アルゴリズムとデータ構造 中間試験 模範解答

2024 年 11 月 27 日，担当：井上

1. 以下の文章中の空欄 **(A)**～**(V)** に入るものを選択肢から選び，記せ．(44 点：各 2 点)

■ 特定の問題をコンピュータで解く場合，コンピュータには人間のような常識や経験的直感がないので，問題を解く手順である **(A：アルゴリズム)** とその過程で扱うデータをどう管理するかを定める **(B：データ構造)** とを詳細に記述する必要がある．

■ 2 数の最大公約数を求めるのに使える (A：アルゴリズム) として，**(C：ユークリッドの互除法)** がある．また，(C：ユークリッドの互除法) による処理の過程で効率を高めるには，何度も同じ数の引き算をする代わりに **(D：剰余)** の計算を用いることができる．

■ 解析的に解けない数理的計算をコンピュータを用いて行うことを **(E：数値計算)** という．ただし，コンピュータでの計算は常に近似的で，誤差を含む．例えば，データ型の桁数制限のために四捨五入を行うことによる誤差を **(F：丸め誤差)** という．また，無限級数の計算を有限時間で打ち切ることで生じる誤差は **(G：打ち切り誤差)** である．

■ 解析的に解けない積分計算を (E:数値計算) により解くことを **(H：数値積分)** という．その初歩的な方法として，積分区間を細かく区分し，各区分の積分値を台形の面積として近似する方法を **(I：台形則)** という．さらに精度を上げる方法としては，区間積分値を放物線とx軸で囲まれた領域の面積として近似する方法があり，これは **(J：シンプソン則)** と呼ばれる．

■ (A：アルゴリズム) の良し悪しを定量的に評価する方法として，特定の (A：アルゴリズム) で問題を解く際の繰り返し回数等を問題のサイズ nで関数化したものを **(K：計算量)** という．また，(K：計算量) の値自体は用いるコンピュータ等によって変動するので，例えば多項式なら最大次数以外の項を無視したり，定数係数を無視したりして (K:計算量) がどのような関数であるかだけを表した表記とする．これを (K：計算量) の **(L：オーダ)** という．

■ 多数のデータを小さい順などに並べ直す作業を **(M：ソート)** といい，それに対して様々な (A：アルゴリズム) が提案されている．例えば，データ中の最小のものをデータ列の先頭と交換する操作を繰り返す方法は **(N：直接選択法)** であり，途中まで (M：ソート) できているデータ列に次のデータを挿入することを繰り返す方法を **(O：基本挿入法)** という．これら 2 つの (A：アルゴリズム) の (K：計算量) の (L：オーダ) は **(P：n^2)** である．

■ 多数のデータから目標データを探し出す作業を **(Q：サーチ)** という．単純にデータを一つずつ調べる逐次探索に対して，**(R：ハッシュ)** という方法では，探索の手がかり（キー）を有限の数値に写像し，その数値に対応した配列要素にデータを格納することで (Q：サーチ) を高速化している．

■ 解くべき問題を「同じ解き方で解ける，サイズの小さな副問題」に分解して副問題を先に解くという考え方で関数を作る場合，問題を解く関数はその処理の中で，自分と同じ関数をより小さな引数を使って呼び出すような構造となる．このような手法を **(S：再帰)** という．

■ (S：再帰) を用いた (M：ソート) の (A：アルゴリズム) に **(T：クイック・ソート)** がある．(T：クイック・ソート) では，データ列を「軸」と呼ばれる基準値に基づいて 2 つのグループに分割するという操作を (S:再帰) を用いて繰り返す．その (K：計算量) の (L：オーダ) は **(U：$n\log n$)** である．

■ データの一時保管場所として広く用いられる (B:データ構造) の一つに **(V：スタック)** がある．(V：スタック) では，最後に格納したデータが最初に取り出される LIFO (last in first out) 方式でデータが扱われ，(S：再帰) における関数呼び出しにおいて関数の内部データを格納しておくなどの用途に使用される．

