H27 [1] アルゴリズムとプログラミング

[1] 【必須問題】アルゴリズムとプログラミング

配点：(1-1) 20 点，(1-2) 20 点，(1-3) 25 点，(2-1) 32 点，(2-2) 28 点

(1)

図 1 に示す ANSI-C 準拠である C 言語のプログラム (program) は，重複のない N (自然数，1≦N≦2000 とする) 個の整数を昇順に配列 (array) A の要素 A[1]～A[N] に格納し，変数 x の値が配列 A の要素に含まれているかどうかを 2 分探索法 (binary search) を用いて探索するプログラムである．このプログラムは，探索の途中経過と，x の値が配列 A の要素に含まれる場合はその値が格納されている配列 A のインデックス (index)，x の値が配列 A の要素に含まれない場合は 0 を出力する．N の値の設定，配列 A[1]～A[N] に整数値を設定する処理，変数 x の値を設定する処理，及び，これらの値が妥当かどうかを確認する処理は省略している．以下の各小問に答えよ．

(1-1)　図 1 のプログラムに対して，N，A[1]～A[N]，x を下記のように設定した時の出力結果を書け．

N=8, A[1]=2, A[2]=5, A[3]=8, A[4]=10, A[5]=15, A[6]=20, A[7]=25, A[8]=30, x=8

(1-2)　N=1000 で，x の値が A[1]～A[N] のどこかに存在する場合，図 1 中の do-while ループが実行される回数は最大何回か．その理由も簡潔に説明せよ．

(1-3)　図 1 の do-while ループ中の □で囲われた部分を下記のように変更する．

変更前
```
if (A[mid]<x) left = mid+1;
else          right= mid-1;
```
⇨

変更後
```
if (A[mid]<x) left = mid;
else          right= mid;
```

…のプログラムは正しく結果が出力されるが，変更後のプログラムは正しく結果が出力されないことがあ…ようなN, A[1]～A[N]，x の設定例を一つあげ，「正しく結果が出力されない」状況がどのようなものか…に説明せよ．

```
#include <stdio.h>
/*  この部分でNの値が設定される. */
int main(void)
{
    int A[N+1];
    int mid, left, right, x, index;
/* この部分に下記の処理が記述されているとする.
    ・配列A[1]～A[N]の値が設定される.
    ・変数xに探索すべき値が設定される.
    ・配列A[1]～A[N], xの値が妥当かどうか確認される.
*/
   left = 1;
   right = N;
   do {
         mid= (left+right)/2;
         printf("%d %d %d\n",left,mid,right);
         if (A[mid]<x) left = mid+1;
         else          right= mid-1;
    } while (A[mid] != x && left <= right);
    if (A[mid]==x) index=mid;
    else           index=0;
    printf("%d\n", index);
    return 0;
}
```

図 1　2 分探索法のプログラム

(1-1)

A | / | 2 | 5 | 8 | 10 | 15 | 20 | 25 | 30 |

① l m r
② l m r
③ lr m ⇒ end

(n) 回目の l, r, m の状態 (¥printf の直前)

よって (例)

1 4 8
1 2 3) ¥の printf
3 3 3
3) ¥¥の printf

※ int 型なので小数は切り捨て!!

(Nが偶数のとき)
小数切り捨てのため
左右対称でなく、右部分木の方が多くなる。

(0- while ループの実行回数が最大となるのは、
配列の index 最大の値を探すときである。

[1] A[1] = x のとき

loop	L	R	mid
0	1	1000	—
1	1	499	500
2	1	249	250
3	1	124	125
4	1	61	62
5	1	30	31
6	1	14	15
7	1	6	7
8	1	2	3
9	1	0	1

finish

←条件判定の直前

[2] A[1000] = x のとき

loop	L	R	mid
0	1	1000	—
1	501	1000	500
2	751	1000	750
3	876	1000	875
4	939	1000	938
5	970	1000	969
6	986	1000	985
7	994	1000	993
8	998	1000	997
9	1000	1000	999
10	999	1000	1000

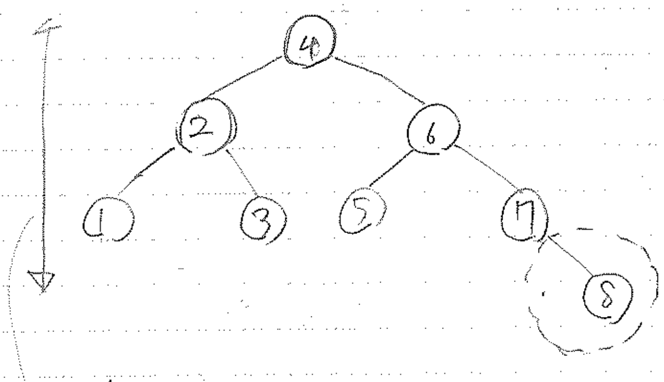
finish

A[mid] == x

完全二分木の左右反転

cf. 木で表現すると、

	1	2	3	4	5	6	7	8
\	2	5	8	10	15	20	25	30



要素数が偶数だと
小数切り捨てのせいで
右部分木の要素数の方が
多くなる。

木の高さ
≈ log N

Ans 10回

(1-3)

[解1] x = A[N] のとき 小数点の切り捨てのため
mid = N とならず かつ、left ≤ right の条件も
崩れず、left = (N-1), right = N,
m = (N-1) の状態が無限に続く。
そのため ループを抜けられる解が出力されない

ここ

	1	2	3	4	5
\	2	5	8	10	(15)

loop	L	R	mid
0	1	5	—
1	3	5	3
2	4	5	4
3	4	5	(4)
⋮	⋮	⋮	

4.5 ⇒ 4

[解2] x が 配列A内に存在しない A[N]より
大きな値の場合、解1と同様無限ループに陥る。

◎ 二分探索 binary search

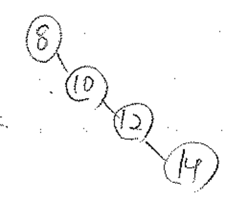
- ソート済み配列に対する探索アルゴリズム
- n個のデータがあるとき、時間計算量は $O(\log N)$
- n個のデータの中央値を見ることで、1回の操作で n/2 個絞り込んでいる
  (odd ⇒ $\frac{n-1}{2}$, even ⇒ $n/2$ or $(n/2)-1$)

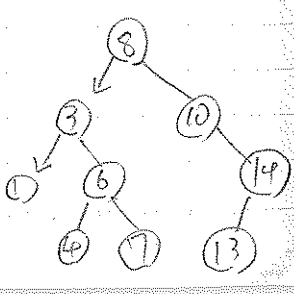
cf. ヒープ

◎ 2分探索木

←手川見もチェック

- 全順序集合に対して、「挿入」、「削除」、「探索」、「最小値探索」を効率的に実施
- 以下の制約を満たす2分木
  "(左の子孫の値) ≤ (親の値) ≤ (右の子孫の値)"
- ソート済み配列を与えると、線形リストと同じになる
- 時間計算量は {平均 $O(\log_2 N)$, 最悪 $O(N)$ ※





（Nが偶数のとき）
小数切り捨ての為
左右対称でなく，右部分木の方が多くなる．

）
(o- while ループの実行回数が最大となるのは、
両てがりの index 最大の値を探すときである．

[1] A[1] = x のとき　　↙条件判定の [2] A[1000] = x のとき

loop	L	R	mid	直前
0	1	1000	—	
1	1	499	500	
2	1	249	250	
3	1	124	125	
4	1	61	62	
5	1	30	31	
6	1	14	15	
7	1	6	7	
8	1	2	3	
9	1	0	1	

finish

loop	L	R	mid
0	1	1000	—
1	501	1000	500
2	751	1000	750
3	876	1000	875
4	939	1000	938
5	970	1000	969
6	986	1000	985
7	994	1000	993
8	998	1000	997
9	1000	1000	999
10	999	1000	1000

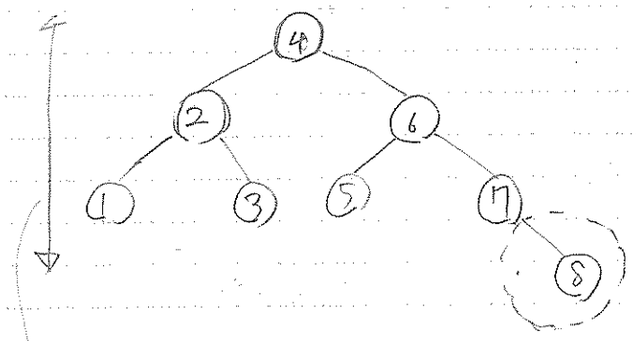
finish

A[mid] == x

完全二分木の左右反転

cf. 木で表現すると

1	2	3	4	5	6	7	8
\	2	5	8	10	15	20	25	30



木の高さ
~ log N

要素数が偶数だと
小数切り捨ての為に
右部分木の要素数の方が
多くなる．

Ans 10回

(1-3)

