

# ソフトウェア基礎論

## 第9回講義

情報工学科

田中哲雄

# 講義の計画

## 序論

1. アルゴリズム, 基本3構造, 段階的詳細化

## アルゴリズムの具体例

2. 変数と代入, 実行過程
3. 型と変数宣言, 配列
4. 関数
5. 再帰的アルゴリズム
6. フィボナッチ数列, パスカルの3角形, ハノイの塔
7. 前半のまとめ
8. まとめとテスト



9. 探索アルゴリズム, ポインタ型, 型宣言

10. リスト

11. 二分木, 二分探索木

## アルゴリズムの理論

12. 数学的帰納法, アルゴリズムの正当性, 停止性

13. 構造帰納法, 再帰的データ構造と正当性

14. アルゴリズムの複雑さ

15. まとめとテスト

# 今日の講義の目標と内容

## ■ 目標

- 二種類の探索アルゴリズムを理解する。
- 再帰的アルゴリズムの理解を深める。
- ポインタ型, および, ポインタの演算について理解する。

## ■ 内容

1. 逐次探索アルゴリズム 反復的解法
2. 逐次探索アルゴリズム 再帰的解法
3. 末尾再帰
4. 二分探索アルゴリズム 反復的解法
5. 二分探索アルゴリズム 再帰的解法
6. ポインタ型
7. 型宣言

# 1. 逐次探索アルゴリズム 反復的解法

## (1)考え方

- 名前を表す文字列の配列から, 与えられた名前の位置(添字, インデックス)を求める **逐次探索** アルゴリズム
  - **線形探索**, **リニアサーチ** ( linear search )
- 反復的解法
  - 配列の名前を先頭から順に調べ, 与えられた名前と要素が一致すれば, 探索を打ち切り, その要素の添字を返す
  - 最後まで一致しなければ-1を返す
- 例 下記のテーブル t からJohnを探す

	0	1	2	3	4	5	6	7	8
t	Al	Bill	Fred	Jill	John	Kate	Mary	Sam	Ted

↑↓  
Johnと比較

t[4]がJohnと一致したので 4 を返す

# 1. 逐次探索アルゴリズム 反復的解法 (2)追跡

```

1  int linearSearch
2    (string t[], int n, string s ){
3      int i;
4      i = 0;
5      while ( i < n ) {
6          if ( t[i] == s )
7              return i;
8          i++;
9      }
10     return -1;
11 }
  
```

■ sがJohnの場合を追跡せよ

0	1	2	3	4	5	6	7	8
Al	Bill	Fred	Jill	John	Kate	Mary	Sam	Ted

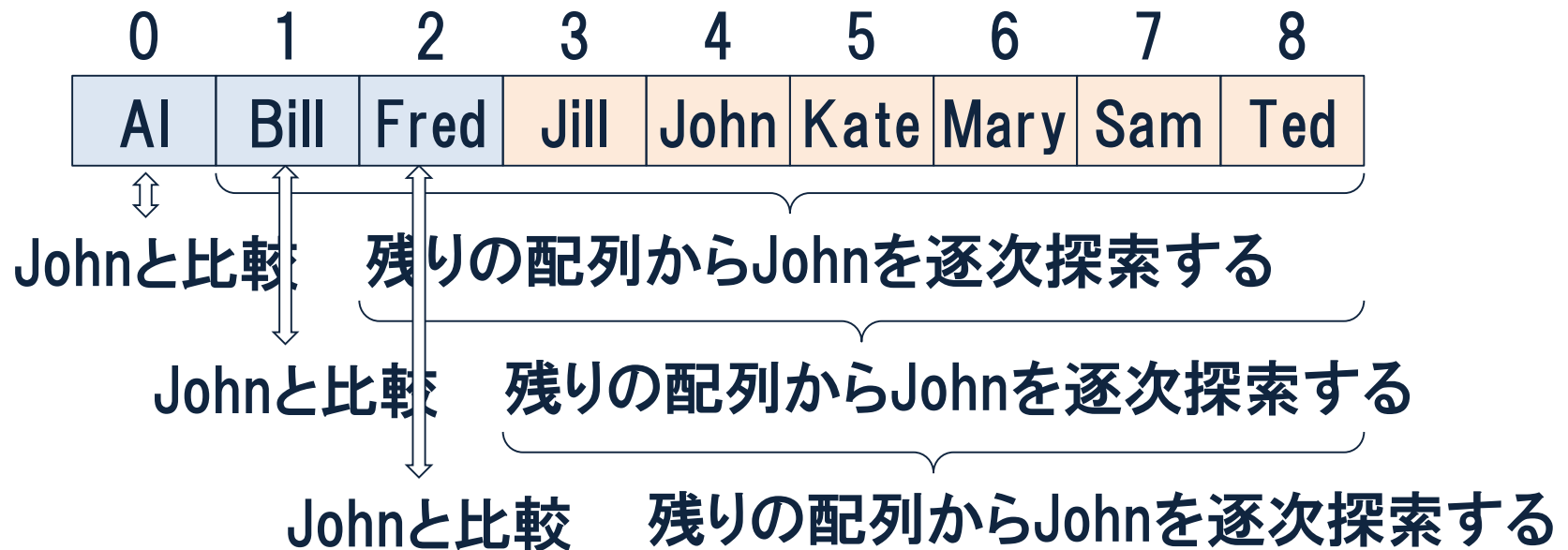
ステップ	関数値	i	i < n	t[i] == s
1		0		
2	演習	↓	真	
3		↓		偽
5		1		
2		↓	真	
3		↓		偽
5		2		
2		↓	真	
3		↓		偽
5		3		
2		↓	真	
3		↓		偽
5		4		
2		↓	真	
3		↓		真
4	4	↓		

