Algorithm HW7_report

A. 程式說明

此題為 optimal binary search tree，它的目標是使其在搜索時，它的期望成本最小化。將題目給好的 p 和 q 直接存入陣列(下圖做了兩個分別為不同題目的 optimal BST，一個為問題討論的題目)。

```
40      int n=9;
41      double p[10]={0,0.05,0.04,0.02,0.07,0.08,0.09,0.04,0.08,0.03};
42      double q[10]={0.08,0.06,0.04,0.04,0.03,0.06,0.07,0.06,0.04,0.02};
43      /*int n=5;
44      double p[6]={0,0.05,0.15,0.15,0.1,0.1};
45      double q[6]={0.05,0.1,0.05,0.1,0.05,0.1};*/
```

經過 Optimal BST 函式能算出最小的 cost 並知道樹的根在哪裡，最後運用遞迴的方式先印出根，再將樹拆解成左右子樹，並一直遞迴下去，直到遇到 base case。

```
24  void print_Optimal_BST(int **root ,int ini , int final ,int r, int n){
25      if(ini>final)
26          return;
27      if(root[ini][final] == r){
28          cout<<"Root of Tree is "<<root[ini][final]<<"\n";
29      }
30      else if( root[ini][final]<r){
31          cout<<r<<"'s leftsubtree is  "<<root[ini][final]<<"\n";
32      }
33      else if(r <root[ini][final] ){
34          cout<<r<<"'s rightsubtree is  "<<root[ini][final]<<"\n";
35      }
36      print_Optimal_BST(root ,ini , root[ini][final]-1 ,root[ini][final], n);
37      print_Optimal_BST(root ,root[ini][final]+1 , final ,root[ini][final], n);
38  }
```

(此部分為加分題)

B. 程式結果

```
Smallest serch cost : 3.45
Root : 5

Root of Tree is 5
5's leftsubtree is  2
2's leftsubtree is  1
2's rightsubtree is  4
4's leftsubtree is  3
5's rightsubtree is  7
7's leftsubtree is  6
7's rightsubtree is  8
8's rightsubtree is  9
```

(註解 43-45 行的結果)

```
Smallest serch cost : 2.75
Root : 3

Root of Tree is 3
3's leftsubtree is  2
2's leftsubtree is  1
3's rightsubtree is  4
4's rightsubtree is  5
```

(註解 40-42 行的結果)

C. 問題討論

在使用動態規劃做 optimal BST 時，判斷期望值時，有時候會出現等於的情況，以至於可能同時會出現多種的 optimal BST 並且它們的 Expected Search Cost 都是一樣且都是最小的。

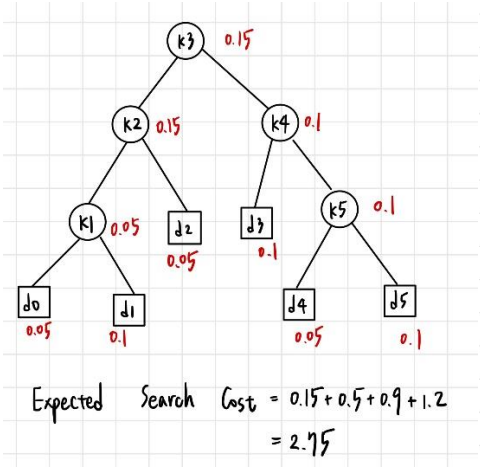
```
double t = e[i][r-1]+e[r+1][j]+w[i][j];
if(t<e[i][j]){
    e[i][j]=t;
    root[i][j]=r;
}
```

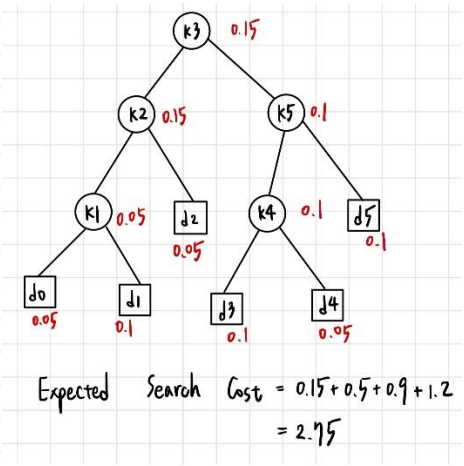
```
Smallest serch cost : 2.75
Root : 3

Root of Tree is 3
3's leftsubtree is  2
2's leftsubtree is  1
3's rightsubtree is  4
4's rightsubtree is  5
```

(判斷沒加等號)

```
Smallest serch cost : 2.75
Root : 3

Root of Tree is 3
3's leftsubtree is  2
2's leftsubtree is  1
3's rightsubtree is  5
5's leftsubtree is  4
```

(判斷有加等號)





最後會發現可能會有兩種以上不同的 Optimal BST 但他們的 Expected Search Cost 都相同並且都是最小的。