2008年度ソフトウェア工学試験解答例

1.漸近量評価（２２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。)

（各２点、１０点）

$(a)\quad T_a(n)=2n^3-5n^2+7n$

最高次数の項で、係数を１とした式である。

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

$(b)\quad T_b(n)=3(\sqrt[3]{n}+1)(\sqrt[3]{n}-2)(\sqrt[3]{n}-3)+7(\sqrt{n}+5)(\sqrt{n}-3)+8(n+5)$

第１項は、$\left(n^{\frac{1}{3}}\right)^3=n^1$ より、$O(n)$。

第２項は、$\left(n^{\frac{1}{2}}\right)^2=n^1$ より、$O(n)$。

以上より、

$T_b(n)=O(n)$

$(c)\quad T_c(n)=\left(\log n+\sqrt{n}\right)^3+\left(2^{\log n}+\sqrt{n}\right)^2$

第１項は、$O\left(\left(\log n+\sqrt{n}\right)^3\right)=O\left(\left(\sqrt{n}\right)^3\right)=O\left(n^{\frac{3}{2}}\right)$。

第２項は、$2^{\log n}=n$ より $O\left(\left(2^{\log n}+\sqrt{n}\right)^2\right)=O\left(n^2\right)$。

よって、$T_c(n)=O(n^2)$

$(d)\quad T_d(n)=8^{\log n}+n^2+\left(\log n\right)^3$

$8^{\log n}=\left(2^3\right)^{\log n}=2^{3\times\log n}=\left(2^{\log n}\right)^3=n^3$ より、$T_3(n)=O(n^3)$

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

各項の分子と分母の次数を考慮する。

第１項、分子$O\left(n^{\frac{5}{2}}\right)$、分母$O\left(n^{\frac{2}{2}}\right)=O\left(n^1\right)$より、$O\left(n^{\frac{3}{2}}\right)$。

第２項、分子$O\left(n^{\frac{3}{2}}\right)$、分母$O\left(n^{\frac{1}{3}}\right)$より、$O\left(n^{\frac{3}{2}-\frac{1}{3}}\right)=O\left(n^{\frac{9}{6}-\frac{2}{6}}\right)O\left(n^{\frac{7}{6}}\right)$。

第３項、分子$O\left(n^{\frac{3}{2}}\right)$、分母$O\left(n^{\frac{3}{2}}\right)$より、$O\left(n^{\frac{3}{2}-\frac{3}{2}}\right)=O\left(n^0\right)=O(1)$。

$0 < \frac{7}{6} < \frac{3}{2}$ なので、$T_e(n) = O\left(n^{\frac{3}{2}}\right)$。

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

内側の for ループは $O(n)$ 回繰り替えし、外側の for ループは $O(n)$ 回繰り替えす。
よって、$T_A(n) = O(n^2)$ の時間計算量である。

(b) 漸近計算量 $T_B(n)$
アルゴリズム B :

```
void algoB(int n){
    for(i=1;i<n;i=2*i){
        for(j=1;j<n;j=2*i){
            (定数 c_b 時間の処理)
        }
    }
    return;
}
```

for(j=1;j<n;j=2*j)の形の繰り返しについて考察する。
いま、l 回め繰りのときに、ループカウンタ j の値を考察する。

$l=0 \quad j=1=2^0$
$l=1 \quad j=1\times 2=2=2^1$
$l=2 \quad j=2\times 2=4=2^2$
$l=3 \quad j=4\times 2=8=2^3$
$\vdots \qquad \vdots$
$l=k \quad j=2^k$

よって、$j=2^k=n$ のときに終了するので、繰り返し回数 l は $l=k=\log n$ と求められる。
外側のループも同様に、$\log n$ 回の繰り返しである。

以上より、$T_B(n)=O\left((\log n)^2\right)$

(c) 漸近計算量 $T_C(n)$

アルゴリズム C：

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

アルゴリズムより、$T_C(n)$ は次の漸化式を満たす。

$$\begin{cases} T_C(1)=c & n=1 \\ T_C(n)=T(n-1)+c_c & n>1 \end{cases}$$

この漸化式を解く。

$T_C(n)=T(n-1)+c_c$
$=\left(T(n-2)+c_c\right)+c_c=T(n-2)+2c_c$
$=\left(T(n-3)+c_c\right)+2c_c=T(n-3)+3c_c$
$\vdots$
$=T(n-\left(n-1\right))+(n-1)c_c=T(1)+(n-1)c_c=c_cn+c-c_c$

