2007 年度ソフトウェア工学再試験解答例

１．.漸近量評価（２２）

（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、各２点）

$(a)\quad T_a(n) = 2n^3 - 10n^2 + 7$

$O(n^3)$

$(b)\quad T_b(n) = (4n-1)(3n-3) + n(n+5)(n-3) + 8(n^2+5)(n-3)$

$O(n^3)$

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

$O(n^{1.5})$

$(d)\quad T_d(n) = n^2 + (\log n)^2 + 8^{\log n}$

$O(8^{\log n}) = O(n^3)$

$(e)\quad T_e(n) = \dfrac{7(n+5)(2n-1)}{\sqrt{n+3}} + \dfrac{2(n^2+5)(n-3)}{n-4} + \dfrac{(3n^3+1)(n-2)}{\sqrt{2n^3+5}}$

$O(n^{2.5})$

（２）次の C 言語風擬似コードの漸近計算量を O 記法で示せ。

（１２、各３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<10;i++){
        for(j=0;j<n;j++){
            (定数 c_a 時間の処理)
        }
    }
    return;
}
```

$O(n)$ 時間

(*b*) 漸近計算量$T_B(n)$

アルゴリズム B :

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

$O(n^2)$ 時間

(*c*) 漸近計算量$T_C(n)$

アルゴリズム C :

```
void algoC(int n){
    for(i=0;i<n;i=++){
       for(j=1;j<n;2*j){
            (定数 c_c 時間の処理)
        }
    }
    return;
}
```

$O(n \log n)$ 時間

(*d*) 漸近計算量$T_D(n)$

アルゴリズム D：

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

時間計算量$T_D(n)$は次の漸化式を満たす。

$$T_D(n)=\begin{cases}c_1 & n=0\\ 2T_D(n-1)+c_d & n>0\end{cases}$$

これを解く。

$$T_D(n)=2T_D(n-1)+c_d$$
$$=2\{2T_D(n-2)+c_d\}+c_d=2^2T_D(n-2)+2c_d+c_d$$
$$=2^2\{2T_D(n-3)+c_d\}+2c_d+c_d=2^3T_D(n-3)+2^2c_d+2c_d+c_d$$
$$\vdots$$
$$=2^iT_D(n-i)+c_d\sum_{j=0}^{i-1}2^j$$
$$\vdots$$
$$=2^nT_D(0)+c_d\sum_{j=0}^{n-1}2^j$$
$$=c_12^n+c_d(2^n-1)$$
$$=(c_1+c_d)2^n-c_d$$

よって、

$O(2^n)$ 時間

２．多項式の計算（２５点）

下のアルゴリズムは多項式 $f(x)=a_0+a_1x^1+\cdots+a_nx^n$ の関数値を計算するホーナーの方法である。ホーナーの方法では、

$$f(x)=a_0+a_1x^1+\cdots+a_nx^n=\left(a_0+\left(\cdots\left(a_{n-1}+\left(a_n\right)x\right)\cdots\right)x\right)$$

の順序で関数値を計算する。各係数 a_i は配列 $A[]$、最高次数 n は n として引数として与えられる。また、引数 x は $f(x)$ における x を表す。この方法について以下の問いに答えよ。

ホーナーの方法:

```
/*多項式を計算するホーナーの方法(繰り返し版)*/
1. double for_horner(double A[],int n,double x){
2. int i;/*ループカウンタ*/
3. double fx＝A[n]; /*関数値f(x)を表す*/
4.   for(i=1;i<n;i++){
5.     printf(“%f¥n”,fx);
6.     fx=A[n-i]+fx*x;
7.   }
8.   return fx;
9. }
```

(1) 関数 $f(x)=6+5x-7x^2+3x^3+5x^4-2x^5$ の $x=-2$ における関数値 $f(-2)$ を上のアルゴリズムで求めるとき、５行目の printf 文で表示される系列を示せ。（１０点）

初期 : -2

$i=1: 5+(-2)\times(-2)=9$

$i=2: 3+9\times(-2)=-15$

$i=3: -7+(-15)\times(-2)=23$

$i=4: 5+23\times(-2)=-41$

$i=5: 6+(-41)\times(-2)=88$

よって、 $-2, 9, -15, 23, -41, 88$

（験算）

$$f(-2)=6+5\times(-2)-7\times(-2)^2+3\times(-2)^3+5\times(-2)^4-2\times(-2)^5$$
$$=6-5\bullet2-7\bullet4-3\bullet8+5\bullet16+2\bullet32$$
$$=6-10-28-24+80+64$$
$$=150-62$$
$$=88$$

(2) 上の疑似コードの最悪時間計算量$T_h(n)$をO記法で示せ。（５点）

For ループが n 回繰り返されるだけである。よって、$T_h(n) = O(n)$時間

(3)ホーナーの方法を再帰的に実現するアルゴリズムを C 言語風の疑似コードにより示せ。ただし、以下のプロトタイプ宣言を持つとする。（１０点）