## 2. 逐次探索アルゴリズム 再帰的解法

### (1)考え方

#### ■ 逐次探索の再帰的構造

- 配列が空でなければ
  - ◆ 先頭の要素と一致するか調べる(一致したらその添字を返す)
  - ◆ 残りの配列を逐次探索する
- 配列が空ならば-1を返す



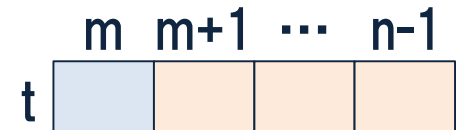
## 2. 逐次探索アルゴリズム 再帰的解法

### (2)アルゴリズム

#### ■ 反復的解法と同様に先頭の要素から探索するには

- 左端の要素と右側の残りの配列ととらえる  
従って, 先頭の要素の添字を引数として渡す
- 先頭の要素と名前が一致すれば探索を打ち切るので  
**行きがけ**に処理する

#### ■ 再帰的アルゴリズムは次のようになる



```
int linearSearch(stirng t[], int m, int n, string s ){  
1   if (m >= n) return -1;  
2   else if (t[m] == s) return m ;  
3   else return linearSearch(t, m+1 , n, s)  
}
```

呼出し例 linearSearch(table, 0, 9, “John”);

## 2. 逐次探索アルゴリズム 再帰的解法

### (3)実行過程の追跡

#### 演習

linearSearch(table, 0, 9, "John")

0	1	2	3	4	5	6	7	8
Al	Bi	Fr	Ji	Jo	Ka	Ma	Sa	Te

ステップ 3 linearSearch(table, 1, 9, "John")

Bi	Fr	Ji	Jo	Ka	Ma	Sa	Te
----	----	----	----	----	----	----	----

ステップ 3 linearSearch(table, 2, 9, "John")

Fr	Ji	Jo	Ka	Ma	Sa	Te
----	----	----	----	----	----	----

ステップ 3 linearSearch(table, 3, 9, "John")

Ji	Jo	Ka	Ma	Sa	Te
----	----	----	----	----	----

ステップ 3 linearSearch(table, 4, 9, "John")

Jo	Ka	Ma	Sa	Te
----	----	----	----	----

ステップ 2 return 4



### 3. 末尾再帰 (1)意味


#### ■ 末尾再帰(tail recursion)

- 関数の最後のステップが再帰呼び出しになっていること
- 言い換えると、帰りがけに何もしない再帰関数のこと

#### ■ 末尾再帰構造は、反復構造に機械的に変換できる (末尾再帰の除去)

##### 末尾再帰

```
T f (T1 a1, T2 a2, ..., Tn an ) {  
    P  
    return f(b1, b2, ..., bn);  
}
```



##### 反復構造

```
T f (T1 a1, T2 a2, ..., Tn an ) {  
    while (1) {  
        P  
        a1 = b1;  
        a2 = b2;  
        :  
        an = bn;  
    }  
}
```

