前期末試験問題 (2E 情報処理応用)

電気情報工学科　　学籍番号　　氏名

2007 年 8 月 2 日

1　データ構造

1.1　リストと配列

[問 1] *10 点*

配列は，データの並びの順序通りにメモリーその値を格納する．データはアドレスの順序通りに並んでいる．配列を使って数列 (63, 27, 82, 79, 12) をメモリーに格納した様子を図 1 に示す．

一方，リストの場合，データの順序とメモリーに格納されている順序は異なる．リストのひとつのデータは，その値と次のデータを表すポインターとがペアになって表現される．次のデータを示すポインターを使うことにより，メモリーのどこに格納しても，リストでは順序の情報を保存することができる．先ほどの数列をリストを使ってメモリーに格納した様子を図 2 に示す．

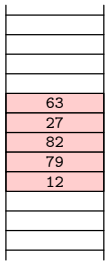


図 1: メモリー中の配列のデータ

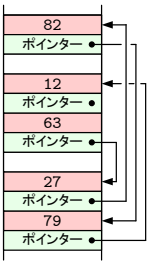


図 2: メモリー中のリストのデータ

[問 2] *10 点*

リストのモデルは，図 3(a) のように書くことができる．この図の 63 と 27 の間に 99 というデータを挿入する．この例を使って，リストの途中にデータを挿入する方法を示す．手順は次の通りである．

- 63 の次のデータを示すポインターを 99 を示すようにする．
- 99 の次のデータを示すポインターを 27 を示すようにする．

この様子を図 3(b) に示す．

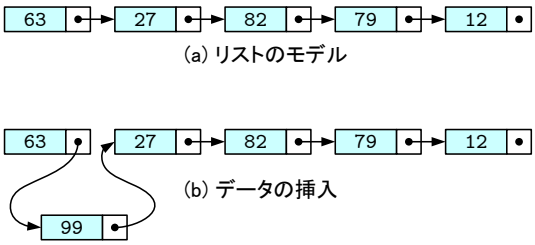


図 3: リストのモデルとデータの挿入

[問 3] *5 点*

```
typedef struct node_tag{
  int value;
  struct node_tag *next;
}node;
```

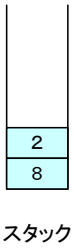
[問 4] *各 1 点*

[ア] J　　[イ] I　　[ウ] A　　[エ] B　　[オ] B　　[カ] A　　[キ] G　　[ク] E

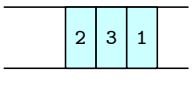
1.2 スタックとキュー

[**問 1**]

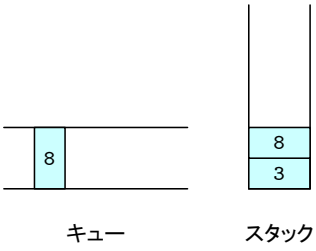
(1) 3 点



(2) 3 点



(3) 3 点

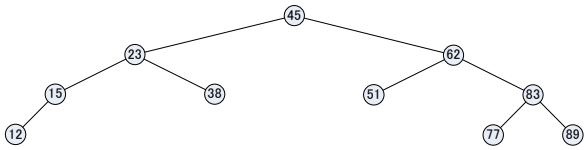


[**問 2**] 5 点

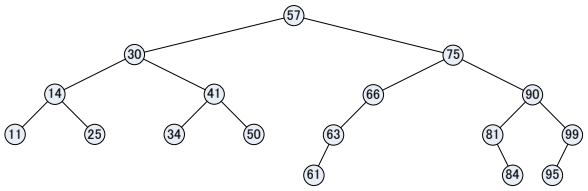
27

1.3 リスト

[**問 1**] 10 点



[**問 2**] 10 点



2 ソートのアルゴリズム

2.1 単純挿入ソート

[**問 1**] 10 点

単純挿入ソートは，a[0] から a[i-1] まで並び替えが終わっているとき，a[i] を正しい位置に入れる—ことを繰り返す方法である．具体的な例を使って，そのアルゴリズムを示す．アルゴリズムは，図 4 の通りで，処理の手順は次のようになる．

1. 処理する a[i] の値 35 よりも，48 は大きいので 35 の場所へ移動させる．
2. 次の 44 も 35 より大きいので，48 の場所へ移動させる．
3. 次の 41 も 35 より大きいので，44 の場所へ移動させる．
4. 次の 39 も 35 より大きいので，41 の場所へ移動させる．
5. 次の 32 は 35 より小さいので，それは移動させない．そして，元の 35 を 39 の場所へ移動させる．

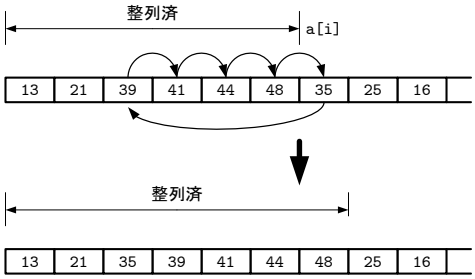


図 4: 単純挿入ソートの基本操作．a[i] を正しい位置に入れる．

[**問 2**] 20 点

```
void sort(int a[], int n)
{
  int i,j, test;

  for(i=1; i<n; i++){
    test=a[i];
    j=i-1;
    while(0<=j && test<a[j]){
      a[j+1]=a[j];
      j--;
    }
    a[j+1]=test;
  }
}
```

2.2 その他のアルゴリズム

[**問 1**] 3 点

例えば，最小値を探して，それを並べるというアルゴリズムを考えることができる．

```
void sort(int a[], int n)
{
  int i, j, minv, mini, tmp;

  for(i=0; i<n; i++){
    minv=a[i];
    mini=i;

    for(j=i+1; j<n; j++){
      if(a[j] < minv){
        minv = a[j];
        mini = j;
      }

    }

  tmp=a[i];
  a[i]=a[mini];
  a[mini]=tmp;

  }

}
```