演算法導論 HW7

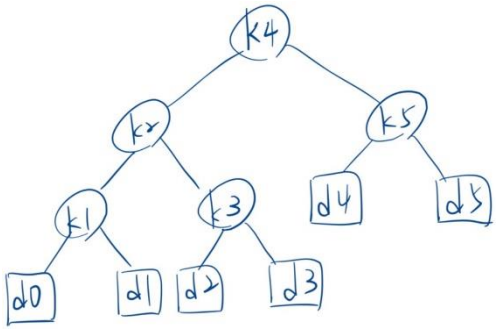
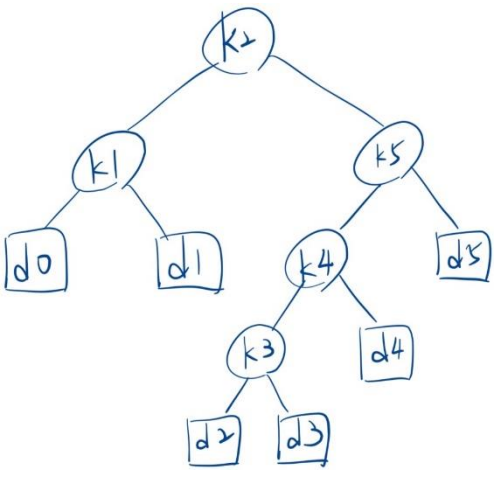
110511254 徐煜絨

一、程式架構

在 main function 中，程式會先詢問使用者要手動輸入 p 和 q，或是直接使用作業說明的 pdf 檔上面的 p 和 q。如果輸入'm'，則會繼續詢問要輸入幾個 k，並要求輸入 p, q；如果輸入'p'，就會直接進入 optimal_bst 這個 function。

```
Enter 'm' if you want to determine the p and q yourself.
Enter 'p' if you want to use the p and q given in the pdf file.
m
How many k: 9
```

因為 optimal binary search tree 並不唯一，可能會有多種建起來的方式的 cost 都是最小值，其原因在於 e[i, r-1]+e[r+1, j]+w[i, j] == e[i, j]，所以當發生這種情況時，程式會詢問是否有修改原先的 root table，如果輸入'y'，則 root[i, j]就會更新成新的 r，如果輸入'n'則維持。最後會 print 出輸入的 y, n 順序，對於 y, n 數量或順序的不同，都有可能影響最後 tree 的長相。以課本 Figure 15.9 為例，輸入'y', 'y'和輸入'y', 'n'的 cost 都是 2.75，但兩者卻不同。

y y	y n
Wanna change the root table? (y/n): y Wanna change the root table? (y/n): y Is the root table updated when t = e[i][j]: y y Smallest search cost:2.75 Root:4 Optimal Binary Search Tree: k4 is the root k2 is the left child of k4 k1 is the left child of k2 d0 is the left child of k1 d1 is the right child of k1 k3 is the right child of k2 d2 is the left child of k3 d3 is the right child of k3 k5 is the right child of k4 d4 is the left child of k5 d5 is the right child of k5	Wanna change the root table? (y/n): y Wanna change the root table? (y/n): n Is the root table updated when t = e[i][j]: y n Smallest search cost:2.75 Root:2 Optimal Binary Search Tree: k2 is the root k1 is the left child of k2 d0 is the left child of k1 d1 is the right child of k1 k5 is the right child of k2 k4 is the left child of k5 k3 is the left child of k4 d2 is the left child of k3 d3 is the right child of k3 d4 is the right child of k4 d5 is the right child of k5
	

除了 optimal_bst，還有另外三個 function 協助印出 tree，輸出方式參考課本 Exercise 15.5-1。在 pri 裡，會先輸出整個 tree 最上面的 node，即 tree 的 root。

```
void pri(vector<vector<int>> root, int n){
    int r = root[1][n];  // the root of the whole tree
    cout << "k" << r << " is the root" << endl;

    // func(ans.r, lower bound, upper bound, root of the sub tree)
    left(root, 1, r-1, r);
    right(root, r+1, n, r);
}
```

接著會進到 left 或 right function 中，印出 node 的 child。若輸出是 dn，則代表走到底並 return。

```
void right(vector<vector<int>> root, int l, int u, int r){

    if(l>u){
        cout << "d" << u << " is the right child of k" << r << endl;
        return;
    }

    else{
        int rt = root[l][u];
        cout << "k" << rt << " is the right child of k" << r << endl;
        left(root, l, rt-1, rt);
        right(root, rt+1, u, rt);
        return;
    }
}
```