採点基準

- (K)と(L)：逆の場合は各 1 点

2. 以下は，教科書に書かれた基本挿入法のソース・コードを改変したものである．変更部分 (太字・下線) は，データ列の初期化，データ列の表示の関数化，およびソートの途中経過の表示に関するものである．これについて，以下の問に答えよ．(24 点：それぞれ 12 点．(1) は各 3 点)

```
#include <stdio.h>
#include <stdlib.h>
#define N 6
void disp(int *);

void main(void)
{
  int a[]={80,41,35,90,40,20};
  int i,j,t;

  disp(a);

  for (i=1; (A：i<N); i++){
    for (j=i-1; (B：j>=0); j--){
      if ((C：a[j]>a[j+1])){
        t=a[j];
        a[j]=a[j+1];
        a[j+1]=t;
      }
      else (D：break);
    }
    disp(a);
  }
}

void disp(int a[])
{
  int i;

  for (i=0; i<N; i++)
```

```
  printf("%d ", a[i]);
  printf("¥n");
}
```

(1) 空欄 **(A)**～**(D)** に入るものを記せ.
(2) プログラム実行時の表示内容を記せ（disp(a)の呼び出し位置に注意すること).

```
80 41 35 90 40 20
41 80 35 90 40 20
35 41 80 90 40 20
35 41 80 90 40 20
35 40 41 80 90 20
20 35 40 41 80 90
```

採点基準

- (1) (A) : N-1 : 1, i<N-1 : 2, i<6 : 2, i=N : 1
- (1) (B) : j>1 : 1, j>0 : 2, j>i : 1, i>=0 : 2, j<=0 : 2
- (1) (C) : a[i]>a[j] : 0, a[j]<a[j-1] : 2
- (1) (D) : return : 1
- (2) : 他のソート法の過程を正しく記載 : 3, 交換が起こるごとに記載 : -2, 1 行目だけ OK : 2, 2 行目まで OK : 4, 4 行目まで OK : 8, 5 行目が抜けた : -2

3. 以下は，教科書に記載された組み合わせの数を再帰的に求める関数 combi() である．ただし，関数が呼び出されるたびに画面表示をするコードを追加した．このとき，combi(4,2) を呼び出したときの画面表示がどうなるかを示せ． (再帰呼び出しの連鎖がどのように起こるかを図に描いてみるなどして考えるのがよい) (16 点)

```
long combi(int n,int r)
{
  printf("%dC%d¥n", n, r);

  if (r==0 || r==n)
    return 1L;
  else
    return combi(n-1,r)+combi(n-1,r-1);
}
```

```
4C2
3C2
2C2
2C1
1C1
1C0
3C1
2C1
1C1
1C0
2C0
```

採点基準

- printf()の指定通りの表記ではない : -2
- 1 行目まで OK : 3, 2 行目まで OK : 4, 3 行目まで OK : 5, 6 行目まで OK : 8, 9 行目まで OK : 9, 再帰呼び出しから戻ったときにも printf : -2, 図の構造は OK : 8, 図に順序まで正しく記載 : 12

4. 以下は, 教科書に書かれたスタックのソース・コードである. ただし, 配列 **stack[]** のサイズは **2** とした. このプログラムを実行し，以下の順序でキー入力をしたときにスタックから取り出されるデータを順序も含めて示せ. (キー操作においてリターンキーは省略した) (16 点)

キー操作 : **i,1,i,2,i,3,o,i,4,i,5,o,o,o**

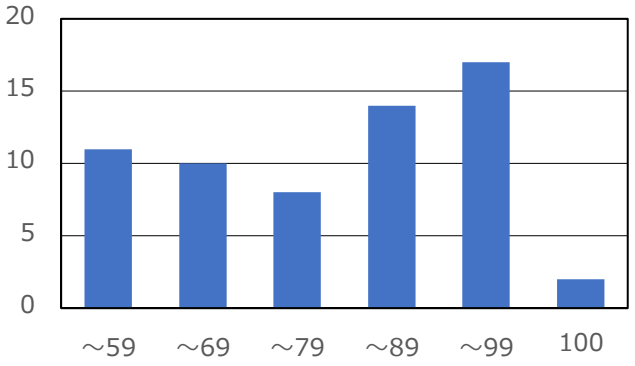
(ソース・コード省略)

取り出されるデータ : 2, 4, 1.

採点基準

- オーバーフローの概念だけ理解 : 2
- スタックの中身の推移は正しく記載 : 10
- サイズ 1 または 3 と誤解して正しく回答 : 13

平均 76.3 点



以上