平成 21 年度

東京大学大学院情報理工学系研究科

コンピュータ科学専攻

入学試験問題

専門科目 I

解答

東京大学理学部情報科学科 2009

2012 年 8 月 7 日

問題 1

(1)

```
#include <stddef.h>

struct bintree {
        int val;
        struct bintree *left, *right;
};

int sum(const struct bintree *bt)
{
        return bt->val + (bt->right != NULL? sum(bt->right): 0) + (bt->left != NULL? sum(bt->left):
            0);
}
```

(2)

```
#include <stddef.h>
#include <stdlib.h>

#define STACK_SIZE (4 * 1024)

struct bintree {
        int val;
        struct bintree *left, *right;
};

static void push(const void ***stack, const void *data)
{
        *((*stack)++) = data;
}

static const void *pop(const void ***stack)
{
        return *(--(*stack));
}

int sum(const struct bintree *bt)
{
        const void **stack = malloc(sizeof(void *) * STACK_SIZE); // XXX should check NULL
        int s = 0;
        const struct bintree *b;
        push(&stack, NULL);
        push(&stack, bt);
        while ((b = pop(&stack)) != NULL) {
                if (b->left != NULL) {
                        push(&stack, b->left);
                }
                if (b->right != NULL) {
                        push(&stack, b->right);
                }
                s += b->val;
        }
        free(stack);
        return s;
}
```

(3)

bintree の left ポインタを、自分の 1 つ上の要素を指すように書き変えながら探索すればよい。

```
#include <stddef.h>

struct bintree {
```

```
4         int val;
5         struct bintree *left, *right;
6 };
7
8 enum {
9         FROM_TOP,
10        FROM_RIGHT,
11        FROM_LEFT,
12 };
13
14 int sum(struct bintree *bt)
15 {
16        int f = FROM_TOP;
17        struct bintree *b = bt;
18        struct bintree *up = NULL;
19        int s = 0;
20        while (b != NULL) {
21                switch (f) {
22                case FROM_TOP:
23                        if (b->left != NULL) {
24                                struct bintree *p = b->left;
25                                b->left = up;
26                                up = b;
27                                b = p;
28                                f = FROM_TOP;
29                                break;
30                        } else {
31                                b->left = up;
32                        }
33                        /* FALL THROUGH */
34                case FROM_LEFT:
35                        if (b->right != NULL) {
36                                up = b;
37                                b = b->right;
38                                f = FROM_TOP;
39                                break;
40                        }
41                        /* FALL THROUGH */
42                case FROM_RIGHT:
43                        s += b->val;
44                        struct bintree *p = b->left;
45                        if (p != NULL) {
46                                if (p->right == b) {
47                                        f = FROM_RIGHT;
48                                } else {
49                                        f = FROM_LEFT;
50                                }
51                        }
52                        b = p;
53                        break;
54                }
55        }
56        return s;
57 }
```

問題 2

(1)

かきゃわかる

(2)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d & 0 & 0 & 1 \end{pmatrix}$$

(3)

$$A_L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{d}{R} & 0 & 0 & -\frac{1}{R} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{d}{R} & 0 & 0 & -\frac{1}{R} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$PL = (x + \frac{dz}{R}, y, 0, 1 - \frac{z}{R})$$

$$PR = (x - \frac{dz}{R}, y, 0, 1 - \frac{z}{R})$$

(4)

PR を用いて各点

$$(0, D/2, R/2, 1)$$

$$(0, -D/2, R/2, 1)$$

$$(0, D/2, -R/2, 1)$$

$$(0, -D/2, -R/2, 1)$$

を変換↓

$$(-d/2, D/2, 0, 1/2)$$

$$(-d/2, -D/2, 0, 1/2)$$

$$(d/2, D/2, 0, 3/2)$$

$$(d/2, -D/2, 0, 3/2)$$

どうみても台形ですwww

(5)

$t = 0$ の時点での視差 Q0

$$PL0 = (2xs - d, 2ys, 0)$$

$$PR0 = (2xs + d, 2ys, 0)$$

$$Q0 = 2d$$

$t = 1$ での視差 Q1

$$PL1 = (2xe - d, 2ye, 0)$$

$$PR1 = (2xe + d, 2ye, 0)$$

$$Q1 = 2d$$

$t = 2$ での視差 Q2

$$PL2 = (2xe/3 + d/3, ye, 0)$$

$$PR2 = (2xe/3 - d/3, ye, 0)$$

$$Q2 = -2d/3$$

時刻 t での $P(s = t - 1)$

$$(xe, ye, R/2 - s)$$

このときの視差 Qt を求める

$$PLt = (2xe/(1 + 2s) + d(1 - 2s)/(1 + 2s), ye/w', 0)$$

$$PRt = (2xe/(1 + 2s) - d(1 - 2s)/(1 + 2s), ye/w', 0)$$

$$Qt = -2d(3 - 2t)/(2t - 1)$$

よって

$$0 < t < 1$$

$$Q = d$$

$$1 < t < 2$$

$$Q = -2d(3 - 2t)/(2t - 1)$$

問題 3

(1)

