Algorithms

Classwork# Video 4.3-4.5

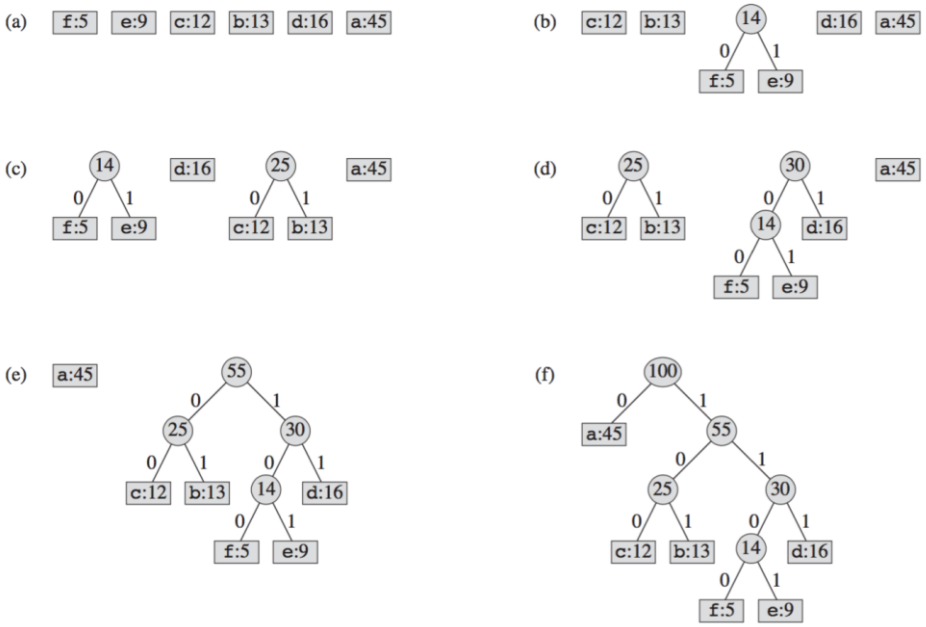
組別: 姓名: 分數: 批改者:

Each problem is 3 points.

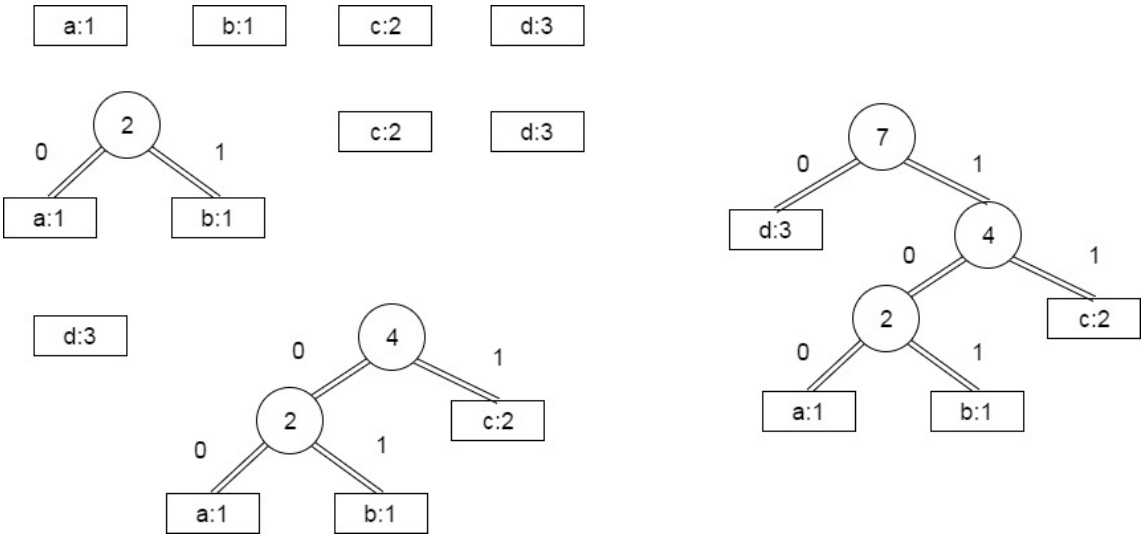
3 = correct. 2 = one mistake. 1 = many mistakes. 0 = no answer.

1. Huffman Coding [4.5]

There are four alphabets in our code: <*a,b,c,d*>. Given the text <*acdbddc*>, please count the frequency of each alphabet. What is the optimal Huffman code for the following sequence? Please show your process to derive the Huffman tree in the same manner as figure below.