以上より、$T_C(n)=O(n)$

(d) 漸近計算量 $T_D(n)$

アルゴリズム D：

```
void algoD(int n){
    if(n<=1){
        return;
    }else{
        algoD(n/2);
        for(i=0;i<n;i++){
            (定数 c_d 時間の処理)
         }
        algoD(n/2);
      return;
    }
}
```

アルゴリズムより、$T_D(n)$ は次の漸化式を満たす。

$$\begin{cases} T_D(1) = c & n = 1 \\ T_D(n) = T(\frac{n}{2}) + c_D \times n + T(\frac{n}{2}) & n > 1 \end{cases}$$

$$\therefore \begin{cases} T_D(1) = c & n = 1 \\ T_D(n) = 2T(\frac{n}{2}) + c_D n & n > 1 \end{cases}$$

よって、$n = 2^k$ より、次の k に関する関数 $T'_D(k)$ の漸化式を導く。

$$\begin{cases} T'_D(0) = c & k = 0 \\ T'_D(k) = 2T(k-1) + c_D 2^k & k > 0 \end{cases}$$

この漸化式を解く。

$$T'_D(k) \le 2T'(k-1) + c_D 2^k$$
$$\le 2\left(2T'(k-2) + c_D 2^{k-1}\right) + c_D 2^k = 2^2 T'(k-2) + 2 \times c_D 2^{k-1} + c_D 2^k = 2^2 T'(k-3) + 2c_D 2^k$$
$$\le 2^2\left(2T'(k-3) + c_D 2^{k-2}\right) + 2c_D 2^k = 2^3 T'(k-3) + 2^2 \times c_D 2^{k-2} + 2c_D 2^k = 2^3 T'(k-3) + 3c_D 2^k$$
$$\le 2^3\left(2T'(k-4) + c_D 2^{k-3}\right) + 3c_D 2^k = 2^4 T'(k-4) + 2^3 \times c_D 2^{k-3} + 3c_D 2^k = 2^4 T'(k-4) + 4c_D 2^k$$
$$\vdots$$
$$\le 2^k T'(0) + kc_D 2^k = c_D \times k2^k + c \times 2^k$$

よって、$T_D'(k) = O\left(k2^k\right)$

$2^k = n, k = \log n$ であるので、$T_D(n) = O\left(n \log n\right)$。

２．データ構造（ヒープ）

（２８点）

配列 A を利用してデータ構造のヒープを作成する。ヒープは次の条件を満たすように構築されているとする。

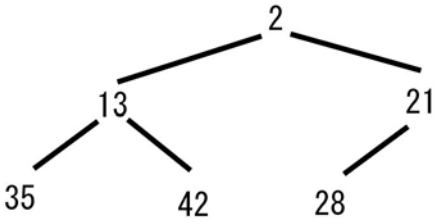
- $A[0]$ を根 r とし、根 $A[0]$ には最小値が蓄えられる。
- 根以外の各頂点 v に対して、親の方が小さい要素を蓄えているとする。（すなわち、頂点 v を根とする部分木中の最小の値が頂点 v に蓄えられる。）
- 頂点 v が $A[i]$ に蓄えられているとき、頂点 v の左の子は $A[2i+1]$ に、右の子は $A[2i+2]$ に蓄えられるとする。