開始	終了	プロセス	状態
0	50	P_1	残 30
50	100	P_2	残 120
100	150	P_3	残 150
150	180	P_1	P_1 終了
180	230	P_2	残 70
230	280	P_3	残 100
280	330	P_2	残 20
330	380	P_3	残 50
380	400	P_2	P_2 終了
400	450	P_3	P_3 終了

上表よりターンアラウンドタイムは (180,400,450)。

(2)

プロセス P_i が終了したとき、プロセス $P_j(j \leq i)$ は終了しており、プロセス $P_j(j > i)$ は それぞれ 時間 Q で $(\lceil \frac{C_i}{Q} \rceil - 1)$ 回 実行されてるので

$$T_i = \sum_{k=1}^{i} C_k + Q(N-i)(\lceil \frac{C_i}{Q} \rceil - 1)$$

とくに

$$T_1 = C_1 + Q(N-1)(\lceil \frac{C_1}{Q} \rceil - 1)$$

$$T_2 = C_1 + C_2 + Q(N-2)(\lceil \frac{C_2}{Q} \rceil - 1)$$

であるが、T_2 を T_1 を用いて簡潔に表す方法はよくわからない (もちろん無理矢理なら書けるが)。

問題 4

(1)

$$\begin{pmatrix} 0 & 0 & 2 \\ 5 & 0 & 2 \\ 5 & 4 & 0 \end{pmatrix}$$

解説

計算するだけ（あってるかなあ）

(2)

元のグラフ G に，各頂点に重み 0 の自己ループを加えたグラフを G' とする．

D^l の i, j 要素は，頂点 i から頂点 j への辺を l 本含む G' 上のパスのうち，含まれる辺の重みの最大値が最小のパスの，含まれる辺の重みの最大値．

解説

対角要素が 0 なのが微妙にトリッキー．もう少し良い扱い方があるかもしれないです．

つまり，頂点 i から頂点 j へ，通った辺の重みの最大値ができるだけ小さくなるように行きたい．このとき，l 回動くのだったら，D^l の i, j 要素の値の重み以下の辺だけで行けるよ，ということ．

2 乗がどういう意味を持っているかを考えてみると，自然と導けると思います．

(3)

辺を n 本含む閉路であり，含まれる辺の重みが全て負のものがグラフに存在する．

解説

(2) からすぐわかる．

「ハミルトン閉路問題が帰着できるじゃん．辺があるところだけ負の重みなグラフを作って (4) のアルゴリズム使えばいいじゃん．P=NP キタ！！」と思った人が居るかもしれませんが，冷静になりましょう．違います．

(4)

解答例 1

繰り返し二乗法を用いて計算する．

まず，この代数系でも，行列の積は $O(n^3)$ で計算可能．繰り返し二乗法を用いれば，$O(\log n)$ 回の積の計算で n - 1 乗が計算可能．

以上より，このアルゴリズムは $O(n^3 \log n)$ で動作する．

解説 1

素直に計算してるだけなので，正当性が自明．しかし，非負性や対称性を使っていない．

繰り返し二乗法は，n 乗を求めるには $\frac{n}{2}$ 乗を求めて二乗すればよく，$\frac{n}{2}$ 乗を求めるには $\frac{n}{4}$ 乗を求めて二乗すればよく・・・というイメージで $O(\log n)$ 回積を計算すればよい，という感じのもの．良く知られているので，調べてください．愚直に n 回掛け算する $O(n^4)$ のアルゴリズムはさすがにあまり点がもらえないと思う．

「できるだけオーダーを小さく」と言われているからって，行列乗算に Strassen とか知ったかぶって書かないこと．この代数系では引き算ができないので，使えません．

解答例 2

ワーシャルフロイドのアルゴリズムを応用する．

行列 D が非負の時，D^{n-1} の i,j 要素は，元のグラフ G 上の頂点 i から頂点 j へのパスのうち，含まれる辺の重みの最大値が最小のパスの，含まれる辺の重みの最大値である．

