情報理工学演習III　演習課題（第3回）

テーマ　　：二分探索
実施日　　：2020年4月30日（木）
担当　　：有村 博紀，堀山 貴史，金森 憲太朗（TF），加井 丈志（TA）
授業連絡先：enshuiii-csit@ist.hokudai.ac.jp（演習課題の提出・質問など）
担当連絡先：kanamori@ist.hokudai.ac.jp（金森），kai@ist.hokudai.ac.jp（加井）

以下の各問に関して，それぞれプログラムを実装せよ． 実装したコードは必ず，以下に示す提出様式を守って，**本授業のメールアドレス（enshuiii-csit@ist.hokudai.ac.jp）宛にメールで提出**すること．

- メールの件名　：「[csit ex3]xxxxxxxx」，ここで xxxxxxxx は自分の学生番号．
- 添付ファイル名：第 n 回演習の問 m に対するコードのファイル名は，「n_m.c」とする．

例：学生番号02180000の学生の場合

- メール宛先　：enshuiii-csit@ist.hokudai.ac.jp
- メール件名　：[csit ex3]02180000
- 添付ファイル：3_1.c ，3_2.c ，3_3.c ，3_4.c

メール本文には，学生番号，所属コース，名前，実施回（第3回演習），解いた問題番号を記載すること． 課題提出によって出席をとるため，解いている途中の状態でも構わないので，演習時間内に一度は提出すること．

実装にはC言語を用い，gcc（ver.4.8.5 以降）でコンパイルすること．

コンパイル・実行例

```
% gcc 3_1.c        // コンパイル
% ./a.out          // 実行
```

演習時間内に解き終わらなかった場合は，次回の演習時間までに提出すること．期間中であれば，何度でも再提出してよい． 提出状況等は本授業のホームページまたはELMSグループ（moodle）上に掲示予定なので，適宜確認すること．

問題番号に * が付されたものは，発展課題である．解答は必須ではないが，解ければ成績評価に加点されるので，是非挑戦してほしい．

【問1】 以下の Sample 1 は，

1. int 型の値を格納する空の二分探索木 T を作成
2. T に 7，3，10，1，4，8，6，5 を順に追加
3. T を深さ優先探索し，通りがけ順（in-order）で表示

を行うプログラムである． これを 3_1.c として実装せよ．

Sample 1

```
1: #include <stdio.h>
2: #include <stdlib.h>
3:
4: typedef struct _node{
5:     int element;
```

```
 6:     struct _node *right;
 7:     struct _node *left;
 8: }node;
 9:
10: typedef struct _tree{
11:     node *root;
12: }tree;
13:
14: /* 空の二分探索木を作成する関数 */
15: tree* create() {
16:     tree *T = (tree *)malloc(sizeof(tree));
17:     T->root = NULL;
18:     return T;
19: }
20:
21: /* 二分探索木 T に,木の高さに対して線形時間で
22:    ノード node を追加する関数 */
23: void insert_body(node *now, node *add) {
24:     // 左に行く
25:     if (now->element > add->element) {
26:         if(now->left == NULL) { now->left = add; }
27:         else{ insert_body(now->left, add); }
28:     }
29:     // 右に行く
30:     else{
31:         if(now->right == NULL) { now->right = add; }
32:         else{ insert_body(now->right, add); }
33:     }
34:     return;
35: }
36:
37: void insert(tree *T, int element) {
38:     node *add = (node *)malloc(sizeof(node));
39:     add->element = element;
40:     add->right = NULL;
41:     add->left = NULL;
42:
43:     if(T->root == NULL) { T->root = add; }
44:     else{ insert_body(T->root, add); }
45:     return;
46: }
47:
48: /* 二分探索木 T を深さ優先探索し,ノード数に対して線形時間で
49:    通りがけ順(in-order)で表示する関数 */
50: void in_order_body(node *now) {
51:     if(now == NULL) { return; }
52:     in_order_body(now->left);
53:     printf("%d ", now->element);
54:     in_order_body(now->right);
55:     return;
56: }
```

```
57:
58: void in_order(tree *T) {
59:     in_order_body(T->root);
60:     printf("\n");
61:     return;
62: }
63:
64: int main() {
65:     tree *T = create();
66:
67:     insert(T,7);
68:     insert(T,3);
69:     insert(T,10);
70:     insert(T,1);
71:     insert(T,4);
72:     insert(T,8);
73:     insert(T,6);
74:     insert(T,5);
75:
76:     in_order(T);
77:
78:     return 0;
79: }
```

