2006 年度ソフトウェア工学試験解答例

1.漸近量評価（２５点）

（１） 次の関数を O 記法で示せ。(出来るだけ漸近計算量が小さいもので表すこと。)（１０点、各２点）

$(a)\quad T_a(n) = 0.5n^2 - 10n + 5$

$T_a(n) = O(n^2)$

最高次数の多項式。定数なので、係数は O 記法には表れない。

$(b)\quad T_b(n) = 5(n+1)(n-2)(n+3) + 7(n+5)(n-3) + 32$

$T_b(n) = O(n^3)$

展開するまでもなく３次式である。

$(c)\quad T_c(n) = \sqrt{n} + \log n$

$T_c(n) = O\left(\sqrt{n}\right)$

一般に、任意の微小正定数 $\varepsilon > 0$ に対して、$O(\log n) \subseteq O(n^{\varepsilon})$ である。$O\left(\sqrt{n}\right) = O\left(n^{\frac{1}{2}}\right)$ であるので、上のように求められる。

$(d)\quad T_d(n) = n + \log n + 0.5^n$

$T_d(n) = O(n)$

$0.5^n < 1$ に注意する。

$(e)\quad T_e(n) = n^5 + 2^{\frac{n}{5}}$

$T_e(n) = O\left(2^{\frac{n}{5}}\right)$

一般に、正定数 $K > 0$ に対して、$O\left(n^k\right) \subseteq O(2)^n$ である。

（２）次の擬似コードの漸近計算量を O 記法で示せ。（１５点、各５点）

ただし、以下のコードでは引数 n が入力サイズとし、すべて $n = 2^k$ の形であるとする。

(a) 漸近計算量 $T_A(n)$

アルゴリズム A:

```
void algoA(int n){
    for(i=0;i<n;i++){
        for(j=i;j<n;j++){
            (定数c1時間の処理)
        }
    }
    return;
}
```

内側のループの回数は外側の添え字による。

$$T_A(n) = c_1(1+2+\cdots+n)$$

$$= c_1 \frac{n(n+1)}{2}$$

$$= \frac{c_1}{2}n^2 + \frac{c_1}{2}n$$

$$\therefore T_A(n) = O(n^2)$$

(b) 漸近計算量 $T_B(n)$

アルゴリズム B :

```
void algoB(int n){
    for(i=2;i<n;i=i*i){
            (定数c2時間の処理)
    }
    return;
}
```

繰り返しの回数は K とすると次式が成り立つ。

$$2^K = n$$

$$\therefore K = k = \log n$$

$$\therefore T_B(n) = O(\log n)$$

(*c*) 漸近計算量$T_C(n)$

アルゴリズム C：

```
void algoC(int n){
   if(n<=0){
      return;
   }else{
     algoC(n-1);
       (定数c3時間の処理)
     algoC(n-1);
     return;
   }
}
```

まず、時間計算量$T_C(n)$に関して次の漸化式が成り立つ。

$$T_C(n)=\begin{cases} c & n=0 \\ 2T_C(n-1)+c_3 & n>0 \end{cases}$$

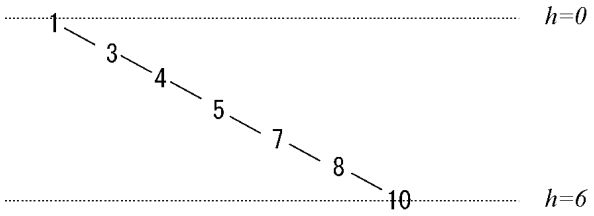
この漸化式より、$T_C(n)$を求める。

$$\begin{aligned}
&T_C(n)=2T_C(n-1)+c_3 \\
&=2\{2T_C(n-2)+c_3\}+c_3=4T_C(n-2)+(2+1)c_3 \\
&=4\{2T_C(n-3)+c_3\}+(2+1)c_3=8T_C(n-3)+(4+2+1)c_3 \\
&\vdots \\
&=2^nT(0)+(2^n-1)c_3 \\
&=(c+c_3)2^n-c_3 \\
&\therefore T_C(n)=O(2^n)
\end{aligned}$$

