問

広島市立大学大学院 情報科学研究科

模擬問題Ｂ

データ構造とアルゴリズム

（９０分）

科　　目
データ構造とアルゴリズム

注意事項

本問題は，平成３０年度に実施される広島市立大学大学院情報科学研究科博士前期課程一般入試のために作成した模擬問題です。筆記試験を学習する際の参考資料として使用して下さい。

第１問（１／１）

【問１】アルゴリズムの計算量がオーダー記法で与えられているとする。例にならい，それぞれのオーダー記法を簡略化せよ。

（例）$O(4n^3+2n^4+83)$ →解答例 $O(n^4)$

（１）$O(8n^3+n^2+14)$

（２）$O(n^3\log n+3^n)$

（３）$O(n^2\log n+n^2+4n)$

【問２】図 1.1 の再帰関数 func(n)について考える。この関数 func(n)を引数 n = 3 として呼び出したときの実行の様子を表 1.1 に示す。以下の解説を参考に表の空欄を埋めよ。

（解説）初めに呼ばれた func(3)から func(2)が 2 番目に呼び出される。その後，関数 func()は再帰的に処理され，func(2)の返り値は 18 となり，最後に func(3) の返り値が得られる。

```
int func(int n){
  int f;
  if(n > 6) f = n + 4;
  else if(n % 3 > 0)
       f = func(n + 2) + 3;
  else f = func(n - 1) - 2;
  return f;
}
```

図 1.1　関数 func

表 1.1　func(n)の様子

順序	引数 n	返り値 func(n)
1	3	
2	2	18
3		
4		
5		
6		

第２問（１／３）

配列 data[] に格納された 6 個の整数 data[] = {123, 551, 235, 213, 312, 132}に対して図 2.1 に示す基数ソートアルゴリズム radixsort(data,6,N,K)を実行する(図 2.2 の注意事項も参照すること)。N は基数を表し，ここでは N=10 とする。K は正の整数である。配列の添字は 0 から始まることに注意する。このアルゴリズムについて以下の問いに答えよ。

【問１】行 16 から行 21 の動作を簡潔に説明せよ。

【問２】j=0 のとき，行 22 におけるバケットの様子を，図 2.3 を参考にして記述せよ。また，printdata(data,n)で出力されるものを記述せよ。

```
01: typedef struct cell {
02:   int              val;
03:   struct cell      *next;
04: } Cell;
05:
06: void radixsort(int data[], int n, int rdx, int k) {
07:   int   i, j, m, val, key;
08:   Cell *bucket[rdx], *p;
09:   for(j = 0; j < k; j++) {
10:         initbucket(bucket, rdx);
11:         for(i = 0; i < n; i++) {
12:               val = data[i];
13:               key = rdxkey(val, j, rdx);
14:               append(bucket, key, val);
15:         }
16:         i = 0;
17:         for(m = 0; m < rdx; m++) {
18:               for(p = bucket[m]; p != NULL; p = p->next) {
19:                     data[i++] = p->val;
20:               }
21:         }
22:         printdata(data, n);
23:   }
24: }
```

図 2.1　基数ソートアルゴリズム radixsort

第２問（２／３）

- initbucket(bucket, rdx)：個数 rdx のバケット(rdx 個のリスト)を初期化する。
- rdxkey(val, j, rdx)：整数値 val を，基数を rdx として表現したときの rdx の j 乗の位の数を返す。例えば val=59876, j=3, rdx=10 のとき，9 を返す。
- append(bucket, key, val)：リスト bucket[key]の最後尾に val を追加する。例として，append(bucket, 2, 325)；による bucket[] の変化を図 2.3 に示す。
- printdata(data, n)：整数の配列の先頭から n 個の要素 data[0], data[1],..., data[n-1]を 10 進数で出力する。

