];
            int i = j-1;
            while(i > 0 && a[i] > k){
                a[i+1] = a[i];
                i = i - 1;
            }
            a[i+1] = k;
        }
}
```

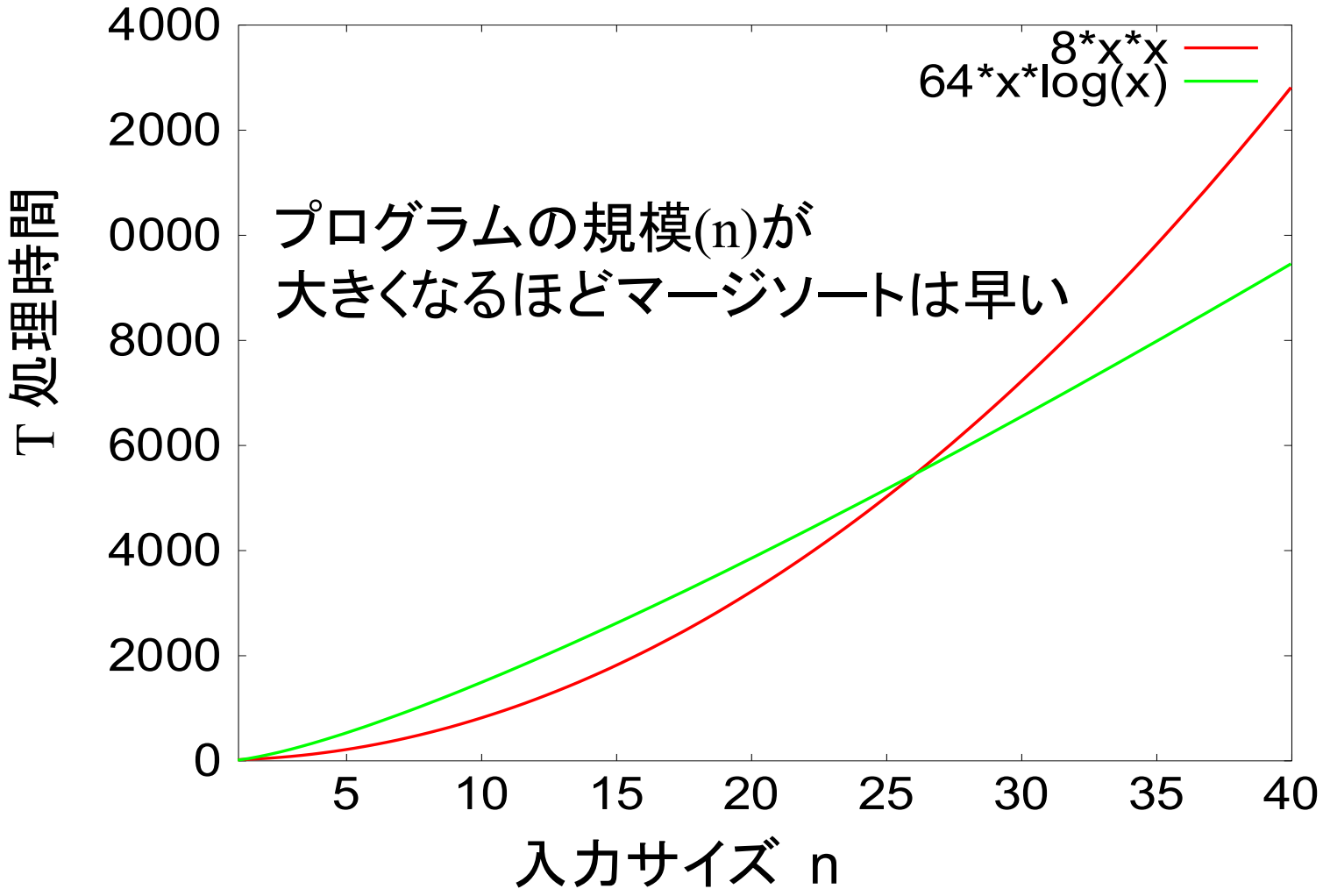
T: n = 5に比例した時間

InsertSort(A) 最悪の時

```
public static void sort(int a[]){   a = {10, 8, 7, 3, 1}
        for(int j = 1; j < a.length; ++j){
            int k = a[j];
            int i = j-1;
            while(i > 0 && a[i] > k){     ; 1, 2, 3, 4
                a[i+1] = a[i];
                i = i - 1;
            }
            a[i+1] = k;
        }
}
```

T: $1+2+3+4 = (n^2+n)/2 < n^2$に比例した時間

Insert SortとMerge Sort



その他のソート

- バブルソート（隣接交換法）
  - 隣あるデータの比較（泡が浮かぶように整列）
- ヒープソート
  - 2項ヒープというデータ構造（配列）にデータを格納する．ヒープ化処理．
- シェルソート（改良挿入法）
  - 隣り合う要素でなく一定間隔で取り出した要素をそれぞれ整列する．
- クィックソート★
  - 「クイック」なソート．ピポットを基準にデータを二つの山に分割しながらソートしていく

Binary Heap

- 2分木を表す配列A

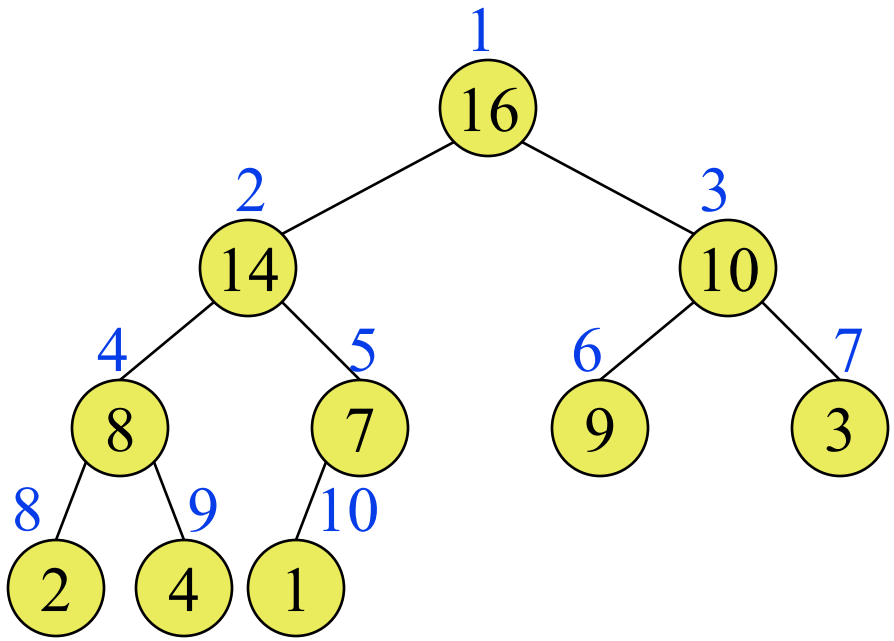
  p(*i*) = floor(*i* /2)

  left(*i*) = 2* *i*

  right(*i*) = 2* *i* +1

- ヒープ条件

  key[p[x]] ≧ key[x]