### 3. 末尾再帰 (2)末尾再帰の除去の例

#### ■ 末尾再帰

```
int linearSearch(string t[], int m, int n, string s ){  
    if (m >= n) return -1;  
    else if (t[m] == s) return m;  
    else return linearSearch(t, m+1, n, s)  
}
```

#### ■ 反復構造 末尾再帰の 除去

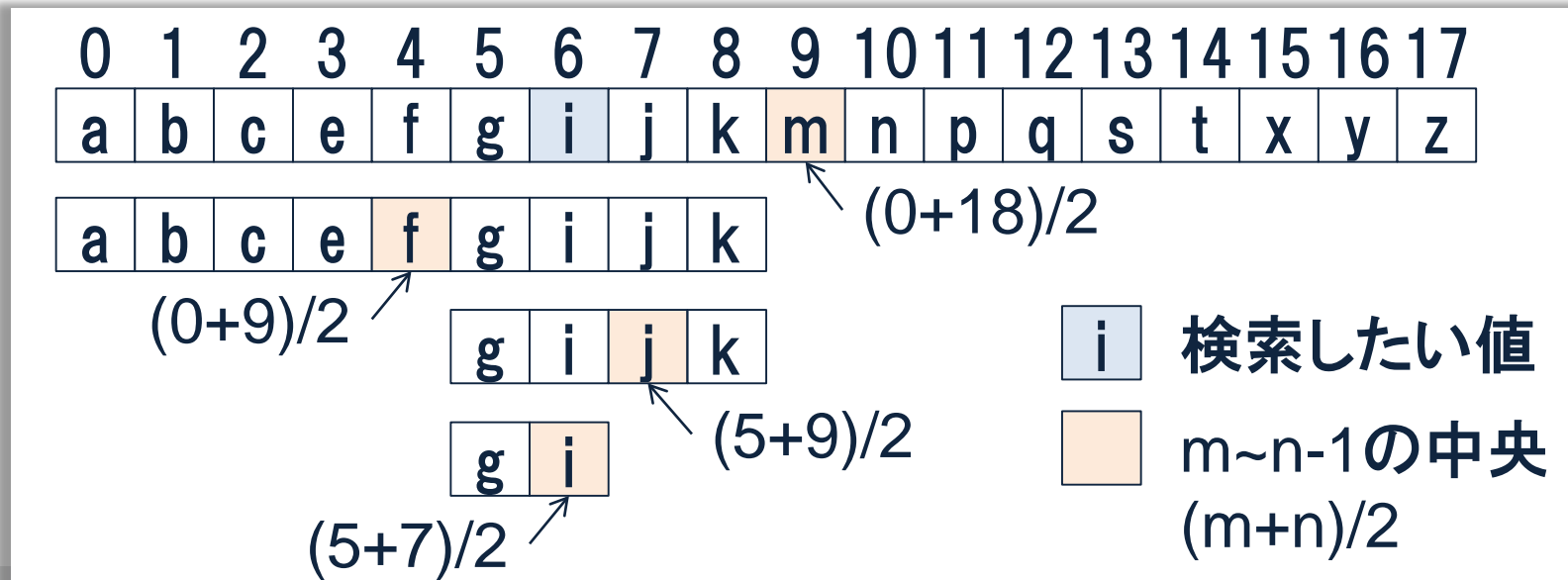
```
int linearSearch(string t[], int m, int n, string s ){  
    while(1) {  
        if (m >= n) return -1;  
        else if (t[m] == s) return m;  
        else {  
            m=m+1;  
        }  
    }  
}
```

## 4. 二分探索アルゴリズム 反復的解法

### (1) 二分探索とは

#### ■ 二分探索 アルゴリズム ( バイナリサーチ )

- 配列がアルファベット順に **整列** されているとき利用できる
- 中央の値を見て、検索したい値が中央の値の右にあるか、左にあるかを判断して、ある方のみを探索していく
- 逐次探索アルゴリズムの実行時間は配列の大きさに比例するが、二分探索はこれより効率がよい