このとき、次の設問に答えよ。

（１）配列が次の状態であるとき、ヒープの木の形状を図示せよ。（５点）

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]
A	2	13	21	35	42	28						

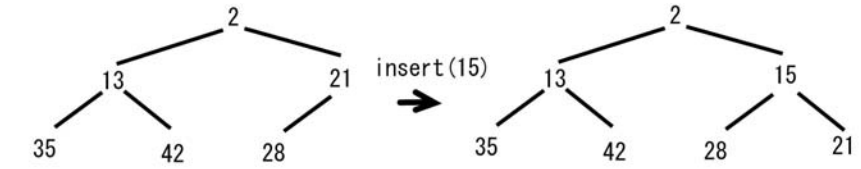
次のようになる。



（２）（１）の状態から次のように操作を行うとする。このとき、配列の状態とヒープの木をそれぞれ示せ。（16 点、各２点）

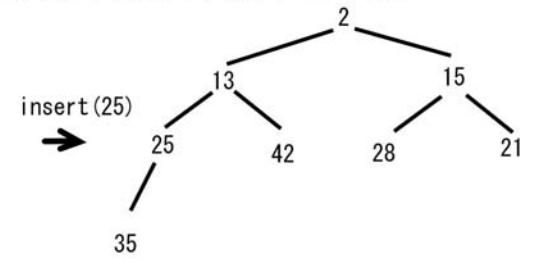
$$insert(15) \rightarrow insert(25) \rightarrow get\_\min() \rightarrow insert(30)$$
$$\rightarrow get\_\min() \rightarrow get\_\min() \rightarrow insert(3) \rightarrow get\_\min()$$

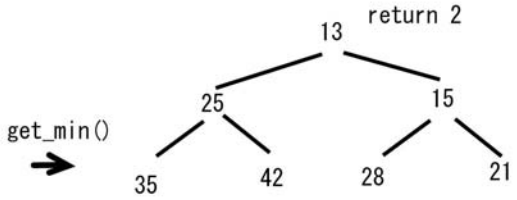
次のように変化する。



A：２｜１３｜２１｜３５｜４２｜２８

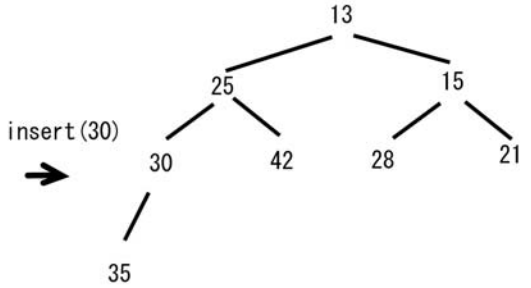
A：２｜１３｜１５｜３５｜４２｜２８｜２１

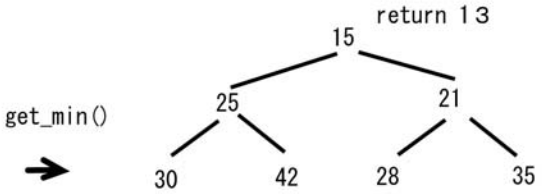




A：２｜１３｜１５｜２５｜４２｜２８｜２１｜３５

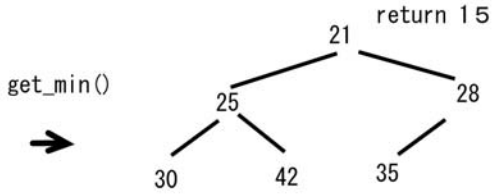
A：１３｜２５｜１５｜３５｜４２｜２８｜２１

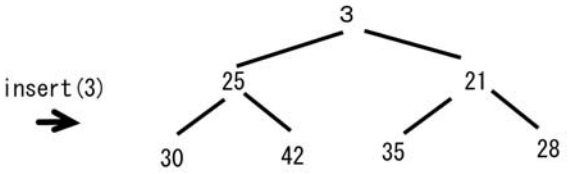




A：１３｜２５｜１５｜３０｜４２｜２８｜２１｜３５

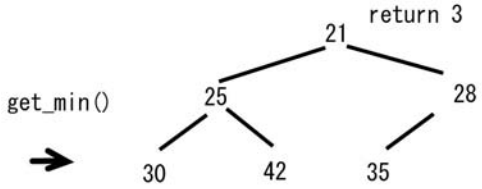
A：１５｜２５｜２１｜３０｜４２｜２８｜３５





A：２１｜２５｜２８｜３０｜４２｜３５

A：３｜２５｜２１｜３０｜４２｜３５｜２８



A：21｜25｜28｜30｜42｜35

（３）一般に n 個のデータが蓄えられているときに、操作１回に必要な最悪時間計算量を求 O 記法で示せ。ただし、結果だけでなく、理由も記述すること。（7 点）

最悪の場合、insert(x)も get_min()もヒープの高さ h に比例する時間計算量が必要になる。高さ h のヒープでの最大の要素数 $n_{\max}$ は、完全２分木状になっているときであり、次式が成り立つ。