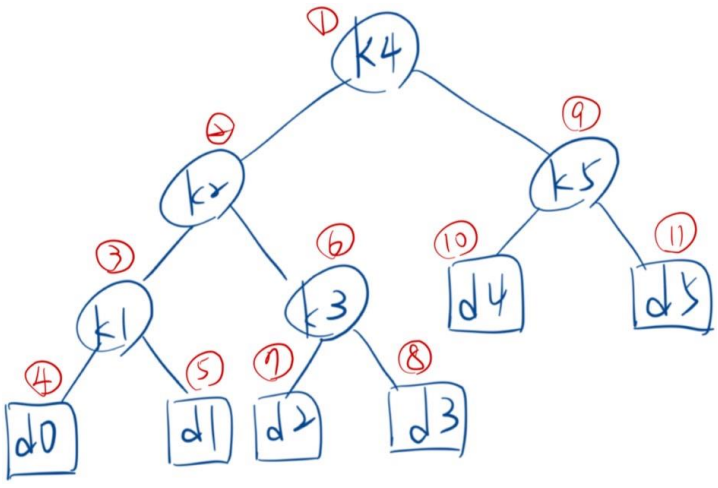
```
void left(vector<vector<int>> root, int l, int u, int r){

    if(l>u){
        cout << "d" << u << " is the left child of k" << r << endl;
        return;
    }

    else{
        int rt = root[l][u];
        cout << "k" << rt << " is the left child of k" << r << endl;
        left(root, l, rt-1, rt);
        right(root, rt+1, u, rt);
        return;
    }

}
```

右圖為輸出順序
（以課本 Figure 15.9 的其中一種解為例）

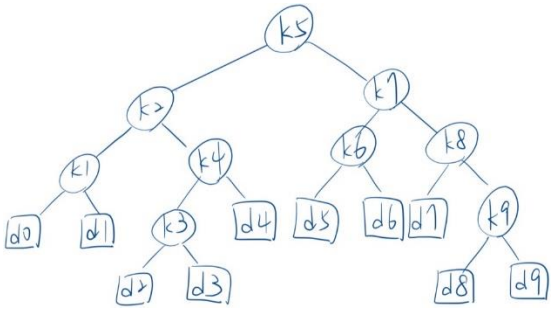


二、實際測試

這次作業有給定 p 和 q 的值，所以這邊展示的結果是用給定的資料作測試所得。其中可以發現會有兩次 t == e[i][j]，而最後得到的 smallest search cost 是 3.45、root 是 5，和計算結果相同。

```
Enter 'm' if you want to determine the p and q yourself.
Enter 'p' if you want to use the p and q given in the pdf file.
p
Wanna change the root table? (y/n): y
Wanna change the root table? (y/n): y
Is the root table updated when t = e[i][j]:
y y
Smallest search cost:3.45
Root:5
Optimal Binary Search Tree:
k5 is the root
k2 is the left child of k5
k1 is the left child of k2
d0 is the left child of k1
d1 is the right child of k1
k4 is the right child of k2
k3 is the left child of k4
d2 is the left child of k3
d3 is the right child of k3
d4 is the right child of k4
k7 is the right child of k5
k6 is the left child of k7
d5 is the left child of k6
d6 is the right child of k6
k8 is the right child of k7
d7 is the left child of k8
k9 is the right child of k8
d8 is the left child of k9
d9 is the right child of k9
```

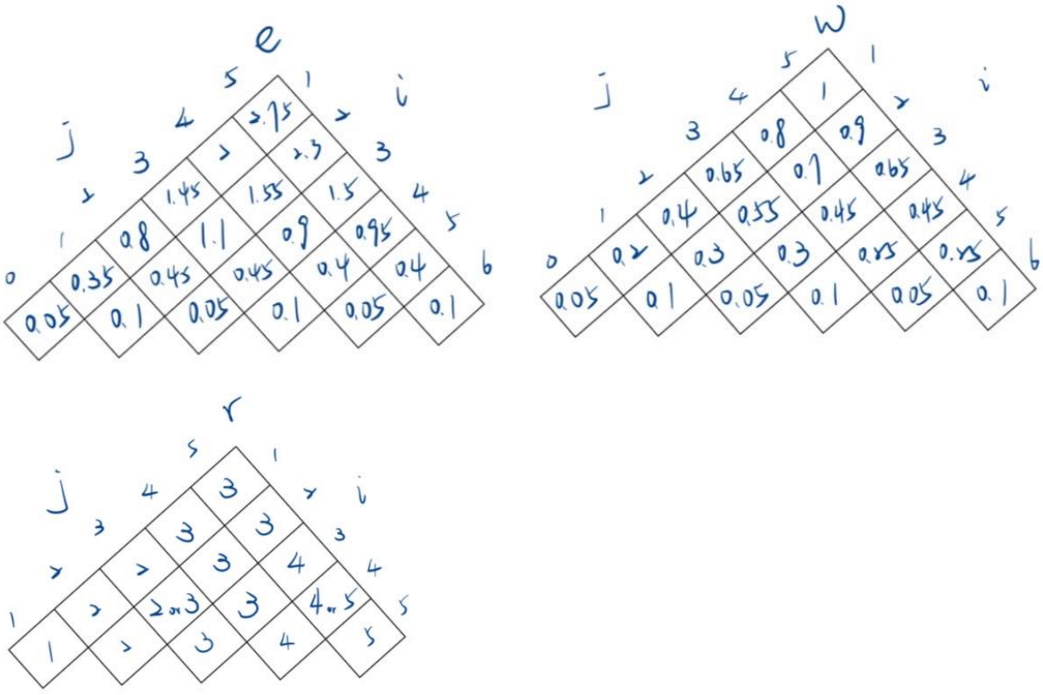
經過測試會發現，不管輸入'y' 'y', 'y' 'n', 'n' 'y', 'n' 'n'，都不會影響 tree 的長相。這邊定義可能被改變的 root table 的格子為 x，不影響長相的原因在於執行 left 和 right 兩個 function 時，不會使用到 x 的值，所以不論有沒有更新 x，都不會造成結果改變。