実装したプログラム 3_1.c を実行し，結果を確認せよ．

【問2】 以下を実行するプログラムを 3_2.c として実装せよ．

1. int 型の値を格納する空の二分探索木 T を作成
2. T に 7，3，10，1，4，8，6，5 を順に追加
3. T を深さ優先探索し，行きがけ順（pre-order）で表示
4. T を深さ優先探索し，帰りがけ順（post-order）で表示

二分探索木の表示は Sample 1 における通りがけ順（in-order）の出力と同様に，各ノードの直後に半角スペースを置いてこれらを区切り，二分探索木全体の出力の末尾には改行を入れること．

実装したプログラム 3_2.c を実行し，結果を確認せよ．

【問3】 非負整数を5つ受け取り，それぞれがソート済みの配列 [2, 3, 5, 6, 7, 9, 10, 12, 15, 20, 21, 27, 28, 29, 39, 41, 42, 44, 46, 48, 50] 中に存在するか否かを，各 $O(\log n)$ 時間で探索するプログラムを 3_3.c として実装せよ．ここで，n は配列の要素数である．
入力は標準入力から与えられるものとする．入力された非負整数が配列中に存在する場合は Found を，存在しない場合は Not Found を，それぞれ改行区切りで順に出力すること．

入力例

```
4
43
20
1
29
```

出力例

```
Not Found
Not Found
Found
Not Found
Found
```

実装したプログラム 3_3.c に以下の Input 1 を入力して実行し，結果を確認せよ．

Input 1

```
21
33
52
5
49
```

【問4*】 農夫のクラーク氏は，飼っている M 頭の牛たちを入れるために，N 個の牛舎を持つ小屋を作った（$2 \leq M \leq N$）．各牛舎は直線上に並んでおり，i 番目の牛舎は位置 x_i にある（$0 \leq x_1 < x_2 < \cdots < x_N$）．昨今の情勢に鑑みて，牛たちのソーシャルディスタンスの確保を重要視したクラーク氏は，牛たちを他の牛とできるだけ離れるようにそれぞれ牛舎に入れることにした．N，M，$x_1, \ldots, x_N$ を受け取り，最も近い二頭の牛の間の距離を最大化し，その値を $O(N \cdot \log x_N)$ 時間で出力するプログラムを 3_4.c として実装せよ（ヒント：求める最大値に関して二分探索を行う！）．
入力は以下の形式で標準入力から与えられるものとする．出力の末尾には改行を入れること．

入力形式

```
N M        // N は牛舎の数，M は牛の頭数
x_1
x_2
...
x_N        // x_i は i 番目の牛舎の位置（1 ≤ i ≤ N，0 ≤ x_1 < x_2 < ... < x_N）
```

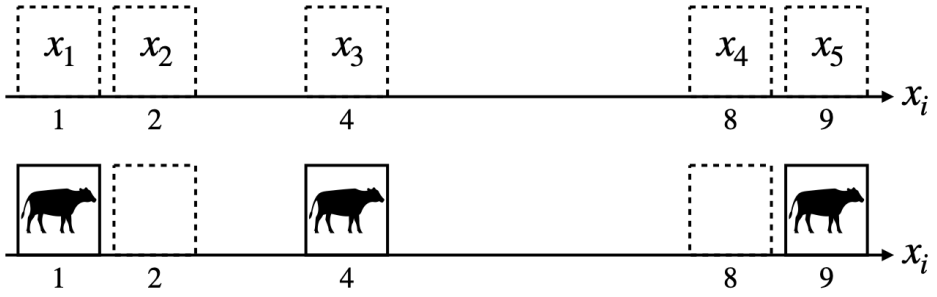

図 1　$N = 5$，$M = 3$，$x_1 = 1$，$x_2 = 2$，$x_3 = 4$，$x_4 = 8$，$x_5 = 9$ のとき，位置 $x_1 = 1$，$x_3 = 4$，$x_5 = 9$ にある牛舎に牛を入れることで，最も近い二頭の牛の間の距離を最大化（最大値 3）できる．

実装したプログラム 3_4.c に以下の入力例（$N = 5$，$M = 3$，$x_1 = 1$，$x_2 = 2$，$x_3 = 4$，$x_4 = 8$，$x_5 = 9$）を入力して実行し，結果を確認せよ．

入力例

```
5 3
1
```

```
2
4
8
9
```

出力例

```
3
```