[例1] x = A[N] のとき 小数点の切り捨てのため
mid = N とならずに、left ≤ right の条件も
崩れず，left = (N-1)，right = N，
m = (N-1) の状態が無限に続く．
そのため ループを抜ける 解が出力されない

	1	2	3	4	5
	2	5	8	10	15

loop	L	R	mid
0	1	5	—
1	3	5	3
2	4	5	4
3	4	5	4

4.5 ⇒ 4

[例2] x が 配列 A 内に存在しない A[N] より
大きな値の場合，例1と同様 無限
ループに陥る．

⑨ 二分探索 binary search

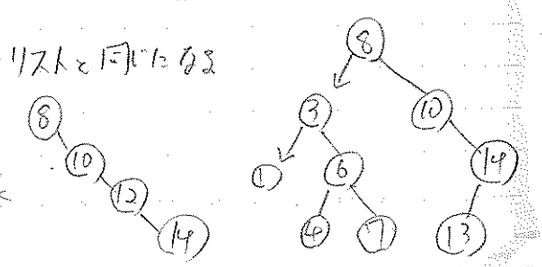
- ソート済み配列に対する探索アルゴリズム
- n 個のデータがあるとき、時間計算量は O(log N)
- n 個のデータの中央値を見ることで、1回の操作で n/2 個相当減らせます
  （odd ⇒ (n-1)/2 こ，even ⇒ n/2 or (n/2)+1，）

cf. ヒープ

⑩ 二分探索木

↙ 手順などをチェック

- 生順序木等合に対して、「挿入」，「削除」，「探索」，「最小値探索」を効率的に実施．
- 以下の制約を満たす二分木
  "(左の子孫の値) < (親の値) < (右の子孫の値)"
- ソート済み配列を与えると、線形リストと同じになる
- 時間計算量は {平均 O(log₂N), 最悪 O(N) 未



H27

(2) 図 2 に示す ANSI-C 準拠である C 言語のプログラム（program）は，0-1 ナップサック問題（0-1 knapsack problem）の解を求めて表示するプログラムである．なお，図の左端の数値は行番号を表す．0-1 ナップサック問題は，ある容量（capacity）のナップサックが一つと，それぞれ大きさ（size）と価値（value）が定められた複数の品物が与えられた時，ナップサックの容量を超えない範囲で価値の和が最大となる品物の組み合わせを求める問題である．本プログラムでは，四つの品物のそれぞれに他と重複しない番号が与えられており，番号 i（1≦i≦4）の品物の大きさと価値が，配列 size の要素 size[i] と配列 value の要素 value[i] としてそれぞれ格納されている．なお，各番号の品物はそれぞれ一つしかなく，また，分割できない．品物の大きさと価値は自然数とする．以下の各小問に答えよ．

(2-1) 22 行目の処理を実行する直前の配列 sack の内容を，解答用紙の表の空欄を埋めることにより答えよ．

sack[i][j]	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9	j=10	j=11	j=12	j=13	j=14	j=15
i=0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
i=1	0	-1	20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
i=2	0	-1	20	-1	-2	-1	30	-1	50	-1	-1	-1	-1	-1	-1	-1
i=3																
i=4																

(2-2) 図 2 のプログラム中の空欄（ア），（イ）を適切な式（expression）で埋めよ．

```
1  #include <stdio.h>
2  #define capacity 15        /* ナップサックの容量 */
3  #define item 4             /* 品物の個数 */
4  int main(void) {
5    int size[] = {0, 2, 6, 6, 2};
6    int value[] = {0, 20, 30, 15, 25};
7    int sack[item+1][capacity+1];
8    int i, j, max, index;
9
10   for (i = 0; i <= item; i++)
11     for (j = 0; j <= capacity; j++)
12       sack[i][j] = -1;   /* ナップサックの初期化 */
13   sack[0][0] = 0;        /* 品物が入っていない(大きさの和が 0)のナップサックの総価値は 0 */
14
15   for (i = 1; i <= item; i++)
16     for (j = 0; j <= capacity; j++)
17       if (sack[i-1][j] != -1) {
18         if (sack[i-1][j] > sack[i][j]) sack[i][j] = sack[i-1][j];
19         if (j + size[i] <= capacity) sack[i][j+size[i]] = sack[i-1][j] + value[i];
20       }
21
22   max = 0; index = 0;
23   for (j = 0; j <= capacity; j++)
24     if (sack[item][j] > max) {max = sack[item][j]; index = j;}
25   for (i = item; i >= 1; i--)
26     if (index >= size[i] && (ア) == (イ)) {
27       printf("item %d is in a knapsack\n",i); index = index - size[i];
28     }
29   return 0;
30 }
```

図 2　0-1 ナップサック問題を解くプログラム

(2-1)

j（ナップサックに入っている容量）

sack[i][j]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0															
1	0	20														
2	0	20			30		50									
3	0	20			30		50						45		65	
4	0	25		45	30		55		75				45		70	

Max

(-1 は省略)

size = [0, 2, 6, 6, 2]
value = [0, 20, 30, 15, 25]

```
for (i=0; i <= item; i++)          ← アイテム
  for (j=0; j<=capacity; j++)      ← capacity
    if ( sack[i-1][j] )
      if ( sack[i-1][j] > sack[i][j] )   sack[i][j] = sack[i-1][j]
        ① 1行上（アイテム i を入れない場合）の方が
           総計の価値が高い
      if (j+size[i] <= capacity)   sack[i][j+size[i]]
        ② 今対象のアイテム i を入れても                = sack[i-1][j] + value[i]
           capacity オーバーにならない
```

(2-1)

入れた順を逆からなぞる ⇒ ② を逆にする．

sack[i][index] == sack[i-1][index-size[i]] + value[i]

※ 逆にもどれたらそのアイテムを入れたということ．

ナップサック問題 knapsack problem　（size と value）

- 「容量 C のナップサックと，n 種類の品物が与えられたとき，容量 C を越えない範囲でいくつかの品物を詰め，V を最大化せよ」
- この決定問題は NP 困難．

動的計画法

- 下記 2 条件を満たすアルゴリズムの総称．
  - ① 分割統治法（部分問題を解き，その結果を利用し問題全体をとく）
  - ② メモ化　（部分問題の計算結果を再利用，表にまとめて計算済みかチェック）
- 実現するには履歴管理を用いるトップダウン方式 or ボトムアップ方式
- それによって指数オーダーの計算時間が，多項式時間におちる．（先に部分問題をとく）

- 動的計画法で厳密解がもとまる．

H27

2 計算機システムとプログラム

配点：(1-1) 5 点，(1-2) 40 点，(1-3) 15 点，
(2-1) 10 点，(2-2) 18 点，(2-3) 24 点，(2-4) 13 点

(1) 浮動小数点数（floating point number）に関する以下の小問に答えよ．なお x を 2 進数（binary number）として解釈する場合，$[x]_2$ と表記する．角括弧（square bracket）をつけない場合は 10 進数と解釈する．

(1-1) 0.625 を 2 進数として表記せよ．

(1-2) IEEE 754 で規定された半精度浮動小数点数表現（representation of half precision floating point number）において，以下の値に最も近い値を表すビット列を解答用紙に書け．

(a) $[1.101]_2$

(b) 5

(c) 0.125

(d) 0.1

(1-3) 2 進数の浮動小数点数表現では，0.1 を誤差なく有限桁で表現できない理由を述べよ．

---

r進数

$$A = \sum_{i=-m}^{n-1} a_i r^i$$
$$= a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \cdots + a_1 r^1 + a_0 + a_{-1} r^{-1} + \cdots + a_{-m} r^{-m}$$

ただし $0 \leq a_i \leq r-1$

r：基数（radix base），$\begin{cases} n: \text{整数部の桁数} \\ m: \text{小数部の桁数} \end{cases}$

小数点の表示

- 固定小数点表示
  - (a) 符号－絶対値表現 ⇒ 最上位ビットを符号
  - (b) 1の補数表現 ⇒ 負の数は1の補数で表現
  - (c) 2の補数表現 ⇒ 負数は2の補数で表現
- 浮動小数点

10進数 ⇔ 2進数

◎ 整数部

○ 10進数 ⇒ 2進数

ex 60 = $(111100)_2$

2) 60		
2) 30	…	0
2) 15	…	0
2) 7	…	1
2) 3	…	1
2) 1	…	1
0	…	1

最下位が上

最上位が0

（たくさん2でわっている。）

○ 2進数 ⇒ 10進数

ex $(111100)_2 = 60$

$(111100)_2$
$= 2^5 + 2^4 + 2^3 + 2^2 + 0 + 0$
$= 32 + 16 + 8 + 4$
$= 60$

◎ 小数

○ 10進数 ⇒ 2進数

ex 0.625 = $(0.101)_2$

整数部

0.625		
×		2
1.250		
1		2
0.50		
×		2
1		0

小数

小数部2倍

0.25
× 2
0.50

○ 2進数 ⇒ 10進数

$(0.101)_2$
$= 0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3}$
$= 0 + \frac{1}{2} + \frac{1}{8}$
$= \frac{5}{8}$
$= 0.625$

8) 5.000
0.625
48
20
16
40

1の補数と2の補数.