$n_{\max} = 2^h - 1$

一方、高さhのヒープでの最小の要素数$n_{\min}$は、高さ$h-1$の完全２分木に１要素が加わったときであり、次式が成り立つ。

$n_{\min} = \left(2^{h-1} - 1\right) + 1 = 2^{h-1}$

よって、要素数nに関して、次式が成り立つ。

$n_{\min} \le n \le n_{\max}$

$\therefore 2^{h-1} \le n \le 2^h - 1 < 2^h$

$\therefore h - 1 \le \log n < h$

$\therefore \log n < h \le \log n + 1$

よって、各操作の最悪時間計算量は、$O(h) = O(\log n)$である。

３．選択ソート

配列Bにn個のデータが蓄えられている。このとき、最小値を求めるアルゴリズムを利用した選択ソートにより、配列Bを昇順にソートしよう。このようなアルゴリズムを下に示す。

```
int find_min(int left,int right){
    int j,min=left;
    for(j=left+1;j<=right;j++){
           if(B[min]>B[j]){
                  min=j;
           }
    }
    return min;
}
void select_sort(){
    int i,min;
    for(i=0;i<n-1;i++){
           min=find_min( （ア） );
           swap( （イ） );
    }
    return;
}
```

ここで、find_min は配列 B の $B[left]-B[right]$ 中の最小値を保持している配列要素 $B[j]$ の添え字 j を返す。（２５点）

(1)　選択ソートが適切に動作するように、(ア)、（イ）を適切に埋めよ。（１０点、各５点）

ア：find_min(i,n-1)

イ：swap(&B[i],&B[min])

(2) 1 回の find_min 呼び出しに必要な最悪時間計算量 $T_{\min}(n)$ を示せ。

f ind_min では、 $O(right-left)$ の時間計算量が必要である。一方、呼び出す際には、最悪の場合 $right=n-1, left=0$ として呼び出される。よって、

$$T_{\min}(n)=O\big((n-1)-(0)\big)=O(n)$$

が求める時間計算量である。

(3)アルゴリズム全体における交換回数 $N_{swap}(n)$ を示せ。なお、$N_{swap}(n)$ は O 記法ではなく、厳密な式で答えること。

s wap は select_sort の for 文で常に実行される。for 文は丁度 $n-1$ 回繰り返されるので、

$N_{swap}(n)=n-1$ である。

(4)アルゴリズム全体の最悪時間計算量 $T_{sort}(n)$ を O 記法で示せ。

Select_sort のおける for 文の繰り返しに応じて、find_min の計算時間が減少する。よって、次のように求められる。

$$T_{sort}(n)=O(1+2+3+\cdots+n-1)$$
$$=O\left(\frac{n(n-1)}{2}\right)$$
$$=O(n^2)$$

４．２分探索（２５点）

配列Cに次のようなデータが蓄えられているとする。このとき、下に示すプログラムで２分探索を行おうとしている。

	C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]
C	3	5	13	22	41	63	73	85

```
int binary_search(int key,int left,int right){
  int mid;/*中央の添え字*/
  if(left>right){return -1;/*存在しない*/}
  else{
    mid=(left+right)/2;
    if(C[mid]==key){
       return mid;
    }else if(key < C[mid]){
         return binary_search( （ア）  );/*小さい方*/
    }else{
         return binary_search( （イ）  );/*大きい方*/
  }
}
```

(1) ２分探索が適切に動作するよういに、(ア)、（イ）に入る適切な引数を示せ。
（１０点、各５点）

ア：binary_search(key,left,mid-1);

イ：binary_search(key,mid+1,right);

(2) 次の引数でこの関数を呼び出したとき、調べられる配列の要素を順に示せ。
（１０点、各５点）

(a) `binary_search(13,0,7);`

再帰の深さkとし、そのときの left,right,mid をそれぞれ、$left_k, right_k, mid_k$とする。これらは次のように計算される。

$k=1 \quad left_1=0 \qquad right_1=7 \qquad mid_1=\left\lfloor \frac{left_1+right_1}{2} \right\rfloor=\left\lfloor \frac{0+7}{2} \right\rfloor=3$

$k=2 \quad left_2=left_1=0 \qquad right_2=mid_1-1=2 \quad mid_2=\left\lfloor \frac{left_2+right_2}{2} \right\rfloor=\left\lfloor \frac{0+2}{2} \right\rfloor=1$

$k=3 \quad left_3=mid_2+1=2 \quad right_3=right_2=2 \qquad mid_3=\left\lfloor \frac{left_3+right_3}{2} \right\rfloor=\left\lfloor \frac{2+2}{2} \right\rfloor=2$