```
/*ホーナーの方法を実現する再帰的関数*/
double rec_horner(double A[],int n,double x);
```

```
/*多項式を計算するホーナーの方法（再帰的関数）
引数 n:内側から n 番目の括弧内の計算
引数 x:f(x)の x*/
1. double rec_horner(double A[],int n,double x){
2.   double fx;
3.   if(n==0){
4.     fx=A[N];
5.   }else{
6.     fx=A[N-n]+rec_horner(A,n-1,x)*x;
7.   }
8.   printf("%f¥n",fx);
9.   return fx;
10. }
```

３．マージソート(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。この配列 A をマージソートにより昇順にソートすることを考える。マージソートのアルゴリズムを下に擬似コードで示す。$l < m < r$ に対して、関数 merge は昇順の部分配列 $A[l] \cdots A[m]$ と昇順の部分配列 $A[m+1] \cdots A[r]$ から、部分配列 $A[l] \cdots A[r]$ を昇順にする。関数 print_array(l,r,A)は部分配列 $A[l] \cdots A[r]$ の内容を表示する。このとき、以下の問いに答えよ。

マージソート：

```
1. void m_sort(int l,int r,double A[N]){
2.   int m=(l+r)/2;/*中央の添え字*/
3.   if(l>=r){
4.     return;
5.   }else{
6.       /*print_array(l,r,A);*/
7.     m_sort( (ア) );/*前半のソート*/
8.     m_sort( (イ) );/*後半のソート*/
9.       /*print_array(l,r,A);*/
10.    merge(l,m,r,A);
11.      /*print_array(l,r,A);*/
12.    return;
13.  }
14.}
```

（１）マージソートが動作するように、(ア)、（イ）に入る適切な引数を示せ。(１０点、各５点)

(ア)l,m,A　(イ)m+1,r.A

(2) $n = 8$ とし、配列 A に下のように値が蓄えられているとする。m_sort(0,7,A)を実行したときに 6,9,11 行目の各 print_array(0,7,A)により表示される配列の内容を示せ。(５点)

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
A	79	28	60	41	57	11	85	73

6行目 :

79	28	60	41	57	11	85	73

9行目 :

28	41	60	79	11	57	73	85