▶ 1の補数 One's Complement

$$\begin{cases} |A| = \sum_{i=-m}^{n} \overline{a_i} \cdot 2^i \\ \overline{a_i} = 1 - a_i \end{cases}$$ ただし ビット反転.

ex. $(00100100)_2 = 36$

$\Downarrow$

$(11011011)_{C1}$

▶ 2の補数 Two's Complement.

$$|A| = \sum_{i=-m}^{n-1} \overline{a_i} 2^i$$

ただし $\begin{cases} \overline{a_{-m}} = (1 - a_{-m}) + 1 = 2 - a_{-m} \\ \overline{a_i} = 1 - a_i \end{cases}$ ただし 1の補数の最下桁に1を加えたもの.

ex $(11.011011)_{C1} = (11.011100)_{C2}$

浮動小数点表示. Floating point

$A = m r^e$

[m: 仮数 (mantissa), e: 指数 (exponent), r: 基数 (radix)]

ex. IEEE 754 半精度浮動小数点表現

[$S = 1$ bit, $e = 5$ bits, $m = 10$ bits, $a = 15$]

$$A = (-1)^s \times [1.f]_2 \times 2^{e-15}$$

※ $\begin{cases} e=0 \rightarrow A=0 \\ e=32 \rightarrow A = NAN \ (\infty, \sqrt{-1}, -) \end{cases}$

(1-1)

説明済み $0.625 = (0.101)_2$

(1-2) 半精度浮動小数点変換.

(a) $[1.101]_2$

$(1.101)_2 = +1 \cdot (1.101)_2 \cdot 2^0$

$= +(1.101)_2 \cdot 2^{e-15}$

$e = 15 = (01111)_2$ より

Ans $\overbrace{0}^{s}$ $\overbrace{01111}^{e}$ $\overbrace{1010000000}^{m\ (10\text{bits})}$

(b) 5

$= (101)_2$

$= (1.01)_2 \times 2^2$

$= (1.01)_2 \times 2^{e-15}$

$e = 17 = (10001)_2$ より Ans 0 10001 0100000000

(c) 0.125

$= (0.001)_2$

$= (1.00)_2 \times 2^{-3}$

$= (1.00)_2 \times 2^{e-15}$

$e = +12 = (01100)_2$ より

Ans 0 01100 0000000000

0.125 × 2 = 0.250

0.250 × 2 = 0.50

0.50 × 2 = 1.0

(d) 0.1

$= (0.0001\ 1001\ 1001\ 1001\ \cdots)_2$

$= (1.1001\ 1001\ 1001\ \cdots)_2 \times 2^{-4}$

$= (1.1001\ 1001\ 1001\ \cdots)_2 \times 2^{e-15}$

$e = 10 = (01011)_2$ より Ans 0 01011 1001100110

(1-3)

0.1は2進表現において、(無限)循環小数となるため、仮数部を有限長で表現できない。

(2) メモリ管理に関する以下の小問に答えよ．解答は全て解答欄の太線内に書くこと．

(2-1) 計算機 (computer) においては，読み書きの速度 (reading and writing speed)，容量 (capacity) が異なる複数種類の記憶素子 (memory device) や記憶装置 (memory unit) を連携動作させて，高速な読み書きと大容量の両立を図る，記憶階層 (memory hierarchy) の考え方が導入されている．以下に示す記憶素子または記憶装置 (ア)〜(オ) を有する一般的な計算機において，これらを，読み書きが速く，容量の少ない順に整列し，記号で答えよ．

(ア) 主記憶装置 (main memory unit)，(イ) 2 次キャッシュメモリ (secondary cache memory)，(ウ) 補助記憶装置 (auxiliary memory unit)，(エ) レジスタ (register)，(オ) 1 次キャッシュメモリ (primary cache memory)

(2-1)

速 ↑ R/W ↓ 遅
小 ↑ 容量 ↓ 大

(エ) レジスタ
(オ) 1次キャッシュ
(イ) 2次キャッシュ
(ア) 主記憶装置
(ウ) 補助記憶装置 (⇔ 2次記憶装置)

揮発性
不揮発性，可搬

② 2次キャッシュ (L2キャッシュ; level2 cache)
- L1キャッシュより下位のキャッシュメモリ
- CPUの速度と (メイン) メモリの速度のギャップを埋める．

① 記憶階層、memory Hierarchy
の (小容量×高速)，(中容量×中速)，(大容量×低速) など複数段階にまたがった記憶構造として，各レベルを，為々下位レベルの一時記憶として用いること。
この構造が記憶階層。

(2-2) 以下の文章の空欄 (a)〜(i) に当てはまる最も適切な語句を，下記の選択肢から選び，記号で答えよ．同じ選択肢を複数回用いても良い．

記憶階層の一部を成す仮想記憶 (virtual memory) では，主記憶装置に置くべきプログラムやデータ (以下では "情報" と呼ぶ) の一部を [(a)] に置く．これにより，主記憶装置の持つアドレス空間よりもより広いアドレス空間を有するように，ユーザープログラムに見せかける効果がある．機械語命令 (machine instruction) の中に記載されるアドレスは [(b)]，アドレスバスに送出されるアドレスは [(c)] と呼ばれる．

仮想記憶において，必要な情報が主記憶装置上に無く，[(d)] 上にある場合，この情報は主記憶装置に移動される．この動作は [(e)] と呼ばれる．この時，主記憶装置上に空き領域が無ければ，使用される可能性が低い情報が主記憶装置から [(f)] へ移動され，空き領域が作られる．この動作は [(g)] と呼ばれる．[(e)] や [(g)] が頻繁に発生すると，計算機の処理性能が低下する．

仮想記憶の代表的な実現方法として，メモリを固定長 (fixed size) のブロック (block) で管理する [(h)] 方式と，可変長 (variable size) のブロックで管理する [(i)] 方式とがある．

アドレスバス
⇒ メモリ素子の物理アドレスを伝える。

【選択肢】

(ア) レジスタ	(イ) スワップイン (swap in)	(ウ) フラグメンテーション (fragmentation)
(エ) キャッシュメモリ	(オ) 仮想アドレス (virtual address)	(カ) セグメンテーション (segmentation)
(キ) 補助記憶装置	(ク) スワップアウト (swap out)	(ケ) コンテクストスイッチ (context switch)
(コ) ページング (paging)	(サ) 実アドレス (real address)	(シ) 直接マッピング (direct mapping)

スワッピング
⇒ プロセスのコンテクストスイッチを2次記憶を用いて行っている。
(2次記憶と主記憶のギャップ)
※CPUとのギャップはダメ。

(2-2)

(a) キ：補助記憶装置
(b) オ：仮想アドレス
(c) カ：実アドレス
(d) キ：
(e) イ：スワップイン
(f) キ
(g) ク：スワップアウト
(h) コ：ページング
(i) カ：セグメンテーション

(2-3) ページ枠 (page frame) 数が 4 のページング方式を採用した仮想記憶を考える．以下に示すページ番号 (仮想アドレス空間におけるページの番号) の順にメモリ参照が起こった時，ページ置換アルゴリズム (page replacement algorithm) が LRU (least recently used) 法と FIFO (first-in-first-out) 法のそれぞれについて，主記憶装置上に置かれる各ページ枠が保持するページの番号を答えよ．なお，初期状態では全てのページ枠は空であるとし，解答欄に示されている前半の例を参考に，後半の太線内について解答すること．また，ページフォルト (page fault) が発生する場合はページ番号を○で囲むこと．