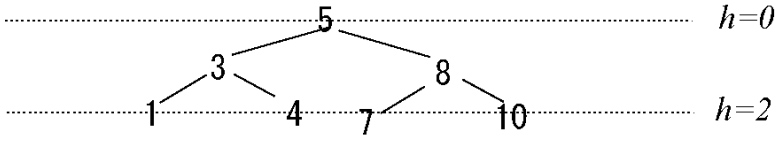
２．データ構造（２５点）

２分探索木 T に集合$S=\{1,3,4,5,7,8,10\}$を蓄えることを考える。２分探索木 T にデータ x を挿入する操作を `insert(x)`で表し、２分探索木 T からデータ x を削除する操作を `delete(x)`で表す。このとき、以下の問いに答えよ。

（１）集合Sを蓄える２分探索木で高さが最大のものを１つ図示せよ。（１種類示せばよい。）例えば、次の図のような２分探索木が高さが最大である。（2 点）



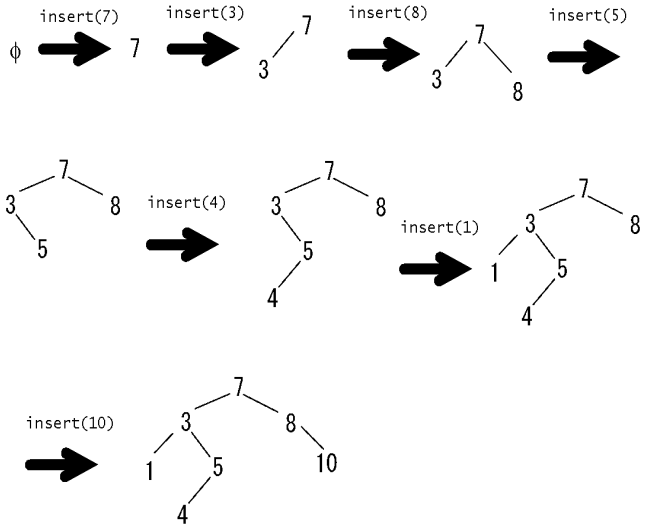
（２)集合Sを蓄える２分探索木で高さが最小のものを１つ図示せよ。(１種類示せばよい。)例えば、次の図のような２分探索木が高さが最小である。(この場合、２分探索木は完全２分木である。高さの最小の木はこの１種類しか存在しない。)（2 点）



（３）空の（要素数０の）２分探索木 T に次の順序で集合の要素を挿入するとき、２分探索木の状態の変化を図示せよ。(7 点、各１点)

`insert(7)`→`insert(3)`→`insert(8)`→`insert(5)`→`insert(4)`→`insert(1)`→`insert(10)`

挿入する値の大小にしたがって、根から順に NULL 要素までたどる。その場所に挿入する。なお、一度挿入された頂点は変更されない。

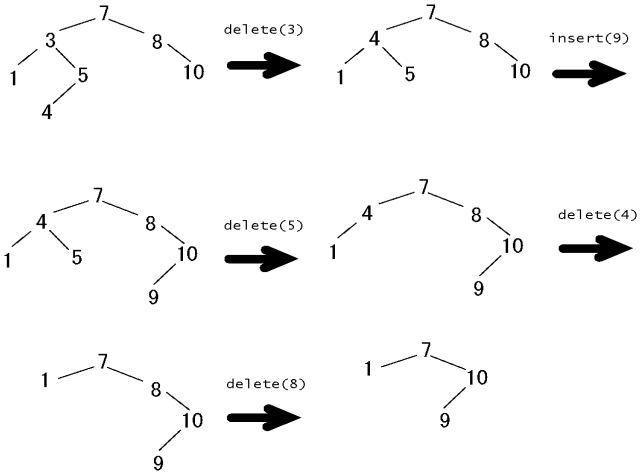


（４）（３）の木から次の操作を行ったときの２分探索木を変化を図示せよ。

削除において、削除する頂点に２つの子がある場合に、２種類の動作が可能である。次の解答例では、右の子を根とする部分木の最小の要素で置き換えるように示してある。

(左の子を根とする部分木の最大の要素で置き換えるような動作としても良い。ただし、

混ぜている場合は減点する。）（9 点、insert１点、delete２点）
delete(3)→insert(9)→delete(5)→delete(4)→delete(8)
子供の数に応じて場合わけを行う。
子供がいない場合：単純にその頂点を削除する。
子供が１の場合：その頂点を削除して、子頂点を親に接続する。
子供が２の場合：右の部分木の最小値を削除する頂点の位置に移動し、右の部分木において最小値を削除する。（左の最大値にすることもできる。）