よって、

$C[3]=22 \rightarrow C[1]=55 \rightarrow C[2]=13: return \quad 2$

という系列で探索される。

(b) `binary_search(77,0,7);`

$left_k, right_k, mid_k$ は次のように計算される。

$k=1 \quad left_1=0 \qquad right_1=7 \qquad mid_1=\left\lfloor \frac{left_1+right_1}{2} \right\rfloor=\left\lfloor \frac{0+7}{2} \right\rfloor=3$

$k=2 \quad left_2=mid_1+1=4 \quad right_2=right_1=7 \quad mid_2=\left\lfloor \frac{left_2+right_2}{2} \right\rfloor=\left\lfloor \frac{4+7}{2} \right\rfloor=5$

$k=3 \quad left_3=mid_2+1=6 \quad right_3=right_2=7 \quad mid_3=\left\lfloor \frac{left_3+right_3}{2} \right\rfloor=\left\lfloor \frac{6+7}{2} \right\rfloor=6$

$k=4 \quad left_4=mid_3+1=7 \quad right_4=right_3=7 \quad mid_4=\left\lfloor \frac{left_4+right_4}{2} \right\rfloor=\left\lfloor \frac{7+7}{2} \right\rfloor=7$

よって、

$C[3]=22 \rightarrow C[5]=63 \rightarrow C[6]=73 \rightarrow C[7]=85:$ 失敗

という系列で探索される。

(3)データ数が一般の n であるとき、上のプログラムの時間計算量 $T(n)$ が満たすべき漸化式を示し、最悪時間計算量 $T(n)$ を O 記法で答えよ。（５点）

（解法１）漸化式を立てて、解く方法。

時間計算量を $T(n)$ とする。

このとき、次の漸化式を満たす。（c_1, c_2 はある定数。）

$$T(n) \leq \begin{cases} c_1 & n \leq 1 \\ T(\frac{n}{2}) + c_2 & n \geq 2 \end{cases}$$

ここで、まず$n = 2^k$と表せるとする。
このとき、

$$T'(k) \leq \begin{cases} c_1 & k = 0 \\ T'(k-1) + c_2 & k \geq 1 \end{cases}$$

と表せる。この漸化式を解く。

$$T'(k) \leq T'(k-1) + c_2$$
$$\leq \left(T'(k-2) + c_2\right) + c_2 = T'(k-2) + 2c_2$$
$$\leq \left(T'(k-3) + c_2\right) + 2c_2 = T'(k-3) + 3c_2$$
$$\vdots$$
$$\leq T'(1) + (k-1)c_2$$
$$= c_1 + (k-1)c_2$$

よって、$T'(k) = O(k)$
また、$k = O(\log n)$より、$T(n) = O(\log n)$
　次に、一般のnについて考察する。$2^{l-1} \leq n < 2^l$なるlが存在する。このとき、上述のように、次式が成り立つ。

$$T(n) < T(2^l) = O(l)$$
$$\therefore T(n) \leq cl$$

一方、

$$l - 1 \leq \log n < l$$
$$\therefore \log n < l \leq \log n + 1$$

に注意すると次のように計算できる。

$$T(n) \leq cl$$
$$\leq c\left(\log n + 1\right) = c\log n + c$$
$$\therefore T(n) = O(\log n)$$

以上より、２分探索のアルゴリズムの最悪時間計算量は、
$O(\log n)$である。
（解法２）各ステップで探索できる要素を考えて、配列すべての要素を網羅する方法。
（配布資料では、この方法が記述されている。）
深さjの再帰までで探索が可能な要素数の最大値は、次式で表される。

$$\sum_{i=0}^{j} 2^i = 1 + 2 + 4 + \cdots + 2^j = \frac{2(2^j - 1)}{2 - 1} = 2^{j+1} - 1$$

最悪の場合、深さ$j-1$までの再帰でキーが発見されず、深さjでキーが発見される。また、

全ての要素を探索する必要がある。したがって、再帰の深さの最大値に関して次式が成り立つ。

$2^j - 1 < n \leq 2^{j+1} - 1$

$\therefore j < \log_2(n+1) \leq j+1$

$\therefore j+1 = \lceil \log_2(n+1) \rceil$

$\therefore j = O(\log_2 n)$