参照ページ番号：0, 1, 2, 3, 3, 4, 2, 1, 0, 5, 1, 2

(2-3)

LRU：least recently used，(最後にアクセスされてから最も時間が経過したもの)
FIFO：First in First Out (先入れ先出し)
LFU：least frequently used

LRU

参照ページ	0	1	2	3	3	4	2	1	0	5	1	2
ページ枠 ①	0	0	0	0	0	4	4	4	4	5	5	5
②		1	1	1	1	1	1	1	1	1	1	1
③			2	2	2	2	2	2	2	2	2	2
④				3	3	3	3	3	0	0	0	0

FIFO

参照ページ	0	1	2	3	3	4	2	1	0	5	1	2
ページ枠 ①	0	0	0	0	0	4	4	4	4	4	4	2
②		1	1	1	1	1	1	1	0	0	0	0
③			2	2	2	2	2	2	2	5	5	5
④				3	3	3	3	3	3	3	1	1

~~(2-4)~~

(2-4) ページング方式を採用した仮想記憶を有する計算機で，主記憶装置に入り切らないサイズの整数型配列 (array) A[W*H] を扱うプログラムを考える．以下に示すC言語で書かれた二つのプログラム断片 (ア)，(イ) の内，その処理時間が短くなる方を選び，記号で答えよ．また，その理由を仮想記憶と関連付けて説明せよ．なお，コンパイラ (compiler) による最適化は行われず，キャッシュメモリを有しない計算機で実行するものとする．

(ア)

```
for(x=0; x<W; x++) {
    for(y=0; y<H; y++) {
        sum = sum + x * A[x+(y*W)];
    }
}
```

(イ)

```
for(y=0; y<H; y++) {
    for(x=0; x<W; x++) {
        sum = sum + x * A[x+(y*W)];
    }
}
```

(2-4)

(イ)

仮想記憶の場合，ページまたはセグメントには連続したアドレスのデータが格納される。したがって (イ) のように y を固定して x をシフトするような連続したインデックスでアクセスする方が効率がよい。

(ア) のようにとびとびのインデックスでアクセスすると、データが主記憶になく，ブロックの置き換えの確率が上がり効率が悪い。

ページじゃなくページセグメントじゃなくセグメント

Date 4/27

4 計算理論

配点：(1-1) 10 点，(1-2) 15 点，(1-3) 20 点，(1-4) 15 点，(2-1) 25 点，(2-2) 40 点

(1) 決定性有限オートマトン (deterministic finite automaton) M は 5 項組 $M = (Q, \Sigma, \delta, q_0, F)$ で与えられる．ここで，$Q, \Sigma, \delta, q_0, F$ は，それぞれ，状態 (state) の有限集合，入力記号 (input symbol) の有限集合 (アルファベット (alphabet))，状態遷移関数 (state transition function)，初期状態 (initial state) ($q_0 \in Q$)，受理状態 (accepting state) の集合 ($F \subseteq Q$) である．また，M が受理する言語 (認識する言語) を $L(M)$ と表す．有限オートマトンに関する以下の各小問に答えよ．

(1-1) 決定性有限オートマトン $M = (Q, \Sigma, \delta, q_0, F)$ の状態 p, q ($p \in Q, q \in Q$) が区別不能 (indistinguishable) とは，任意の記号列 $w \in \Sigma^*$ に対し，$\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$ が成り立つことであり，$p \simeq q$ と表す．ここで，Σ^* は，Σ 上の記号列すべての集合 (空系列を含む) を表す．また，任意の状態 $r \in Q$ と任意の記号列 $x \in \Sigma^*$ に対し，$\hat{\delta}(r, x) \in Q$ を以下のように定義する．

$x = \varepsilon$ のとき，$\hat{\delta}(r, x) = r$　(ε は空系列を表す)

$x = ya$ ($y \in \Sigma^*, a \in \Sigma$) のとき，$\hat{\delta}(r, x) = \delta(\hat{\delta}(r, y), a)$

状態集合 Q 上の 2 項関係 (binary relation) $\simeq$ が Q 上の同値関係 (equivalence relation) であることを証明せよ．

	0	1
a	e	d
b	f	d
c	d	f
d	i	g
e	g	i
f	g	h
g	c	b
h	a	c
i	b	c

(1-2) 決定性有限オートマトン $M_1 = (Q_1, \{0,1\}, \delta_1, a, \{a, b\})$ が右の状態遷移表 (state transition table) で与えられたとき，状態集合 $Q_1 = \{a, b, c, d, e, f, g, h, i\}$ が同値関係 $\simeq$ によって，どのような同値類 (equivalence class) に分割 (partition) されるか示せ．

(1-3) $L(M_1) = L(M)$ となる決定性有限オートマトン M の中で，状態数最小のものを M_2 とする．M_2 を状態遷移図 (state transition diagram) で示せ．状態遷移図では，初期状態，受理状態それぞれが分かるように明示すること．

(1-4) $L(M_1) = L(M)$ となる決定性有限オートマトン M の中で，(1-3) で示した M_2 の状態数が最小であることを証明せよ．

(1-1)

同値であることを示すには，反射律，対称律，推移律が成り立つことを示せばよい

[反射律]

$\forall r \in Q, \forall w \in \Sigma^*$ に対して $\hat{\delta}(r, w) \in F \Leftrightarrow \hat{\delta}(r, w) \in F$ が成り立つので 反射律は成立する ($r \simeq r$)

[対称律]

$p \simeq q$

$\forall p, q \in Q, \forall w \in \Sigma^*$ に対して $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$ が成立するならば $\hat{\delta}(q, w) \in F \Leftrightarrow \hat{\delta}(p, w) \in F$ も成立する．

よって 対称律も成立。 ($p \simeq q \Rightarrow q \simeq p$)

[推移律]

$\forall r, p, q \in Q, \forall w \in \Sigma^*$ に対して $r \simeq p$, $p \simeq q$ からは，

$\hat{\delta}(r, w) \in F \Leftrightarrow \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$,

つまり $\hat{\delta}(r, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$ かつで 推移律も成立、

($r \simeq p$ かつ $p \simeq q \Rightarrow r \simeq q$)

以上より $\simeq$ は同値関係を表す．

(def) 同値関係

集合 S において，S の任意の元，a, b, c が

① 反射律 ($a \sim a$) ② 対称律 ($a \sim b \Rightarrow b \sim a$) ③ 推移律 ($a \sim b$ かつ $\Rightarrow a \sim$

の全てを満たすとき、$\sim$ が同値関係であるという．

(1-2) 穴埋めアルゴリズム.

Q ~~Subset Construction~~ (DFA 最小化アルゴリズム)

DFA: $M = (Q, \Sigma, \delta, q_0, F)$
- Q: 状態
- Σ: 入力記号
- δ: 遷移関数
- q_0: 初期状態
- F: 終了状態

▶ 2つの状態が区別できる条件.

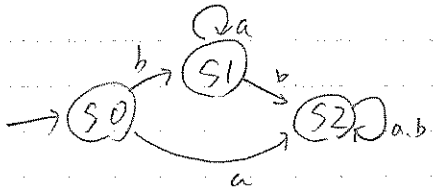
① 基底ステップ ("$p, q \in Q$ について $p \in F$ かつ $q \notin F$ または $p \notin F$ かつ $q \in F$")

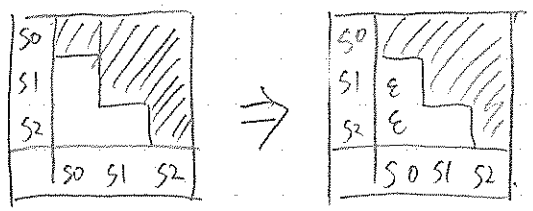
② 帰納的ステップ ("$\forall \alpha \in \Sigma$ に対し $\delta(p, \alpha)$ と $\delta(q, \alpha)$ が区別可能")

▶ アルゴリズム

(Step 1) 下三角テーブル(DISTINCT)を作成する(初期値は空)
2つの状態が統合可能か否かを表現.

