DSA HW3    B05902022 張雅信    Problem 1~3

Problem 1

1.hash table

S={18, 34, 9, 37, 40, 32, 89}

h1(S)={ 7, 1, 9, 4, 7, 10, 1}

h2(S)={ 9, 5, 10, 8, 1, 3, 10}

```
double hashing
round 1   | 0| 0| 0| 0| 0| 0| 0|18| 0| 0| 0|
round 2   | 0|34| 0| 0| 0| 0| 0|18| 0| 0| 0|
round 3   | 0|34| 0| 0| 0| 0| 0|18| 0| 9| 0|
round 4   | 0|34| 0| 0|37| 0| 0|18| 0| 9| 0|
round 5   | 0|34| 0| 0|37| 0| 0|18|40| 9| 0|
round 6   | 0|34| 0| 0|37| 0| 0|18|40| 9|32|
round 7   |89|34| 0| 0|37| 0| 0|18|40| 9|32|

Linear probing
round 1   | 0| 0| 0| 0| 0| 0| 0|18| 0| 0| 0|
round 2   | 0|34| 0| 0| 0| 0| 0|18| 0| 0| 0|
round 3   | 0|34| 0| 0| 0| 0| 0|18| 0| 9| 0|
round 4   | 0|34| 0| 0|37| 0| 0|18| 0| 9| 0|
round 5   | 0|34| 0| 0|37| 0| 0|18|40| 9| 0|
round 6   | 0|34| 0| 0|37| 0| 0|18|40| 9|32|
round 7   | 0|34|89| 0|37| 0| 0|18|40| 9|32|
```

2.(a)

According to the graph, n=2.

11不會指到10指到的那欄

[00] = [4k], [01] = [4k+1], [10]=[4k+2], [11]=4k+3.  thus,
2 in [01] is wrong -> 2 belongs to [10]
13 in [10] is wrong -> 13 belongs to [01]

| Graph | | | | |
|---|---|---|---|---|
| **0 0** | 128 | 64 | 32 | 16 |
| **0 1** | 25 | 1 | 17 | 13 |
| **1 0** | 2 | 10 | 14 | |
| **1 1** | 31 | 7 | 3 | |

2.(b)

| Index | | Data | | | |
|---|---|---|---|---|---|
| **0 0** | 0 0 0 | 128 | 64 | 32 | 16 |
| | 1 0 0 | 4 | | | |
| **0 1** | **0 0 1** | 25 | 1 | 17 | 49 |
| | **1 0 1** | 13 | | | |
| **1 0** | | 2 | 10 | 14 | 30 |
| **1 1** | | 31 | 7 | 3 | |

3.
兩個玩家分別先將自己要出的拳輸入hash function, 然後在FB訊息貼上hash值, 雙方在互相確認對方出拳.但為了避免hash值被認出, 需要為每次出拳配置一個亂碼.

Algorithm:
Let (rock, paper, scissors)=(1,2,3)
要求每位玩家出拳的同時,自己附上3碼數字亂碼abc,
hash function= 7^a+11^b+13^c+17^(出拳)
FB訊息上雙方同時打出 ( hash值 , abc )
一眼看不出來對方出什麼就解決了有人慢出的問題
雙方都出拳後再互相由hash值及亂碼反推出拳即可得到結果

4.(a)
[6, 31, 2, 41, 30, 45, 44]

| T1 index | Data |
|---|---|
| 0 | |
| 1 | |
| 2 | 44 |
| 3 | 31 |
| 4 | |
| 5 | |
| 6 | 6 |

| T2 index | Data |
|---|---|
| 0 | 2 |
| 1 | |
| 2 | |
| 3 | |
| 4 | 30 |
| 5 | 41 |
| 6 | 45 |

4.(b)
取 T1 三格與 T2 兩格（ h1(k)=x1,x2,x3  h2(k)=y1,y2,y3 ） ，總共只能填 6 筆資料，
但( xi,yi ) 共有 9 種組合。當 7 種組合出現時，會出現infinite sequence of displacement
Let x1=2,x2=3,x3=6, y1=4,y2=0,y3=6
2 -> (2,0)
6 -> (6,0)
30 -> (2,4)
31 -> (3,4)
44 -> (2,6)
45 -> (3,6)

34 -> (6,4)

Insert 34-> move 6 -> 2-> 44-> 45-> 31-> 30-> 2 -> 6 -> 34……

infinite sequence of displacement occur when inserting 34

Problem 2

1. Max Heap:[10, 9, 8, 6, 7, 2, 1, 5, 4, 3 ]
   In-order: [ 5, 6, 4, 9, 7, 3,10, 2, 8, 1]

2. Min Heap: [1, 4, 2, 5, 7, 8, 3, 10, 6, 9 ]
   In-order: [10, 5, 6, 4, 7, 9, 1, 8, 2, 3]

3.

```
int q,index=0;
int arr[q];

void find_smaller(struct Node *t){
        arr[index]=t->data;
        index++;
        if(t->left != NULL)
            if(t->left->data < q)
                find_smaller(t->left);
        if(t->right != NULL)
            if(t->right->data < q)
                find_smaller(t->right);
        return;
}
int main(){
        #gain q and min tree
        find_smaller(root);
        print arr, k=strlen(arr)
}
```

The function find_smaller will be called k times,
Each time has a complexity of O(1),
so total complexity if O(k).