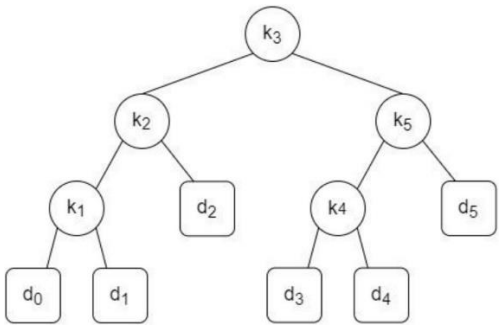
node	depth	probability	contribution	node	depth	probability	contribution
k1	2	0.05	0.15	d0	3	0.08	0.32
k2	1	0.04	0.08	d1	3	0.06	0.24
k3	3	0.02	0.08	d2	4	0.04	0.2
k4	2	0.07	0.21	d3	4	0.04	0.2
k5	0	0.08	0.08	d4	3	0.03	0.12
k6	2	0.09	0.27	d5	3	0.06	0.24
k7	1	0.04	0.08	d6	3	0.07	0.28
k8	2	0.08	0.24	d7	3	0.06	0.24
k9	3	0.03	0.12	d8	4	0.04	0.2
				d9	4	0.02	0.1
						total	3.45

三、討論

先計算 smallest search cost：



可以知道 smallest search cost 是 2.75。



node	depth	probability	contribution
k1	2	0.05	0.15
k2	1	0.15	0.3
k3	0	0.15	0.15
k4	2	0.1	0.3
k5	1	0.1	0.2
d0	3	0.05	0.2
d1	3	0.1	0.4
d2	2	0.05	0.15
d3	3	0.1	0.4
d4	3	0.05	0.2
d5	2	0.1	0.3
Total			2.75

由表格計算結果可以發現作業 pdf 檔所提供的 tree，其 cost 為 2.75。和上面所計算的 smallest search cost 相同，所以這個是 optimal binary search tree。

輸入不同數量 y, n 或同數量不同順序（左邊兩張建立起來的 tree 長一樣，右邊兩張也一樣）：

```
Is the root table updated when t = e[i][j]:
y y
Smallest search cost:2.75
Root:3
Optimal Binary Search Tree:
k3 is the root
k2 is the left child of k3
k1 is the left child of k2
d0 is the left child of k1
d1 is the right child of k1
d2 is the right child of k2
k5 is the right child of k3
k4 is the left child of k5
d3 is the left child of k4
d4 is the right child of k4
d5 is the right child of k5
```

```
Is the root table updated when t = e[i][j]:
y n
Smallest search cost:2.75
Root:3
Optimal Binary Search Tree:
k3 is the root
k2 is the left child of k3
k1 is the left child of k2
d0 is the left child of k1
d1 is the right child of k1
d2 is the right child of k2
k4 is the right child of k3
d3 is the left child of k4
k5 is the right child of k4
d4 is the left child of k5
d5 is the right child of k5
```

```
Is the root table updated when t = e[i][j]:
n y
Smallest search cost:2.75
Root:3
Optimal Binary Search Tree:
k3 is the root
k2 is the left child of k3
k1 is the left child of k2
d0 is the left child of k1
d1 is the right child of k1
d2 is the right child of k2
k5 is the right child of k3
k4 is the left child of k5
d3 is the left child of k4
d4 is the right child of k4
d5 is the right child of k5
```

```
Is the root table updated when t = e[i][j]:
n n
Smallest search cost:2.75
Root:3
Optimal Binary Search Tree:
k3 is the root
k2 is the left child of k3
k1 is the left child of k2
d0 is the left child of k1
d1 is the right child of k1
d2 is the right child of k2
k4 is the right child of k3
d3 is the left child of k4
k5 is the right child of k4
d4 is the left child of k5
d5 is the right child of k5
```