11行目 :

11	28	41	57	60	73	79	85

(3)マージソートの時間計算量を$T_m(n)$と表す。このとき、$T_m(n)$が満たすべき漸化式を示せ。ただし、関数 merge の時間計算量は$O(r-l)=O(n)$時間であるとし、表示用関数 `print_array` の計算量は含めないとする。(５点)

c_1, c_2を定数として、アルゴリズムより、次の漸化式が成り立つ。

$$T_m(n) \le \begin{cases} c_1 & n=1\text{のとき} \\ 2T_m\left(\frac{n}{2}\right)+c_2 n & n>1\text{のとき} \end{cases}$$

(4)(3)の漸化式を解き、マージソートの最悪時間計算量$T_m(n)$を O 記法で示せ。ただし、$n=2^k$であるとして良い。(５点)

$n=2^k$より、$k=\log n$に関する漸化式をたてる。

$$T_m'(k) \le \begin{cases} c_1 & k=0\text{のとき} \\ 2T_m'(k-1)+c_2 2^k & k>0\text{のとき} \end{cases}$$

繰り返し代入することにより、漸化式を解く。

$$\begin{aligned}
&T_m'(k) \le 2T_m'(k-1)+c_2 2^k \\
&\le 2\left\{2T_m'(k-2)+c_2 2^{k-1}\right\}+c_2 2^k = 2^2 T_m'(k-2)+c_2\left(2\bullet 2^k\right) \\
&\le 2^2\left\{2T_m'(k-3)+c_2 2^{k-2}\right\}+c_2 2\bullet 2^k = 2^3 T_m'(k-3)+c_2\left(3\bullet 2^k\right) \\
&\vdots \\
&\le 2^i T_m'(k-i)+c_2\left(i\bullet 2^k\right) \\
&\vdots \\
&\le 2^k T_m'(0)+c_2\left(k\bullet 2^k\right) \\
&= c_1\left(2^k\right)+c_2\left(k\bullet 2^k\right)
\end{aligned}$$

よって、$T_m'(k)=O\left(k\bullet 2^k\right)$。$n=2^k$、$k=\log n$であるので、

$$T_m(n)=T_m'(k)=O\left(k\bullet 2^k\right)=O(n\log n)$$

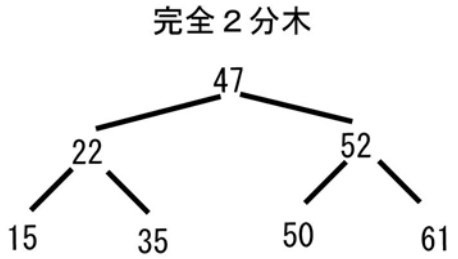
以上より、最悪時間計算量$T_m(n)=O(n\log n)$時間のアルゴリズムである。

４.２分探索木（２８点）