（５）空の２分探索からある順序で挿入を繰り返して、(2)の木が得られた。このような挿入順序を１つ示せ。（５点）
例えば、次のような挿入順序にすれば良い。
（２）の木をなぞったときの先行順（初めて頂点に到達する順番）で挿入を行えばよい。なぞりかたは複数考えられるが、代表的な２つを示す。）

（幅優先型）
insert(5)→insert(3)→insert(8)→insert(1)→insert(4)→insert(7)→insert(10)

（深さ優先型）
insert(5)→insert(3)→insert(1)→insert(4)→insert(8)→insert(7)→insert(10)

3.ヒープソート（２５点）

配列 A には n 個の要素 $a_0, a_1, \cdots, a_{n-1}$ が $A[0], A[1], \cdots, A[n-1]$ にそれぞれ蓄えられているとする。この（単一の）配列 A だけを用いてヒープソートを行うアルゴリズムを下に擬似コードで示す。ここで、ヒープは親の方が大きい要素を蓄えるものとする。（すなわち、ある頂点 v を根とする部分木中の最大の値を頂点 v に蓄える。）また、print_array(n,A)は配列 A の最初の n 要素の内容を表示する関数である。このアルゴリズムについて問いに答えよ。

ヒープソート：

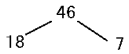
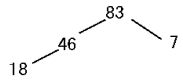
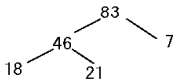
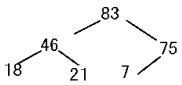
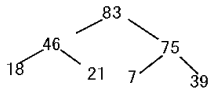
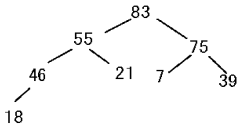
```
void h_sort(int n,int A[]){
  /*ヒープ状態作成*/
  for(i=0;i<n;i++){
       （A[i]をヒープに挿入し、A[0]から A[i]までをヒープ状態にする。）
       print_array(n,A);
   }
  /*並べ変え*/
  for(i=n-1;i>0;i--){
       （A[i]と A[0]を交換し、A[i]から A[n-1]までをソート状態にする。）
       print_array(n,A);
   }
  return;
}
```

(1) $n=8$ とし、配列 A に下のように値が蓄えられているとする。h_sort(8,A[])を実行したときに print_array(n,A)の各命令により表示される配列の内容を示せ。また、print_array(n,A)の各命令が実行されたときのヒープの状態を木として図示せよ。（１０点）

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
A	18	46	7	83	21	75	39	55

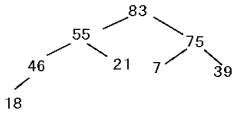
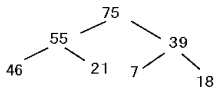
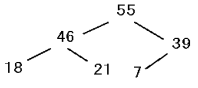
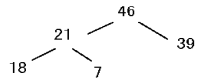
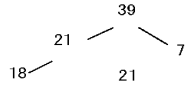
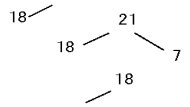
配列

ヒープ

配列	ヒープ	
18 | 46 7 83 21 75 39 55	φ	初期状態
18 | 46 7 83 21 75 39 55	18	ヒープ状態作成
46 18 | 7 83 21 75 39 55		
46 18 7 | 83 21 75 39 55		
83 46 7 18 | 21 75 39 55		
83 46 7 18 21 | 75 39 55		
83 46 75 18 21 7 | 39 55		
83 46 75 18 21 7 39 | 55		
83 55 75 46 21 7 39 18 |		

配列

ヒープ

配列	ヒープ	
83 55 75 46 21 7 39 18 |		
75 55 39 46 21 7 18 | 83		並べ替え
55 46 39 18 21 7 | 75 83		
46 21 39 18 7 | 55 75 83		
39 21 7 18 | 46 55 75 83		
21 18 7 | 39 46 55 75 83		
18 7 | 21 39 46 55 75 83		
7 | 18 21 39 46 55 75 83	7	
7 18 21 39 46 55 75 83 |	φ	ソート終了