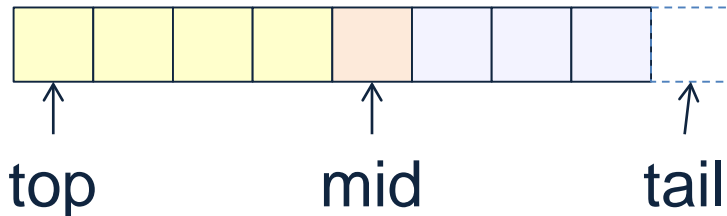


## 4. 二分探索アルゴリズム 反復的解法

### (2)アルゴリズム

#### ■ 反復的解法

- 与えられた名前と**配列の中央の要素**が一致すれば探索を打ち切り, その要素の添字を返す
- 一致しなければ,  
中央より**前にあるはずか**,  
**後ろにあるはずか**を調べ,  
あるはずの部分についてこれを繰り返す
- あるはずの部分が空になったら, -1を返す



```
int binSearch(string t[], int n, string s){
    int top, tail, mid; //先頭, 末尾, 中央
1   top = 0;
2   tail = n;
3   while (top < tail) {
4       mid = (top + tail)/2; //切り捨て
5       if (t[mid] == s) return mid;
6       else if (t[mid] < s) top = mid+1;
7       else tail = mid;
8   }
9   return -1;
}
```

# 4. 二分探索アルゴリズム 反復的解法

## (3)実行過程の追跡

演習

■ 右の配列から  
Samを探索

0	1	2	3	4	5	6	7	8
Al	Bill	Fred	Jill	John	Kate	Mary	Sam	Ted

ステップ	関数値	top	tail	mid	top < tail	t[mid] == s	t[mid] < s
1 top=0		0					
2 tail = n		↓	9				
3 while ...		↓	↓		真		
4 mid= ...		↓	↓	4			
5 if t ...		↓	↓	↓		偽	
6 if t ...		↓	↓	↓			真
6 top=mid+1		5	↓	↓			
3 while ...		↓	↓	↓	真		
4 mid= ...		↓	↓	7			
5 if t ...		↓	↓	↓		真	
5 return mid	7	↓	↓	↓			

## 5. 二分探索 再帰的解法 (1) アルゴリズム

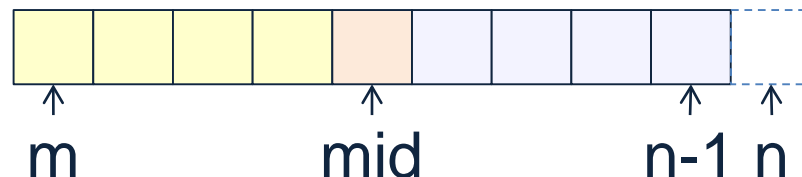
### ■ 二分探索の再帰的構造

- 配列が空でなければ

- ◆ 与えられた名前が**中央の要素と一致するか**を調べる

- ◆ **残りの**右半分または左半分の**配列を二分探索**する

- 配列が空ならば -1 を返す



### ■ 再帰的アルゴリズム

```
int binSearch(stirng t[], int m, int n, string s ){
1   int mid = (m+n)/2;
2   if (m >= n) return -1;
3   else if (t[mid] == s) return mid ;
4   else if (t[mid] < s) return binSearch(t, mid+1 , n, s);
5   else return binSearch(t, m, mid , s);
}
```

## 5. 二分探索 再帰的解法

### (2)実行過程の追跡

#### 演習

- 右の配列tから  
Samを探索

0	1	2	3	4	5	6	7	8
Al	Bill	Fred	Jill	John	Kate	Mary	Sam	Ted

binSearch(t, 0, 9, "Sam")

0	1	2	3	4	5	6	7	8
Al	Bi	Fr	Ji	Jo	Ka	Ma	Sa	Te

ステップ 1 midは 4

ステップ 4 binSearch( t, 5, 9, "Sam" )

5	6	7	8
Ka	Ma	Sa	Te

ステップ 1 midは 7

ステップ 3 return 7