これは，そのようなパスを考える際，辺を n 以上含むものは同じ頂点を 2 度以上訪れているため，考える必要がなく，辺を $n-1$ 未満含むものは，G' 上では，重み 0 の自己ループの存在により，含まれる辺の重みの最大値が等しく辺を $n-1$ 個含むパスが存在するためである．（行列 D が非負より，重み 0 の自己ループを何度通っても含まれる辺の重みの最大値は変化しない．）

また，これはパスの重みを含まれる辺の重みの最大値とした最短路に他ならない．よって，ワーシャルフロイドのアルゴリズムを変形し，以下のようにすればよい．

for $k = 1..n$ **do**
  for $i = 1..n$ **do**
    for $j = 1..n$ **do**
      $D_{i,j} = \min\{D_{i,j}, \max\{D_{i,k}, D_{k,j}\}\}$
    end for
  end for
end for

ワーシャルフロイドのアルゴリズムと同様に，このアルゴリズムも $O(n^3)$ で動作する．

解説 2

コストが変化した最短路みたいなもので，全点対だし，ワーシャルフロイドを使えばよい．非負性を使っている．対称性はなくても動作すると思う．

解答例 3

Kruskal と類似したアルゴリズムを用いる．

(D^{n-1} に意味合いついての説明は解答例 2 と同様なので省略)

また，行列の対称性より，グラフを無向グラフと考えることができる．

よって，頂点集合が G の頂点集合と等しく，辺を持たないグラフをまず用意し，そこに G の辺を重みの小さい辺から順に加えて行き，重み w の辺を加えて頂点 i,j が新たに連結になったとき，$D^{n-1}_{i,j} = D^{n-1}_{j,i} = w$

と分かる．

連結成分を管理するため，n 要素の配列 c を用意し，頂点 i に対し，c_i の値で連結成分を区別する．はじめ $c_i := i$ としておく．頂点 i に対し，辺 (i, j) を追加する際，$c_i = c_j$ であれば，すでに連結であり何もしなくて良く，$c_i \neq c_j$ であれば，(1) $c_x = c_i, c_y = c_j$ なる任意の (x, y) に関して $D^{n-1}_{x,y} = D^{n-1}_{y,x} = w$ と分かり，(2) $c_y = c_j$ なる任意の y について $c_y := c_i$ とする．

(1) の処理は全体で $O(n^2)$ 時間，(2) の処理は一度につき $O(n)$ であり $n - 1$ 回実行されるためやはり $O(n^2)$ 時間である．よって，このアルゴリズムは，辺のソートが実行時間の主要項となり，$O(n^2 \log n)$ 時間で動作する．

解説 3

辺がない状態から，重みの小さい辺からぽいぽい追加していくと，途中で今行き来できるやつらは，その重み以下のパスでつながれてるね，ということ．

非負性も対称性も使っている．

専門科目 II

問題 1

(1)

どこまで丁寧に示せばいいのか分かりません＞＜

全ての状態が R の中に書かれているから (A) と (A) の入れ替えは満たされる.$(s_{13}, s_{23}), (s_{13}, s_{24})$ が R に含まれているので (D) と (D) の入れ替えも満たされる. (B) は大変なので 1 つだけ確認すると,s_{11} が a で s_{12} に遷移することができるが,s_{11} と関係がある s_{21} は a で s_{22} に遷移することができ, $s_{12}Rs_{22}$ が成り立っている.

(2)

以下に示す 2 つの非決定性有限オートマトン A_3,A_4 は同一の言語を受理するが, 等価ではない.

$A_3 = ((s_{31}, s_{32}), \Sigma, ((s_{31}, a, s_{31}), (s_{31}, a, s_{32})), (s_{31}), (s_{32}))$

$A_4 = ((s_{41}, s_{42}), \Sigma, ((s_{41}, a, s_{42}), (s_{42}, a, s_{42})), (s_{41}), (s_{42}))$

これらのオートマトンは明らかに言語 a^+ を受理する. ところが条件 (A),(B),(C) を同時に満たすことはできない. よってこれらのオートマトンは等価ではない.