（２）一般に、高さが h のヒープに蓄えることのできる最大のデータ数 N を求めよ。(なお、根の高さは 0 として扱うこと。)（５点）
深さ $d,(0 \le d \le h)$ において、2^d 個の要素が蓄えられることに注意する。したがって、N は以下のように求められる。

$$N = \sum_{d=0}^{h} 2^d$$
$$= (1 + 2 + \cdots + 2^h)$$
$$= \frac{2^{h+1} - 1}{2 - 1}$$
$$= 2^{h+1} - 1$$

（３）ヒープに n 個のデータが蓄えられているとする。このとき、ヒープの高さ h とデータ数 n との関係を式で表せ。また、高さ h をデータ数 n を用いた O 記法で示せ。（５点）

$$2^h - 1 < n \le 2^{h+1} - 1$$
$$\therefore 2^h < n + 1 \le 2^{h+1}$$
$$\therefore h < \log(n+1) \le h + 1$$
$$\therefore \log(n+1) - 1 \le h < \log(n+1)$$
$$\therefore h = O(\log n)$$

（４）ヒープソートの時間計算量を O 記法で示せ。ただし、`print_array`（`n,A`）に用いられる時間計算量を含めないこと。（５点）

ヒープに対する挿入、最大値削除等の各操作はすべてヒープの高さに比例した時間計算量で行える。よって、（３）より、各操作１回に必要な時間計算量は、$O(\log n)$ である。これらは、ヒープ作成部分では、n 回の挿入が行われ、並べ替え部分では n 回の最大値削除が行われる。したがって、ヒープソートの時間計算量 $T_h(n)$ は次のように求められる。

$$T_h(n) \le \underbrace{n \times c_1 \log n}_{\text{ヒープ作成}} + \underbrace{n \times c_2 \log n}_{\text{並べ替え}}$$
$$\le (c_1 + c_2) n \log n$$
$$\therefore T_h(n) = O(n \log n)$$

４.ハッシュ（２５点）

アルファベット $A = \{a, b, \cdots, z\}$ 中の 3 文字からなる 5 個の名前 $N = \{abe, ebi, kan, ito, oki\}$ を要素数 10 の配列 X にハッシュ法で蓄えることを考える。

ハッシュ関数 $h : A^3 \to \{0,1,2,3,4,5,6,7,8,9\}$ は以下の式を用いるものとする。

$\boldsymbol{x} = x_1 x_2 x_3 \in A^3$ に対して、

$$h(\boldsymbol{x}) \equiv \sum_{i=1}^{3} c(x_i) \pmod{10}$$

ここで、$c : A \to \mathbb{Z}$ は以下に示すようなアスキーコードである。（$\mathbb{Z}$ は整数の集合。）

$c(a) = 97, c(b) = 98, \cdots, c(z) = 122$

また、要素 $\boldsymbol{x}$ を挿入する際に k 回衝突が起きたとするとき、次のハッシュ関数値で得られた値を添え字として配列 X に蓄えるとする。

$h_k(\boldsymbol{x}) \equiv h(\boldsymbol{x}) + k \pmod{10}$

このとき、次の問いに答えよ。

（１） 各要素 $\boldsymbol{x} \in N$ に対して、ハッシュ値 $h(\boldsymbol{x})$ を求めよ。（１０点、各２点）

ハッシュ関数に対して次式が成り立つことに注意する。

$$h(\boldsymbol{x}) = \sum_{i=1}^{3} c(x_i) \pmod{10}$$

$$= \sum_{i=1}^{3} \left[c(x_i) \pmod{10} \right] \pmod{10}$$

$$= \left[c(x_1) \pmod{10} \right] + \left[c(x_2) \pmod{10} \right] + \left[c(x_3) \pmod{10} \right] \pmod{10}$$

各記号 $\alpha \in A$ に対して、$c(\alpha) \pmod{10}$ を求めておく。

a	b	c	d	e	f	g	h	i	j	k	l	m
7	8	9	0	1	2	3	4	5	6	7	8	9

n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	0	1	2

(mod 10)