(Step 2) 任意の状態ペア (p, q) 毎に、基底ステップが否かを判定. 基底ステップから DISTINCT$(p, q) = \varepsilon$ (統合不能)





(Step. 1)

(Step 3) 任意の状態ペア (p, q) と各入力シンボル α に対して

$$\begin{cases} \text{DISTINCT}(p, q) = \text{空} \\ \text{DISTINCT}(\delta(p, \alpha), \delta(q, \alpha)) \neq \text{空} \end{cases}$$

ならば

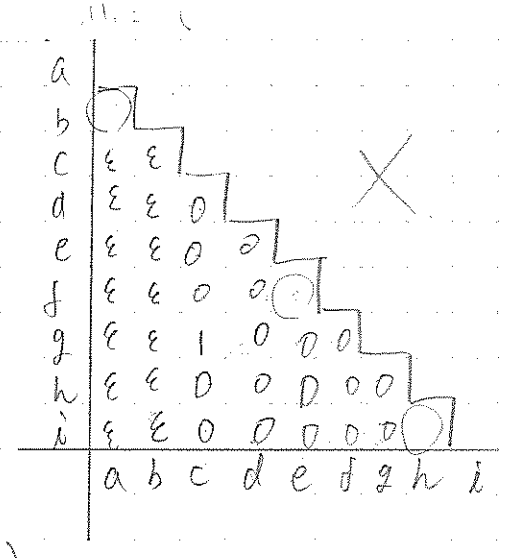
$$\text{DISTINCT}(p, q) = \alpha$$

とおく. (今回の例には該当なし)

(Step 4) 表の中の空の状態ペアが等価な状態.

アルゴリズムのコスト $O(k \cdot n^2)$
- n: 状態数
- k: 繰り返し数

状態	0	1
a	e	d
b	f	d
c	d	f
d	i	g
e	g	i
f	g	h
g	c	b
h	a	c
i	b	c



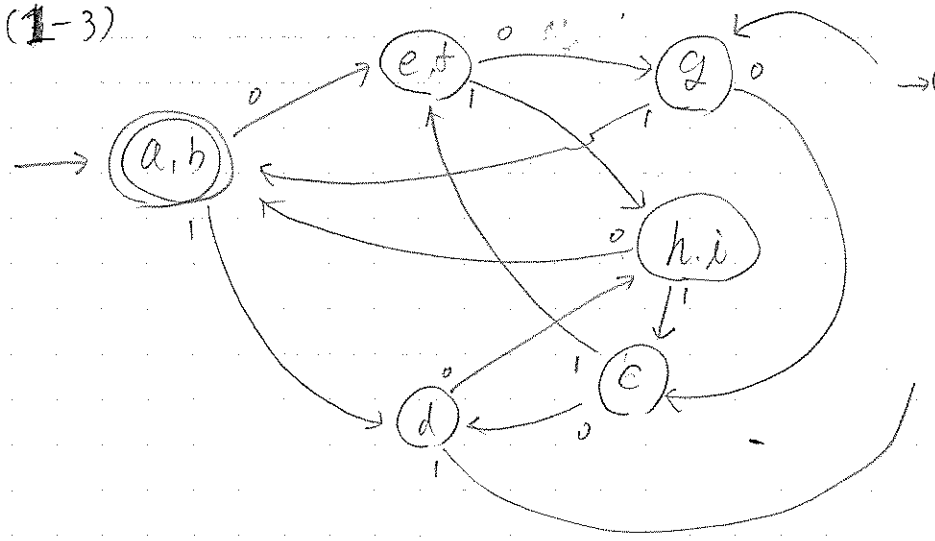
(a,b), (e,f), (h,i) が統合可能

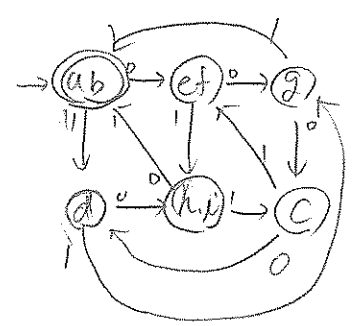
$M_1 = (Q, \{0, 1\}, \delta_1, a, \{a, b\})$

$Q_1 = \{a, b, c, d, e, f, g, h, i\}$

Ans よって 同値類は $\{(a,b), c, d, (e,f), g, (h,i)\}$

(1-3)





こっちの方がキレイ?

(1-4)

M_2 は (1-2) によって互いに同値な状態を全て統合した状態集合からなる DFA である。したがって $L(M_1) = L(M)$ となるオートマトンの中で最小なものである.

(2) 文脈自由言語 (context-free language) に対して次の反復補題 (Pumping Lemma) が知られている．

反復補題 (Pumping Lemma)
文脈自由言語 L に対し，ある非負整数 n が存在し，L に属する n 以上の長さの任意の文字列 z に対して，ある部分文字列分解 $z = uvwxy$ が存在して，以下が成り立つ．

(i) $|vwx| \leq n$

(ii) $vx \neq \varepsilon$　　ただし ε は空系列

(iii) $i \geq 0$ なる任意の i に対して $uv^i wx^i y \in L$

以下の各小問に答えよ．解答用紙には⓪〜㊱ とそれらに対応する解を列挙すること．

(2-1) 文脈自由言語 $L_A = \{a^m b^m \mid m \geq 5\}$ に対して反復補題が成り立つことを確かめたい．以下の文章の空白を埋め，L_A に対し反復補題が成り立つことを示せ．空白⓪〜① にはそれぞれ，指定された条件を満たす要素が入る．例えば［⓪ 非負整数］に対応する要素は非負整数である $1, 40, 100$ 等である．

n として［⓪ 非負整数］を選ぶ．$z = a^K b^K$ ただし，$K \geq$［⑥ 非負整数］なる任意の文 z に対して $z = uvwxy$ なる部分文字列分解として $u =$［⑧ 記号列］，$v =$［⑨ 記号列］，$w =$［⑩ 記号列］，$x =$［⑪ 記号列］，$y =$［⑦ 記号列］を選ぶ．明らかに $|vwx| \leq n$ であり，$vx \neq \varepsilon$，および $i \geq 0$ なる任意の i に対して［② 記号列］$\in L_A$ が成り立つ．なお L_A を生成する文脈自由文法 (context-free grammar) は $(\{S_1, A_1\}, \{a, b\}, P_1, S_1)$ であり (表記の意味は下記参考を参照)，ここで $P_1 = \{S_1 \to$［⑭ 記号列］$, A_1 \to ab,$［① 生成規則］$\}$ である．

(2-1)

$L_A = \{a^n b^m \mid m \geq 5\}$　文脈自由言語

ⓐ $n = 6$

ⓑ $z = a^K b^K$，ただし $K \geq 5$

$z = uvwxy$ と分解

$u = a^{K-1}$, $v = a$ ⓒ, $w = \varepsilon$ ⓓ, $x = b$ ⓔ, $y = b^{K-1}$ ⓕ

ⓖ $uv^i wx^i y \in L$

$P_1 = \{ S \to aaaaA_1 bbbb$ ⓦ
$A_1 \to ab$
$A_1 \to aA_1 b$ ⓚ
$\}$

(2-2)

(2-2) 言語 $L_B = \{a^m b^m c^m \mid m \geq 0\}$ は文脈自由言語ではないことを反復補題を用いて証明したい．以下の証明の空白を埋めよ．空白⑦〜㊱にはそれぞれ，(2-1) と同様指定された条件を満たす要素が入る．ただし，⑤, ⑦については終端記号の数量的性質に言及すること．また，⓪については使用した反復補題の条件を明示すること．

証明
背理法（帰謬法;proof by contradiction）で証明する．言語 L_B が文脈自由文法であると仮定する．このとき仮定より L_B に対して反復補題が成り立つ．以下，反復補題が成り立たないことを示し，矛盾を導く．

n として K という非負整数値を考える．以降の議論は K をパラメータとして考えているので，任意の値を K に与えても成立する．L_B の定義より K を超える長さの文 z が存在する．具体的に $z = a^K b^K c^K$ を考える．z を $uvwxy$ に分割する際，すべての考え得る分割パターンに対して，上記の矛盾を説明する必要がある．

場合 1: vwx が終端記号 c を含まない場合：
このとき y は［⑦ 記号列］を部分文字列として含む．また反復補題の条件 (ii) が成り立つとすると，v, x について以下が成り立つ．
［⑤ v, x に関する条件］
したがって $i = 0$ とおいて L_B に含まれるはずの文 uwy を考えたとき，uwy は L_B に属さない．なぜならば
［① 証明］
すなわち反復補題の条件 (iii) を満たすことはできず，反復補題を満たさない．

場合 2: vwx が［⑨ 終端記号に関する条件］場合：
対象の対称性 (symmetry) より場合 1 と同様である．

場合 3: その他の場合，すなわち vwx が［⑧ 終端記号に関する条件］場合:
場合 3 はそもそも起こりえない．なぜならば
［⓪ 証明］

いずれの場合であっても反復補題を満たさず，よって，矛盾を生じる．
以上の議論は仮定していた命題「言語 L_B は文脈自由文法である」が偽であることを示している．　Q.E.D.