	1	2	3	4	5	6	7	8	9	10
A	16	14	10	8	7	9	3	2	4	1

QuickSort

```
1.  void qsort(int a[],int p,int r)
2.      if(p < r){
3.        int q = part(a, p, r);
4.        qsort(a, p, q);
5.        qsort(a, q+1, r);
6.      }
7.  }
8.  Int part(int a[], int p, int r){
9.      int x = a[p];
10.     int i = p -1;
11.     int j = r + 1;
12.     while(true){
13.       while(a[--j]> x);
14.       while(a[++i]< x);
15.       if(i < j){
16.         int k = a[i];
17.         a[i] = a[j];
18.         a[j] = k;
19.       }else{
20.         return(j);
21.       }
22. }
```

ピポット

5	3	2	6	4	7
4	3	2	6	5	7
4	3	2	**5**	6	7

5以下

5以上

例3. (練習問題 4)

- 問 配列A[i] (i = 1,..,n)を次のアルゴリズムで整列する．行2-3の処理が初めて終了したとき，必ず実現されている状態は？

1. iを1からn-1まで1づつ増やし行2-3を繰り返す．
2. 　jをnからi+1まで減らしながら行3を繰り返す
3. 　　もし A[j-1] > A[j]ならばA[j]とA[j-1]を交換する

❑ア A[1]が最小値　　イ A[1]が最大値
　ウ A[n]が最小値　　エ A[n]が最大値

まとめ

- アルゴリズムを線と記号の組み合わせで記述する流れ図を（　　）という．事後判定型反復は（　　）文を表す．
- データを小さい順に整列することを（　　）という．
- （　　）は，基準値についてデータ列を二つの組に分割し，それを再帰的に繰り返す．
- （　　）は，整列済みの列の正しい位置にデータを挿入していく．
- （　　）は，データ列を分割し，統合しながら整列する．