この式より、各ハッシュ値は次のように求められる。

$h(abe) = 7 + 8 + 1 \pmod{10} = 6$

$h(ebi) = 1 + 8 + 5 \pmod{10} = 4$

$h(kan) = 7 + 7 + 0 \pmod{10} = 4$

$h(ito) = 5 + 6 + 1 \pmod{10} = 2$

$h(oki) = 1 + 7 + 5 \pmod{10} = 3$

（２）空の配列へ順に N の要素をすべて挿入するときの衝突回数の総和 $K = \sum_{x_i \in N} k_i$ を求めよ。

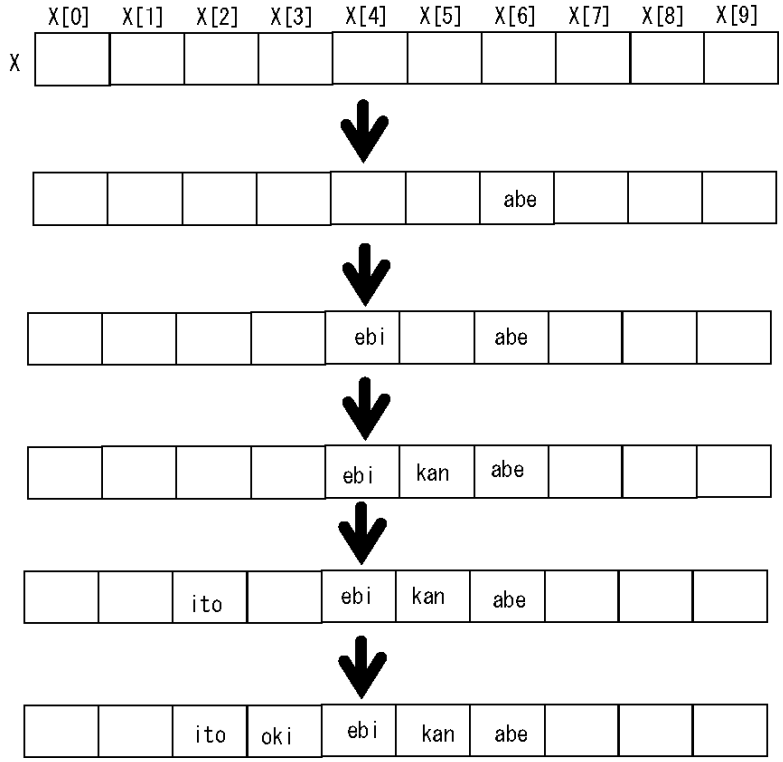
ここで、k_i は要素 x_i を挿入するときの衝突回数である。（５点）

同じ $h(\boldsymbol{x})$ の値を持つ要素が２つある。すなわち、$h(ebi) = h(kan) = 4$ である。一方、$h(\boldsymbol{x}) = 5$ となる要素が N に含まれないので、連鎖的衝突は起きない。したがって、$K = 1$ である。

（ebi と kan の中で、後から挿入したものだけが衝突する。）

（３）次の順序で名前を空の配列に挿入するとき、配列Ｘの内容の変化を図示せよ。（５点）

abe→ebi→kan→ito→oki



（4）ある順序で名前を空の配列に挿入したら、次の配列が得られた。このような格納状態になる挿入順序を全て示せ。（５点）

	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]
X			ito	oki	dai	kan	ebi	abe		

まず、dai のハッシュ値を求める。

$h(dai) = 0 + 7 + 5 \,(\mathrm{mod}\, 10) = 2$

次に、挿入順序を考察する。ハッシュ値の求め方より、後から挿入したものが後ろ（添え字の大きい方）に書き込まれる。したがって、同じハッシュ値を持つ要素について挿入順序が求められる。配列の状態から各２要素について挿入順序の前後が判断できる。すなわち、次の前後関係で挿入しているはずである。

$ito \rightarrow dai$

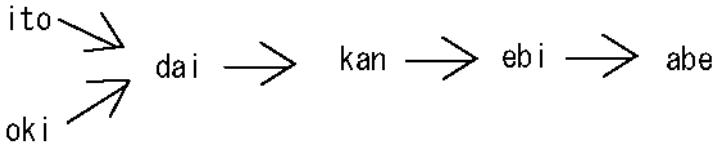
$oki \rightarrow dai$

$dai \rightarrow kan$

kan → *ebi*

ebi → *abe*

これより、次のような図（有向きグラフ、先行グラフ）を描ける。



このグラフより、挿入順序は次の２種類しか存在しないことがわかる。

ito → *oki* → *dai* → *kan* → *ebi* → *abe*

oki → *ito* → *dai* → *kan* → *ebi* → *abe*