２分探索木Tに集合$S=\{15,22,35,47,50,52,61\}$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)で表わし、２分探索木Tからデータ x を削除する操作を delete(x)と表す。ただし、節点 x に子が２つあるとき、delete(x)では、削除される頂点の場所には､左の部分木中の最大の要素で置き換わる｡(このTは単なる２分探索木で AVL 木ではない｡）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Tで高さが最大となるもの図示せよ。(１種類示せばよい。)（２点）

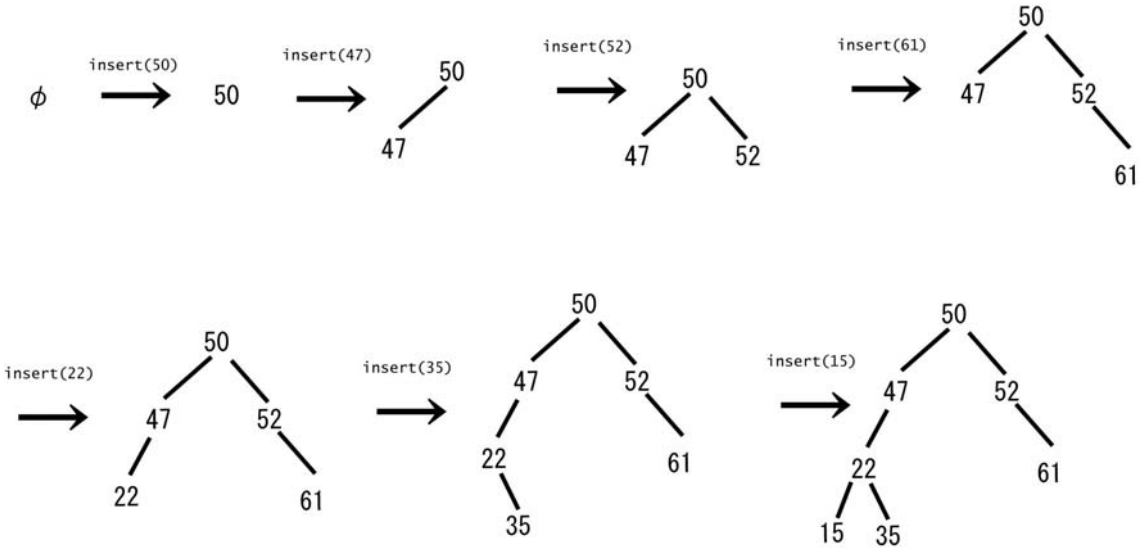
⑵集合Sを蓄える２分探索木Tで高さが最小となるもの図示せよ。(１種類示せばよい。)（２点）



⑶空の（要素数 0 の）２分探索木Tに次の順序で操作を行ったとき、２分探索木Tの状態の変化を図示せよ。（７点、各１点）

$\text{insert}(50) \rightarrow \text{insert}(47) \rightarrow \text{insert}(52) \rightarrow \text{insert}(61) \rightarrow \text{insert}(22) \rightarrow \text{insert}(35) \rightarrow \text{insert}(15)$

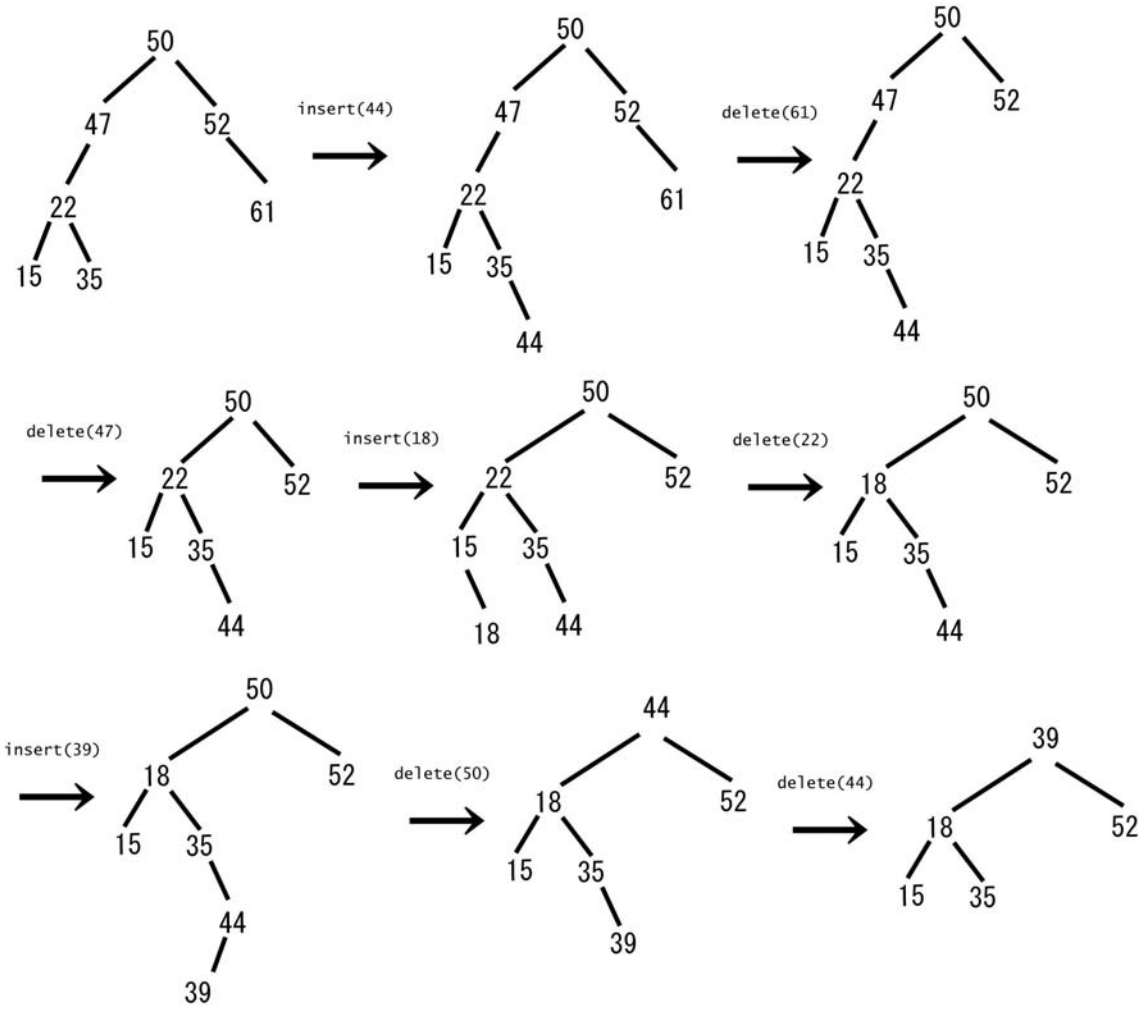
挿入による２分探索木の作成



(4)(3)の木にさらに次の順序で操作を行ったとき、２分探索木Tの状態の変化を図示せよ。
(13 点、各 insert　1 点、各 delete２点)

insert(44) → delete(61) → delete(47) → insert(18)
→ delete(22) → insert(39) → delete(50) → delete(44)

２分探索木への操作



(5) 空の２分探索木Tにn回を挿入を繰り返して２分探索木Tを作成するときの最悪時間計算量$T_{BT}(n)$を示せ。（４点）

Tに昇順にデータを挿入することを考える。

この場合$i,(1 \le i \le n)$番目のデータ挿入直前のTの高さは$i-1$であり、挿入直後のTの高さはiである。よって、i番目のデータの挿入に必要な時間計算量t_iは、定数cを用いて、$t_i = ci$と表せる。これらの総和が２分探索木作成に必要な時間計算量である。よって、

$$T_{BT}(n) = \sum_{i=1}^{n} t_i$$

$$= \sum_{i=1}^{n} ci$$

$$= c\sum_{i=1}^{n} i$$

$$= c(1+2+3+\cdots+n)$$

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

と計算できる。以上より、２分探索木作成にひつような最悪時間計算量は、$T_{BT}(n) = O(n^2)$ である。

なお、最悪時間計算量を実現する挿入例は次の図のようになる。

最悪の２分探索木構築