図 2.2　基数ソートアルゴリズム radixsort の注意事項

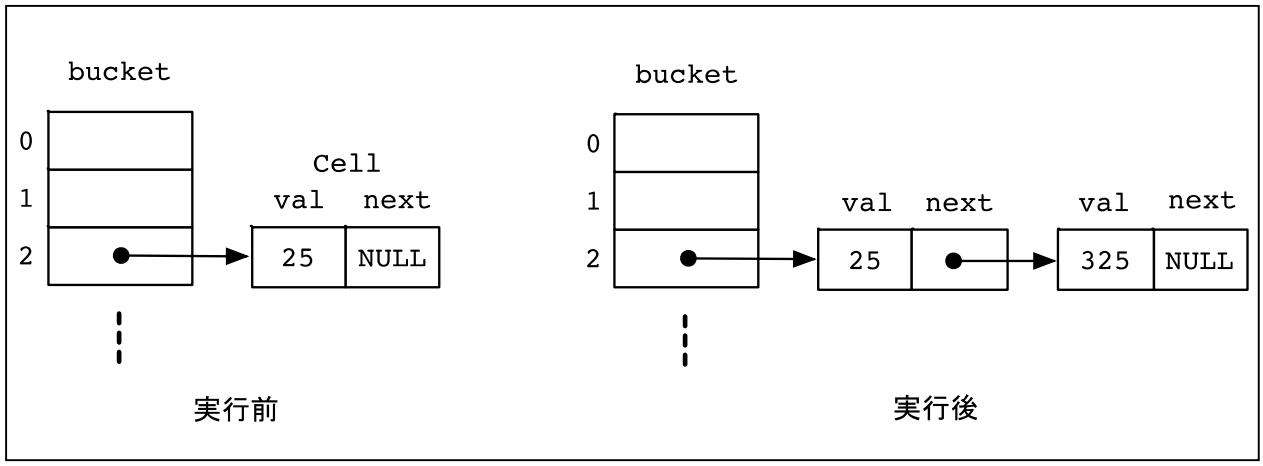

図 2.3　append(bucket, 2, 325)；による bucket[]の変化

【問３】基数 N=10 のとき，このデータ data[]に対してソートが完了するために必要な最小の K を答えよ。あわせてその理由も簡潔に述べよ。

【問４】関数 append のプログラムを図 2.4 に示す。（a）～（d）の空欄に入るものを以下の(ア)～（キ）から選び，記号で答えよ。解答は重複してもよい。

（ア）val（イ）p（ウ）r（エ）NULL（オ）p->next（カ）r->next（キ）bucket[key]

第２問（３／３）

```
void append(Cell **bucket,int key,int val){
   Cell *p,*r;
   r = (Cell*)malloc(sizeof(Cell));
   r->val = val;
   r->next = (a) ;
   if(bucket[key] == NULL) bucket[key] = (b) ;
   else{
      p = bucket[key];
      while(p->next != NULL)  p = (c) ;
      p->next = (d) ;
   }
}
```

図 2.4　関数 append

第３問（１／２）

図 3.1 は，配列 A を整列するための関数 sort と関数 partition である。

```
void sort(int i, int j, int *A){ //配列 A の i 番目から j 番
  目を整列
  int p, k;
  p=A[i]; // 部分配列の左端の要素を軸要素 p とする
  k=partition(i, j, p, A); // k: 分割位置
  if(i<k-1) sort(i,k-1,A);
  if(k<j) sort(k,j,A);
}

int partition(int i, int j, int p, int *A){
  int l, r, k;
  l=i; r=j;
  while(1){
    while(l<j && A[l]<p) l++;
    while(r>i && A[r]>p) r--;
    if(l<=r){
      swap(l, r, A); l++; r--;
      //関数 swap は配列 A の l 番目と r 番目を交換する
    } else break;
  }
  k=l;
  return(k);
}
```

図 3.1　整列プログラムの一部

第３問（２／２）

以下の問いに答えよ。

【問１】この整列アルゴリズムの名称を述べよ。

【問２】今，図 3.1 の関数 partition を，6 個の要素からなる配列に対して適用する。以下に示す配列と軸要素が与えられた時，解答例にならい，適用後の配列および関数 partition の返り値をそれぞれ記述せよ。

配列 A={6, 12, 9, 3, 22, 5}, i=0, j=5, 軸要素 p=6

(解答例)

5	3	9	12	22	6

返り値　2

（１） 配列 A={43, 17, 30, 30, 2, 4}, i=0, j=5, 軸要素 p=43

（２） 配列 A={19, 15, 17, 28, 43, 35}, i=0, j=5, 軸要素 p=19

【問３】図 3.1 に示す関数 sort を，i=0, j=5, 配列 A[]={2, 9, 4, 7, 5, 3}として呼んだ場合，整列が完了するまで複数回関数 sort が呼び出される。関数 sort が呼ばれる毎に，呼ばれた直後の配列 A の内容 A[0]～A[5]をすべて記述せよ。

【問４】以下の空欄を適切に埋めよ。

n 個の要素からなる配列 A を整列する過程において，関数 sort の再帰呼び出しの各レベル(深さ)で必要な時間計算量の合計は $O($ (a) $)$ であり，軸要素 p が常に部分配列の中央値であり,部分配列が常に等しく分割される場合，再帰呼び出しのレベルは全体で $O($ (b) $)$ レベルあるため，整列全体で必要な時間計算量は $O($ (c) $)$ となる。一方，軸要素 p が常に部分配列の最小値(または最大値)である時，再帰呼び出しのレベルは全体で $O($ (d) $)$ レベルあるため，整列全体で必要な時間計算量は $O($ (e) $)$ となる。