問題 2

(1)

$N_{2,2} = 13, N_{2,3} = 25$

(2)

1 文字目の組み合わせは $(A, A), (A, B), (B, A)$ の 3 通りで、各々の場合の数を足せばよい。

$N_{n,m} = N_{n-1,m} + N_{n,m-1} + N_{n-1,m-1}$

(3)

アプローチ 1

$N_{0,n} = N_{n,0} = 1$ と (2) の結果を用いて、2 項間漸化式を構成して解く。

$N_{1,n} = N_{1,n-1} + N_{0,n} + N_{0,n-1} = N_{1,n-1} + 2$ を解いて $N_{1,N} = 2n + 1$

$N_{2,n} = N_{2,n-1} + N_{1,n} + N_{1,n-1} = N_{2,n-1} + 2n + 1$ を解いて $N_{2,n} = 2n^2 + 2n + 1$

アプローチ 2

(2) の結果に惑わされず、(1) での計算を数式にする。アプローチ 1 では、おそらく (4) で詰む。

$N_{m,n}$ を計算するとき、文字列長 l は $m \leq l \leq n + m$ の範囲にあり (正確な下限は $max(m, n)$ であり、以下 $m \geq n$ でなくてもよいことに注意)、$l = m + k$ のときの場合の数は、q に含まれる n 個の A の配置の場合の数 ${}_{m+k}C_n$ と その A に対して p に含まれる B を当てる場合の数 ${}_nC_k$ の積である。

よって、その和を取ることで $N_{m,n}$ は以下の式で表される。

$N_{m,n} = \sum_{k=0}^{n} {}_{m+k}C_n {}_nC_k$

m=1,m=2 を代入して、答えを得る。

$N_{1,n} = {}_{n+1}C_1 + {}_nC_1 = 2n + 1$

$N_{2,n} = {}_{n+2}C_2 + 2_{n+1}C_2 + {}_nC_2 = 2n^2 + 2n + 1$

(4)

(3) のアプローチ 2 で得た式を使って適当に上からおさえる。

$N_{m,n} = \sum_{k=0}^{n} {}_{m+k}C_n {}_nC_k \leq {}_{m+n}C_n \sum_{k=0}^{n} {}_nC_k = {}_{m+n}C_n 2^n = \frac{(m+n)!}{m!}\frac{2^n}{n!} \leq \frac{(m+n)^n 2^n}{n!}$ ■

問題 3

(1)

$Ax_0 = \sum_{i=1}^{n} \xi_i \lambda_i u_i$

(2)

$x_k = C(\sum_{i=1}^{n} \xi_i \lambda_i^k u_i) = C'(\sum_{i=1}^{n} \xi_i (\frac{\lambda_i}{\lambda_1})^k u_i)$ (C, C' は正規化のための係数)
$i > 1$ について $\lim_{k\to\infty}(\frac{\lambda_i}{\lambda_1})^k = 0$ であるので、x_k は u_1 と平行になる。■

(3)

誤差がない場合
$x_k = C(\sum_{i=2}^{n} \xi_i \lambda_i^k u_i) = C'(\sum_{i=2}^{n} \xi_i (\frac{\lambda_i}{\lambda_2})^k u_i)$ (C, C' は正規化のための係数)
(2) と同様の議論から、x_k は u_2 と平行になる。
丸め誤差がある場合
計算機上では x_k を正確に表現できないので、丸めた値 x'_k を用いる。一般に x'_k は $u_i (i = 2, 3, \cdots, n)$ の線形結合で表すことはできず、固有ベクトルの線形和で表すと u_1 の係数が非零になる。そのため、x_k は u_1 と平行になる。

(4)

$A_1 u_1 = 0$ は明らか。$A_1 u_i = \lambda_i u_i - \lambda_1 u_1 u_1^T u_i$ であるので、適当に u_1 の定数倍を足して $u_i + Cu_1$ が固有ベクトルになるように調整する。
固有値 0 固有ベクトル u_1 , 固有値 $\lambda_i (i = 2, 3, \cdots, n)$ 固有ベクトル $u_i - \frac{\lambda_1}{\lambda_i}(u_1^T u_i)u_1$

(5)

y_k を $y_k = \frac{1}{||A_1 y_{k-1}||} A_1 y_{k-1}$ で定義し、y_k が収束する十分大きい k まで計算を行う。
A_1 の最大の固有値は λ_2 であるため、(2) の結論より 十分大きい k について y_k は $u_2 - \frac{\lambda_1}{\lambda_2}(u_1^T u_2)u_1$ と平行になる。$A_1 y_k = \lambda_2 y_k$ となるので、λ_2 の近似値が得られる。$y_k = au_1 + bu_2$ とすると $Ay_k = a\lambda_1 u_1 + b\lambda_2 u_2$ であり、λ_1 が既知であることにより $\lambda_1 y_k - Ay_k$ で u_2 と平行なベクトルが得られ、正規化することで u_2 が得られる。
(3) の方法では丸め誤差があると u_2 が求められないが、(5) の方法であれば、丸め誤差が存在しても問題なく求まる。