(2-2)

⑦ C^K

⑤ vx は a と b しか含まず，そのうち少なくとも 1 つは含む．

① uwy は K 個の c と K 個未満の a または b を含むため

⑨ 終端記号 a を含まない．　$N=0$ のとき uwy 参照

⑧ 終端記号 a と c を共に含む．

⓪ $|vwx| \leq K$ かつ最後の a と最初の c は $K+1$ だけ離れているため

$z = a \cdots a\, b \cdots b\, c \cdots c$.

$vwx \Rightarrow$ ① ② ③

a と c 両方のペア

6 電子回路と論理設計

配点：(1-1) 20 点，(1-2) 20 点，(1-3) 20 点，(1-4) 15 点，(2-1) 25 点，(2-2) 25 点

4 (1) 次の性質を持つ同期式カウンタ(synchronous counter)について，以下の各小問に答えよ．

1 個の入力(input) x と 6 個の状態(state)をもち，入力 $x = 0$ のときはクロックが入力される毎に状態が S0 → S1 → S2 → S3 → S4 → S5 → S0 と遷移(transition)し，$x = 1$ のときはクロックが入力される毎に状態が S0 → S2 → S4 → S0 と遷移するカウンタを考える．ただし，初期状態(initial state)は S0 とし，入力 $x = 1$ で状態が S1, S3, S5 のいずれかのときは，それぞれ S2, S4, S0 に遷移するものとする．本カウンタを図 1 に示すように 3 個の D フリップフロップ(D flip-flop)と論理ゲート(logic gate)を使って順序回路(sequential circuit)として実現する．D フリップフロップの出力を Q_i (i = 0,1,2) と表す．各状態は表 1 のように割り当てるものとする．また，入力 x はクロック CLK に同期して遅れなく入力されるものとし，初期化時にリセット信号 RST が入力されるとすべての D フリップフロップの出力 Q_i (i = 0,1,2) は 0 になるものとする．

(1-1) 本カウンタの状態遷移図(state transition diagram)を作成せよ．

(1-2) 状態 (Q_2, Q_1, Q_0) の次状態(next state)を (Q_2^+, Q_1^+, Q_0^+) で表す．状態割当付きの状態遷移表を作成し，x, Q_2, Q_1, Q_0 を変数とする Q_2^+, Q_1^+, Q_0^+ の最簡積和形（最小積和形，minimal sum-of-products expression）の論理式（logic expression）を導出せよ．

(1-3) (1-2)で求めた最簡積和形の論理式を用いて，図 1 の順序回路のうち破線内の回路を最小のゲート数で実現せよ．解答用紙には，図 1 全体を記入すること．ただし，論理ゲートは，2 入力 NAND ゲートのみが使用できるものとする．最小であることの証明は不要である．

(1-4) (1-3)で設計した順序回路の最大動作周波数(maximum operating frequency)を求めよ．設計に用いた 2 入力 NAND ゲートの 1 段分の遅延時間(delay time)を T_N[s]，D フリップフロップのセットアップ時間(setup time)を T_S[s]とせよ．ただし，それ以外の時間は考慮しなくてもよいものとする．

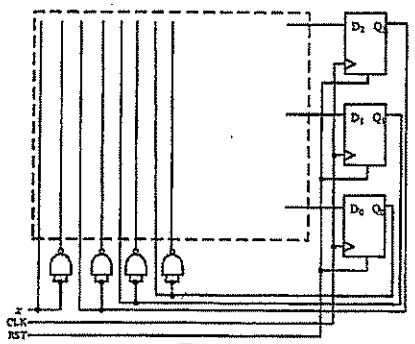
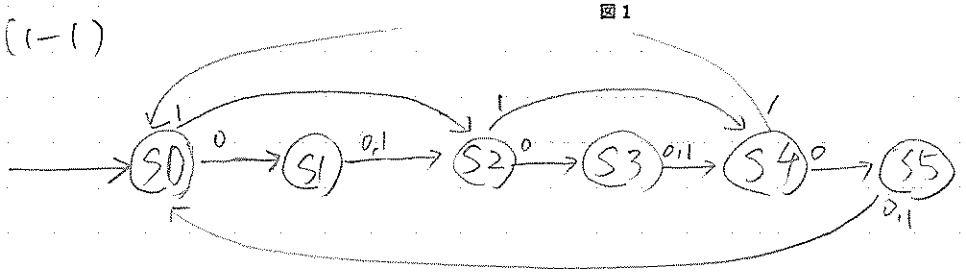


図 1

表 1

状態	状態割り当て		
	Q_2	Q_1	Q_0
S0	0	0	0
S1	0	0	1
S2	0	1	1
S3	1	1	1
S4	1	1	0
S5	1	0	0

(1-1)

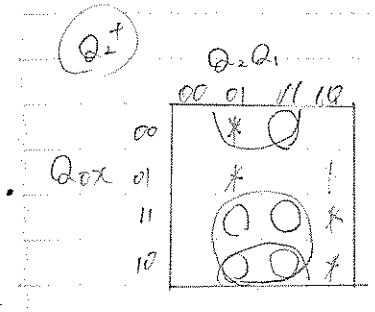


(1-2)

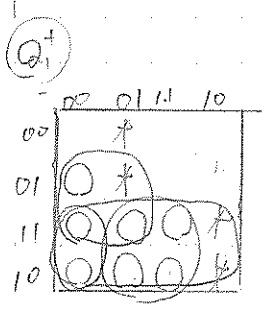
現在の状態	現在の状態割り当て $Q_2\ Q_1\ Q_0$	$Q_2^+ Q_1^+ Q_0^+$ $x=0$	$x=1$
S_0	(0 0 0)	(0 0 1)	(0 1 1)
S_1	(0 0 1)	(0 1 1)	(0 1 1)
S_2	(0 1 1)	(1 1 1)	(1 1 0)
S_3	(1 1 1)	(1 1 0)	(1 1 0)
S_4	(1 1 0)	(1 0 0)	(0 0 0)
S_5	(1 0 0)	(0 0 0)	(0 0 0)

状態遷移図は左の通り．カルノー図を書いて，最小積和形を求める．

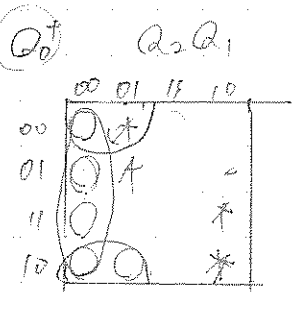
Q_2^+



Q_1^+



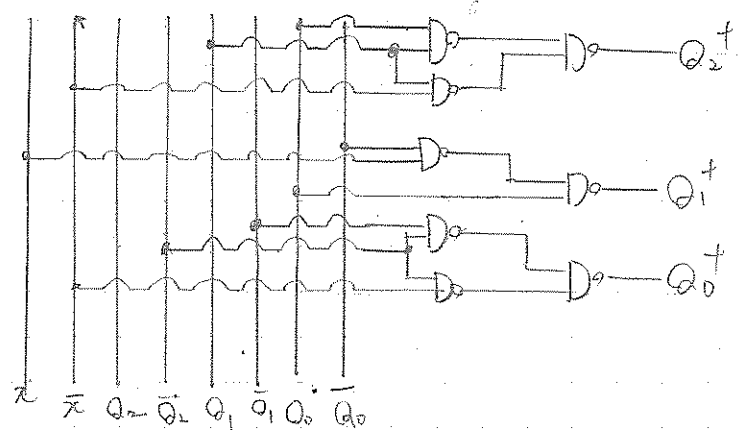
Q_0^+ Q_2Q_1



$$\begin{cases} Q_2^+ = Q_1 Q_0 + Q_1 \bar{x} \\ Q_1^+ = Q_0 + \bar{Q}_0 \bar{x} \\ Q_0^+ = \bar{Q}_2 \bar{Q}_1 + \bar{Q}_2 \bar{x} \end{cases}$$

//

(1-3)



(1-4)

{ NANDゲート1段分の遅延：T_N[s]，
·Dフリップフロップのセットアップ時間：T_S[s]，

回路の最大周波数は
クリティカルパス（最も遅延時間の長いディジタル信号の通り道）によって決まる!!

