

2009 年度ソフトウェア工学試験問題

日時：2009 年 7 月 31 日(金)12:50-14:20 場所：K205

注意事項：

指定された席に着席すること。

答案は机に残して退室すること。

問題用紙は持ち帰ること。

試験用紙、計算用紙が必要なときには申し出ること。

解の簡単な導出過程も書くこと。

1.漸近量評価

(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。)

(a) $T_a(n) = \frac{2}{3}n^5 - \frac{7}{2}n^3 + \frac{1}{5}n$

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

(c) $T_c(n) = (\log n)^5 + (\sqrt{n})^3$

(d) $T_d(n) = 8^{\log n} + n^2 + (\sqrt{n})^5$

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

(2) 次の C 言語風擬似コードの時間計算量を O 記法で示せ。

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。

(d) 時間計算量 $T_d(n)$

アルゴリズム D:

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

(e) 時間計算量 $T_e(n)$

アルゴリズム E:

```
void algoE(int n){
    for(i=1;i<n;i=2*i){
        (定数  $c_e$  時間の処理)
    }
    return;
}
```

(f) 漸近計算量 $T_f(n)$

アルゴリズム F :

```
void algoF(int n){
    if(n<=1){
        return;
    }else{
        (定数  $c_f$  時間の処理)
        algoF(n/2);
        algoF(n/2);
        return;
    }
}
```

2 . べき乗のアルゴリズム

次のC言語の関数 `pow(int n,double x)` は、 $n = 2^k$ に対して、 $f(n,x) = x^n = \prod_{i=1}^n x$ を計算する。

ここで、 n は n に、 x は x に対応する。

```
double pow(int n,double x){
    double f; /*べき乗の関数値*/
    int i;
    f= ( ア ) /*初期化*/
    for(i=0;i<n;i++){
        f=f*x;
    }
    return f;
}
```

このとき、次の設問に答えよ。

(1) (ア) を適切に埋めよ。

(2) このプログラムの時間計算量 $T_{slow}(n)$ を求めよ。

(3) $f(n,x) = x^n = \prod_{i=1}^n x$ を計算するより高速なアルゴリズム (関数)

`double fast_pow(int n,double x)` を示せ。(ここで、入力サイズは次数 n とする。)

(4) (3)のアルゴリズムの時間計算量 $T_{fast}(n)$ を求めよ。

3 . クイックソート

配列 A に次のようにデータが蓄えられているとする。

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
A	12	4	5	11	8	2	15	7

この配列をソートするために以下に示すようなクイックソートを用いる。

```
void qsort(int left, int right){
    int pos ; /*分割位置*/
    if(left >= right){
        return;
    }else{
        pos=partition(left,right);
        print_array(left,right);
        qsort(left,pos-1);
        qsort(pos+1,right);
    }
}
```

ここで、partition はピボット (基準値) の位置を求める関数で、部分配列 A[left]-A[right] をピボットより小さい要素、ピボット、ピボットより大きい要素の順に並べかえて、ピボット位置 (配列の添え字) を返す関数である。なお、ピボットは並び替え前における部分配列の右端 A[right] が選ばれるとする。また、print_array(left,right) は、部分配列 A[left]-A[right] を表示する関数である。

このとき、クイックソートにおける動作について答えよ。

- (1) print_array によって表示される配列の内容をすべて示せ。
- (2) $n = \text{right} - \text{left}$ とし、partition の時間計算量を $T_p(n)$ 、クイックソートの時間計算量を $T_q(n)$ とする。このとき $T_q(n)$ が成り立つべき漸化式を以下の形式で示せ。ただし、表示部分の print_array は $T_q(n)$ の時間計算量に含めないようにせよ。

$$T_q(n) = \begin{cases} c(\text{定数}) & n \leq 0 \\ (\text{right, left, pos を用いた再帰式}) & n > 0 \end{cases}$$

- (3) クイックソートの最悪時間計算量 $T_q^{\text{worst}}(n)$ を O 記法で示せ。
- (4) データ数 $n=8$ において時間計算量が最大となる入力例を一つ示せ。

4 . 線形探索

次のC言語の関数 `linear_search(double key,int n,double A[])` は、 n 個の要素の配列 A 中に `key` があるか探索を行う。配列中に `key` が無い場合には、`-1` を返し、配列中に `key` が存在する場合にはその添え字を返す。このとき、以下の設問に答えよ。

```
int linear_search(double key,int n,double A[]){
    int i=0;
    A[n]=( ア );    /* 番兵の設定      */
    while(key!=A[i]){
        i++;
    }
    if(i==n){
        return -1; /*key は配列 A に存在しない*/
    }else{
        return ( イ ); /*key は配列 A に存在する*/
    }
}
```

- (1) (ア)(イ)を適切に埋めよ。
- (2) このプログラムで最も比較回数が大きくなる場合を答えよ。
- (3) (2)の場合での比較回数 $H_{\text{worst}}(n)$ を求めよ。ただし、 $H_{\text{worst}}(n)$ は O 記法ではなくて、厳密な関数の形で答える事。
- (4) 配列 A 中に `key` が必ず含まれ、各 $A[i]$ に `key` が存在する確率 $P(i)$ が $P(i) = \frac{1}{n}$ と表せるとする。このときの平均比較回数 $H_{\text{ave}}(n)$ を求めよ。ただし、 $H_{\text{ave}}(n)$ は O 記法ではなくて、厳密な関数の形で答える事。