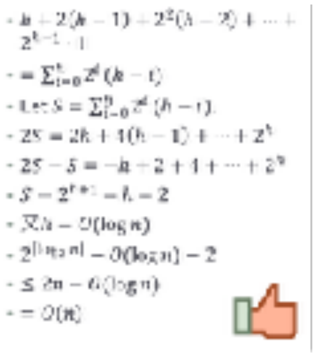
4.

(Reference: heap 上課ppt)
如果有個資料在 a[ i ] ,(index從1開始)
他會在第 [log i] 層, 他的parent是 a[ i/2 ]
他的children是 a[2i], a[2i+1]

```
#array index start from 1
void heapify( int arr[]){
        L=strlen(arr)
        for(i=L/2;i>=0;i--){
                j=i
                while(2j<=L){   //is not leaf
                child_1=arr[2j]
                child_2=arr[2j+1]
                if(arr[j]<child_1)
                        swap(arr[j],child_1)
                        j=2j
                else if(arr[j]<child_2)
                        swap(arr[j],child_2)
                        j=2j+1
                else
                        break
                }
        }
```

}

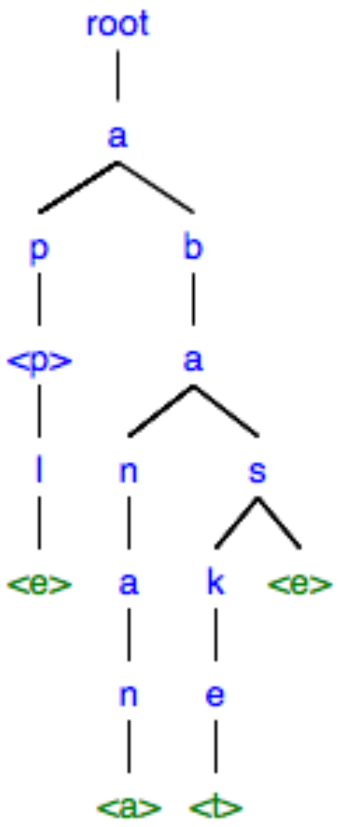
according to [ heap 上課ppt  page10 ] —>
Time complexity analysis =O(N)



- $h + 2(h-1) + 2^2(h-2) + \cdots + 2^{h-1} \cdot 1$
- $= \sum_{i=0}^{h} 2^i (h-i)$
- Let $S = \sum_{i=0}^{h} 2^i (h-i)$.
- $2S = 2h + 4(h-1) + \cdots + 2^h$
- $2S - S = -h + 2 + 4 + \cdots + 2^h$
- $S = 2^{h+1} - h - 2$
- 又 $h = O(\log n)$
- $2^{\lfloor \log_2 n \rfloor} - O(\log n) - 2$
- $\leq 2n - O(\log n)$
- $= O(n)$

Problem 3

1. < x > represent an end of a word(double circle)
   graph———>



2.

```
struct Node *root;
void insert(char word[],int N){
        struct Node *temp=root;
        for(int i=0;i<N;i++){
                temp=temp->children[word[i]-'a'];
                temp->is_word=1;
                temp->tag++;
        }
}
void delete(char word[],int N){
        struct Node *temp=root;
        for(int i=0;i<N;i++){
                temp=temp->children[word[i]-‘a'];
                temp->is_word=0;
                if(temp->tag > 0) temp->tag—;
        }
}
int query(char word[], int N){
        struct Node *temp=root;
        for(int i=0;i<N;i++){
                if(temp->children[word[i]-'a']->is_word==0)
                        return 0;
                temp=temp->children[word[i]-'a'];
        }
        return 1;
}
```

each function contain a single for loop, which will run N times, and each time takes O(1).

3.

建兩棵樹，一顆是原本的trie (trie1)，另一顆先把輸入字串(Wi)反過來再建trie (trie2)。
如此做，則，若S’ 是S反過來的字串，找
“S is a suffix of how many words in trie1”的作法等同於 “S’ is a prefix of how many words in trie2”.
另外，當 insert & delete 的時候改變tag值 (見上題code)，
tag值即是從root到該字的字串是多少個字的prefix。

```
int how_many_prefix(char word[], int N){
    struct Node *temp=root;
    for(int i=0;i<N;i++){
        if(temp->children[word[i]-'a']->is_word==0)
            return 0;
        temp=temp->children[word[i]-'a'];
    }
    return temp->tag;
}
```

4.

```
struct Node *root;
//initialize: level=0,temp=root;
int is_leaf(struct Node *temp){
    int k=1;
    for(int i=0;i<26;i++)
        if(temp->children[i]->is_word!=0)
            k=0;
    return k;
}
int first_player_strategy(struct Node *temp, int level){
    if (is_leaf)        //when is_leaf and level is odd, first player win, return true
        return (level%2);
                        //when level is odd, second player's turn
    if(level%2){        //if all choices lead to first player's win, return true
        int now=1;,
        for(int i=0;i<26;i++)
            if(!first_player_strategy(temp->children[i],level+1))
                now=0;
        return now;
    }
                        //when level is even, first player's turn
    else{               //if any choice lead to first player's win, return true
        int now=0;
        for(int i=0;i<26;i++)
            if(first_player_strategy(temp->children[i],level+1))
                now=1;
        return now;
    }
}
int second_player_strategy(struct Node *root){
    int ans=1;              //if all choices lead to second player's win,return true
    for(int i=0;i<26;i++)
        if(!first_player_strategy(root->children[i],0))
            ans=0;
    return ans;
}
int neither_has(struct Node *root){
    if(!first_player_strategy(root,0) && !second_player_strategy(root))
        return 1;
    return 0;
}
```