(1-3)でのクリティカルパスは Q2 の計算で2段のNANDがある
T_NとT_S以外の時間は考慮しなくてよいので、クリティカルパスの遅延は
$2T_N + T_S$ [s]
よって最大周波数は
$1/(2T_N + T_S)$ [Hz]

(2) MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) に関する以下の各小問に答えよ.

(2-1) 図2はn-MOSFETの構造を示している．また，図3はn-MOSFETを用いた回路の例である．以下の文中の空欄(a)〜(f)を適切な語句で埋めよ．

図中の(A)の部分は (a) 型半導体 (semiconductor) で，(B)の部分は (b) 型半導体でできている．ゲート (gate) 電極は酸化膜 (oxide) によって基盤 (substrate/bulk) とは絶縁されている．Vgはゲート電圧，Vdはドレイン (drain) 電圧である．$Vg = 0$のとき，ソース (source)・ドレイン間は絶縁され，電流が流れない．しかし，Vgを (c) よりも大きくすると，基盤と酸化膜の境界に (d) が誘起され， (e) が形成されて電流が流れるようになる．したがって，Vgを制御することによって，オンオフ・スイッチとして動作させることができる．さらに，極性 (polarity) が逆のp-MOSFETと組み合わせて図4のような回路を構成すると， (f) の機能を持つ論理ゲートを実現できる．

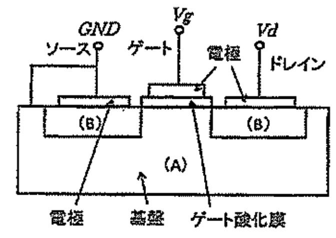


図2

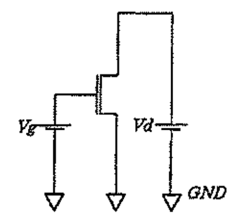


図3

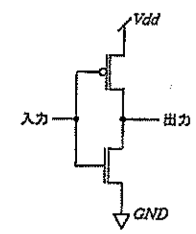


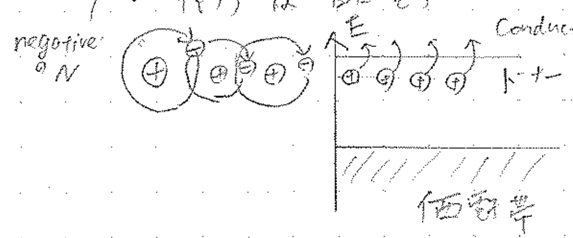
図4

(2-1)

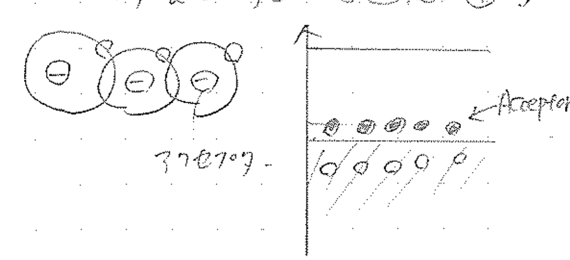
(a) P型半導体　　(d) (自由電子) キャリア
(b) n型半導体　　(e) チャネル
(c)　　　　　　　(f) NOT (否定)

今?　　　??

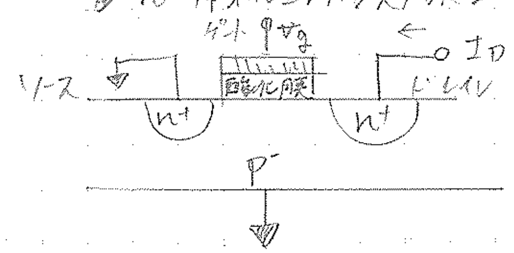
Q N型半導体
· キャリアは自由電子



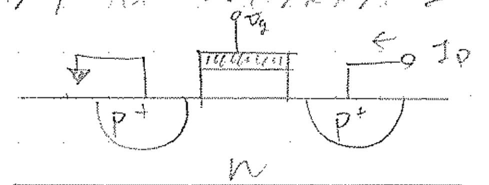
Q P型半導体
· キャリアは正孔（速度が遅い）



Q MOSトランジスタ (Metal-Oxide-Silicon)
n-チャネルエンハンスメント型



p-チャネルエンハンスメント型



(2-2) 以下の図 5 および図 6 に示す回路それぞれについて，入力 A 及び B に電位 Vdd あるいは電位 0 が印加されたときの出力 X の電位を求めよ．解答は表 2 の形式で記せ．ただし，オン状態の MOSFET における電圧降下はないものとする．

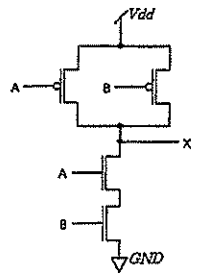


図 5

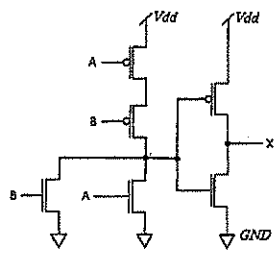


図 6

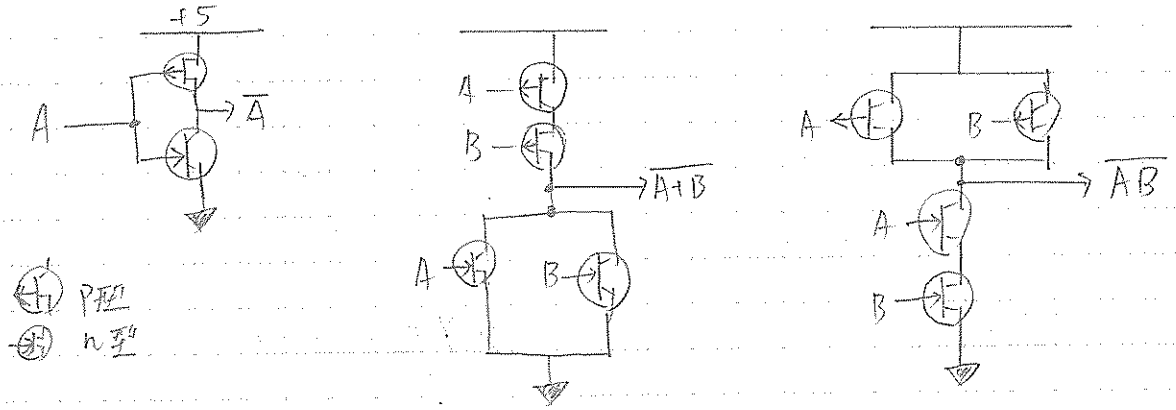
表 2

A	B	X
0	0	
0	Vdd	
Vdd	0	
Vdd	Vdd	

(2-2)

Q 簡単な CMOS 回路．

▷ NOT : $\overline{A}$　▷ NOR : $\overline{A+B} = \overline{A}\cdot\overline{B}$　▷ NAND : $\overline{AB} = \overline{A}+\overline{B}$



Q 与る論理式 X の CMOS 回路の作り方．

① X を NOT, OR, AND の論理式で表現 ⇒ F_1
② ド・モルガンの定理から F_1 の相補式をつくる ⇒ F_2
③ F_1 と F_2 のうち 全体に NOT がかかっている方を n 形チャネル MOS で もう一方を p-形チャネル MOS で 実現
④ ③の回路の抵抗をはずして 完成

(補助)
図 5 は NAND 回路
図 6 は (NOR → NOT) ⇒ OR 回路．

A	B	図 5 X	図 6 X
0	0	Vdd	0
0	Vdd	Vdd	Vdd
Vdd	0	Vdd	Vdd
Vdd	Vdd	0	Vdd

H27

四 数学解析と信号処理

配点: (1-1)30 点, (1-2)35 点, (2)35 点, (3)25 点

以下の各問に答えよ．

(1)

(1-1) 以下のベルヌーイ型の微分方程式 (Bernoulli differential equation) に対して適当な変数変換 (variable transformation) を適用し，線形微分方程式 (linear differential equation) に変換せよ．ただし n は実数であり，$n \neq 0, 1$ とする．

$$\frac{dx}{dt} + P(t)x = Q(t)x^n$$

(1-2) (1-1) の式において，$P(t) = \frac{2}{t}$, $Q(t) = -t^2\cos(t)$, $n = 2$ であるときの $x(t)$ を求めよ．

(1-1)

① 線形微分方程式

独立変数 t をもつ関数 $x = x(t)$ を考える．
一般に t, x, $\frac{dx}{dt}$, $\frac{d^2x}{dt^2}$, …, $\frac{d^nx}{dt^n}$ の関係を表した式を微分方程式 という

$$f\left(t, x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, \cdots, \frac{d^nx}{dt^n}\right) = 0,$$

▷ 変数分離形
$\frac{dx}{dt} = f(t)g(x)$

・$g(x_0)=0$ のとき $x = x_0$ は定数解
・一般解は $\int \frac{1}{g(x)}dx = \int f(t)dt$.