問題 4

(1)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity m1 is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           ld : out  STD_LOGIC);
end m1;

architecture Behavioral of m1 is
        component DFF
        port(D,CLK : in std_logic;
                Q: out std_logic);
        end component;

        signal w0,w1,w2,w3,w4 : std_logic;
begin
        DFF0 :DFF port map(input,clk,w0);
        DFF1 :DFF port map(w0,clk,w1);
        DFF2 :DFF port map(w1,clk,w2);
        DFF3 :DFF port map(w2,clk,w3);
        DFF4 :DFF port map(w3,clk,w4);

        ld <= (not w0) and w1 and w2 and w3 and w4;
end Behavioral;
```

(2)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity m2 is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           ld,sd,ls,ss : out  STD_LOGIC);
end m2;

architecture Behavioral of m2 is
        component DFF
        port(D,CLK : in std_logic;
                Q: out std_logic);
        end component;

        signal w0,w1,w2,w3 : std_logic := '0';
begin
        DFF0 :DFF port map(input,clk,w0);
        DFF1 :DFF port map(w0,clk,w1);
        DFF2 :DFF port map(w1,clk,w2);
        DFF3 :DFF port map(w2,clk,w3);
        DFF4 :DFF port map(w3,clk,w4);

        ld <= (not w0) and w1 and w2 and w3 and w4;
        sd <= (not (w0 and w1)) and w2 and (not w3);
        ls <= w0 and (not (w1 or w2 or w3 or w4));
        ss <= (w0 or w1) and (not w2) and w3;
end Behavioral;
```

(3)

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4
5  entity m3 is
6      Port ( input : in  STD_LOGIC;
7             clk : in  STD_LOGIC;
8             ascii : out STD_LOGIC_VECTOR(6 downto 0);
9             v : out  STD_LOGIC);
10 end m3;
11
12 architecture Behavioral of m3 is
13         component DFF
14         port(D,CLK : in std_logic;
15                 Q: out std_logic);
16         end component;
17         component m2
18         port(input : in  STD_LOGIC;
19         clk : in  STD_LOGIC;
20         ld,sd,ls,ss : out  STD_LOGIC);
21         end component;
22         component rom
23         port(addr : in STD_LOGIC_VECTOR(6 downto 0);
24         data : out STD_LOGIC_VECTOR(6 downto 0));
25         end component;
26
27         signal ld,sd,ls,ss : std_logic:= '0';
28         signal addr,data : STD_LOGIC_VECTOR(6 downto 0);
29         signal w0,w1,w2,w3,w4,w5,w6 : std_logic := '0';
30         signal a0,a1,a2,a3,a4,a5,a6 : std_logic := '0';
31         signal lsb,datb,dat : std_logic := '0';
32 begin
33         DEC0 :m2 port map(input,clk,ld,sd,ls,ss);
34         MEM0 :rom port map(addr,data);
35         ascii <= data;
36         v <= lsb and (not ((not w0) and w1 and w2 and w3 and w4 and w5 and w6));
37
38         dat <= (datb and (not (ld or sd))) or ld;
39         DFFDATA :DFF port map(dat,clk,datb);
40         DFFLS :DFF port map(ls,clk,lsb);
41
42         DFF0 :DFF port map(a0,clk,w0);
43         DFF1 :DFF port map(a1,clk,w1);
44         DFF2 :DFF port map(a2,clk,w2);
45         DFF3 :DFF port map(a3,clk,w3);
46         DFF4 :DFF port map(a4,clk,w4);
47         DFF5 :DFF port map(a5,clk,w5);
48         DFF6 :DFF port map(a6,clk,w6);
49
50         addr <= (w0&w1&w2&w3&w4&w5&w6);
51
52         a0 <= ((datb and (ls or ss)) or (w0 and (not(ls or ss)))) and (not lsb);
53         a1 <= ((w0 and (ls or ss)) or (w1 and (not(ls or ss)))) or lsb;
54         a2 <= ((w1 and (ls or ss)) or (w2 and (not(ls or ss)))) or lsb;
55         a3 <= ((w2 and (ls or ss)) or (w3 and (not(ls or ss)))) or lsb;
56         a4 <= ((w3 and (ls or ss)) or (w4 and (not(ls or ss)))) or lsb;
57         a5 <= ((w4 and (ls or ss)) or (w5 and (not(ls or ss)))) or lsb;
58         a6 <= ((w5 and (ls or ss)) or (w6 and (not(ls or ss)))) or lsb;
59
60 end Behavioral;
```

v = 1 の時正しい出力