▷ 同次形
$\frac{dx}{dt} = f(x/t)$

・変数変換 $x = tu$ によって
変数分離形 $t\cdot\frac{du}{dt} = f(u) - u$ に帰着．

e.g. $(xy+y^2)dx - x^2dy = 0$
$\longrightarrow \frac{dy}{dx} = (y/x)^2 + (y/x)$

▷ 1階線形微分方程式
$\frac{dx}{dt} + P(t)x = Q(t)$

・一般解

$$x = \exp\left[-\int P(t)dt\right]\cdot\left[\int Q(t)e^{\int P(t)dt}dt + C\right]$$

▷ ベルヌーイ型　$\alpha \neq 1, 0$.
$\frac{dx}{dt} + P(t)x = Q(t)x^\alpha$

・変数変換 $y = x^{1-\alpha}$ によって
1階線形微分方程式に帰着．

▷ 完全微分方程式
$P(x,y)\,dx + Q(x,y)\cdot dy = 0$

[完全微分方程式の必要十分条件]
$\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x}$ //

一般解は $\int P(x,y)\,dx - \int\left[\frac{\partial}{\partial y}\int P(x,y)\,dx - Q(x,y)\right]dy = C$ //

[解] $\frac{dx}{dt} + P(t)x = Q(t)x^n$

$y = x^{1-n}$ として変数変換をする。 $y' = (1-n)x^{-n}\frac{dx}{dt}$ であるが
両辺に $(1-n)x^{-n}$ をかけると

$$y' + (1-n)P(t)\,x^{1-n} = Q(t)(1-n)$$

$$\therefore\ y' + (1-n)P(t)\,y = (1-n)Q(t)$$

(1-2) このとき 線形微分方程式は

$$y' + \frac{2(1-n)}{t}y = -(1-n)t^2\cos(t)$$

$n=2$ より

$$y + \frac{2}{t}y = -t^2\cos(t) \quad \cdots ①$$

ここで

$$P(t) = \exp\left(-\int \frac{2}{t}dt\right) = \exp(\log t^2) = t^{-2}$$

を両辺にかけると

$$t^{-2}y' - 2t^{-3}y = \cos(t)$$

$$(t^{-2}y)' = \cos(t)$$

よって

$$t^{-2}y = \int \cos(t)\,dt = +\sin(t) + C \quad (C\text{の積分定数})$$

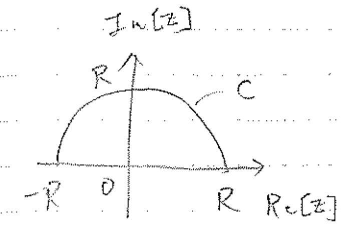
$$\therefore\ y = t^2\sin(t) + Ct^2$$

$y = x^{-n} = x^{-1}$ より $\quad x = \dfrac{1}{t^2(\sin(t)+C)}$

(2)
留数定理 (residue theorem) を用いて，以下の定積分 (definite integral) I を求めよ.

$$I = \int_{-\infty}^{\infty}\frac{dx}{x^2+x+1}$$

(1-2)



$\oint_C \frac{dz}{z^2+z+1}$ の複素積分を考える。 ④テキスト p80〜81
Cは図中の積分経路である。また $R>1$ とする
$(z^2+z+1) = \left(z - \frac{-1+\sqrt{3}i}{2}\right)\left(z - \frac{-1-\sqrt{3}i}{2}\right)$
であり Cの内部に $z = \frac{-1+\sqrt{3}i}{2} = w$ を1位の極としてもつ.
留数定理より

$$\oint_C \frac{dz}{z^2+z+1} = 2\pi i\,\mathrm{Res}_w\left[\frac{1}{z^2+z+1}\right]$$

$$= 2\pi i\left(\lim_{z\to w}(z-w)\frac{1}{z^2+z+1}\right)$$

$$= 2\pi i\left(\lim_{z\to w}\frac{1}{(z-\bar{w})}\right) \quad \bar{w}\text{は}w\text{の共役複素}$$

$$= 2\pi i\left[\frac{-1+\sqrt{3}i}{2} - \frac{-1-\sqrt{3}i}{2}\right]$$

当ってる? ? ⇒

$$= 2\pi i\cdot\sqrt{3}i = -2\sqrt{3}\pi$$

次に $R\to\infty$ を行う. このとき積分路が動く領域には特異点はないので, $\oint_C \frac{1}{z^2+z+1}dz$ の値は変わらない。

積分路 $C' = C - [-R, R]$ とすると

$$\left|\int_{C'}\frac{dz}{z^2+z+1}\right| \leq \max_{z\in C'}\frac{1}{|z^2+z+1|}\cdot\pi R \leq \frac{\pi R}{R^2+R+1}$$

$R\to\infty$ としたとき 0に収束することがわかるので,

$$\int_{[-R,R]}\frac{1}{z^2+z+1}dz \longrightarrow \int_{-\infty}^{\infty}\frac{dx}{x^2+x+1}$$

要復習

よって

$$\int_{-\infty}^{\infty}\frac{dx}{x^2+x+1} = -2\sqrt{3}\pi$$

(3)

図1と図2に示すサンプリング周波数が$1/T$である以下の離散時間信号(discrete-time signal) $x(nT)$と$y(nT)$を考える.

$$x(nT) = \{x(0T) = 2, x(1T) = -3, x(2T) = 1, x(3T) = -2, x(4T) = -1, x(5T) = 0\}$$
$$y(nT) = \{y(0T) = 0, y(1T) = 2, y(2T) = -3, y(3T) = 1, y(4T) = -2, y(5T) = -1\}$$

ただし，nは整数であり，$n < 0$または$n > 5$のとき$x(nT) = 0$，$y(nT) = 0$とする.

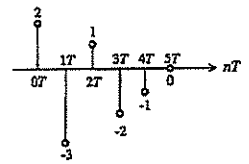


図 1: 離散時間信号 $x(nT)$

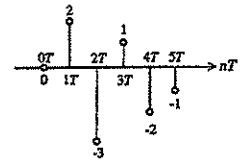


図 2: 離散時間信号 $y(nT)$

以下の文章の空欄 (a) から (e) を適切な語句または数式で埋めよ．ただし，解答用紙には (a) から (e) とそれらに対応する解の組を列挙すること.

図1と図2から，$y(nT)$は$x(nT)$に対して[(e)]サンプル分だけ遅れている (delay) ことが分かる．一方，$x(nT)$と$y(nT)$のZ変換 (Z-transform) はそれぞれ，$X(z) =$[(b)]，$Y(z) =$[(c)]となる．この結果から，$X(z)$と$Y(z)$には

$$Y(z) = \boxed{\text{(d)}}\, X(z)$$

の関係があることが分かる．[(a)]サンプル分の遅延は，上式の[(d)]に対応しており，kサンプル分の遅延は[(e)]の乗算に相当する．これはZ変換の時間シフト (time shifting) に関する性質によるものである.

(3)

(a) 1(サンプル)
(b) $2 - 3z^{-1} + z^{-2} - 2z^{-3} - z^{-4}$
(c) $2z^{-1} - 3z^{-2} + z^{-3} - 2z^{-4} - z^{-5}$
(d) z^{-1}
(e) z^{-k}

◎Z変換.

両側Z変換　$Z[x_n] = X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}$

片側Z変換　$Z[x_n] = X(z) = \sum_{n=0}^{\infty} x_n z^{-n}$

逆Z変換　$x_n = Z^{-1}[x_n] = \frac{1}{2\pi i}\oint_C X(z) z^{n-1} dz$.

(部分分数分解などで逆変換)

※今回の場合

$x[n] = 2\delta(0) - 3\delta(T) + \delta(2T) - 2\delta(3T) - \delta(4T)$

$Z[\delta(n)] = 1$