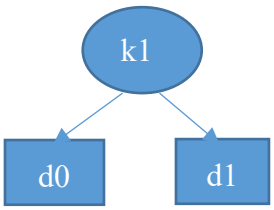
Sol:



2. Optimal binary search tree [4.3]

Write C++ code for the procedure *CONSTRUCT-OPTIMAL-BST* with the global table root. Please outputs the structure of an optimal binary search tree in **pre-order** traversal.

Hint: first call of this function is *CONSTRUCT-OPTIMAL-BST(1, n, 0),* where *n* is the number of distinct keys.



This example will show following words

k1 is the root
d0 is the left child of k1
d1 is the right child of k1

```
void CONSTRUCT-OPTIMAL-BST (int i, int j, int p){
//i is smallest index of subtree, j is largest index of subtree, p is the parent of subtree
    if (root[i][j] == root[1][n]){
        cout << "k" << root[i][j] << " is the root" << endl;
    }
    else if (_____________){
        if (j < p)
            cout << "d" << j << " is left child of " << "k" << p << endl;
        else
            cout << "d" << j << " is right child of " << "k" << p << endl;
        return;
    }
    else{
        if (root[i][j] < p)
            cout << "k" << root[i][j] << " is left child of " << "k" << p << endl;
        else
            cout << "k" << root[i][j] << " is right child of " << "k" << p << endl;
    }
    CONSTRUCT-OPTIMAL-BST (_____________, _____________, _____________);
    CONSTRUCT-OPTIMAL-BST (_____________, _____________, _____________);
}
```

Sol:

```
void CONSTRUCT-OPTIMAL-BST (int i, int j, int p){
    if (root[i][j] == root[1][n]){
        cout << "k" << root[i][j] << " is the root" << endl;
```

```
    }
    else if (j == i - 1){//j<i , j==p || i==p
        if (j < p)
            cout << "d" << j << " is left child of " << "k" << p << endl;
        else
            cout << "d" << j << " is right child of " << "k" << p << endl;
        return;
    }
    else{
        if (root[i][j] < p)
            cout << "k" << root[i][j] << " is left child of " << "k" << p << endl;
        else
            cout << "k" << root[i][j] << " is right child of " << "k" << p << endl;
    }
    CONSTRUCT-OPTIMAL-BST (i, root[i][j] - 1, root[i][j]);
    CONSTRUCT-OPTIMAL-BST (root[i][j] + 1, j, root[i][j]);
}
```

3. Activity selection [4.4]

Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it has *greedy choice property*.

Sol:

The proposed approach - selecting the last activity to start that is compatible with all previously selected activities - is really the greedy algorithm but starting from the end rather than the beginning.

Prove: If S_k is nonempty and a_m is the last activity to start in S_k, then a_m is included in some optimal solution.

Let A_k be an optimal solution to S_k, a_j have the last start time of any activity in A_k. If $a_j = a_m$, then done. Otherwise, let $A_k' = A_k - a_j + a_m$, then activities in A_k' is disjoint because activities in A_k are disjoint and a_j is the last activity to start in A_k, and s_m will less than s_j. Since $|A_k| = |A_k'|$, $|A_k'|$ is also optimal solution to S_k.

4. Find McDonald's stop [4.4]

Patrick decides to bike from Taipei to Kaohsiung to on Saturday. He needs to take a rest at a McDonald's every 20 kilometers, and his map gives the distance between each pair of McDonald's along his linear route (i.e., no detour). Assume McDonald's from Taipei to Kaohsiung are denoted as a set $S = \{c_1, c_2, \dots, c_n\}$ and the distance between each pair of McDonald's are also denotes as $\{d_2, \dots, d_n\}$ and $d_i = |c_i - c_{i-1}|$, $d_i \leq 20$. Assume Taipei is c_0 and Kaohsiung is c_{n+1}. So d_1 is the distance between Taipei and first McDonald's and d_{n+1} is the distance between last McDonald's and Kaohsiung.

Patrick's goal is to minimize the number of McDonald to take a rest along his route from Taipei to Kaohsiung. Please prove the problem has *greedy choice property,* so this problem can be solved by greedy algorithm.

Sol:

Prove: If S is nonempty and c_i is the last McDonald whose distant from last stop is 20 km, then c_i is included in some optimal solution.

Suppose $A \subseteq S$ is an solution by the greedy algorithm, and the first stop which is picked is c_i. Suppose here exists another optimal solution $B \subseteq S$ and its first stop is c_j. Since A is greedy, the first stop c_i of A is farther than c_j which is the first stop of B. Also, both A and B are optimal, the number of McDonald's are the same. Therefore, we can exchange c_j with c_i, B is also an optimal solution.

5. Find McDonald's stop (cont'd) [4.4]

From the previous problem, please give an efficient algorithm to determine at which McDonald's he should stop.

$Find\_stop(d, n)$
1 $A = \emptyset$
//TODO

return A

Sol:

$Find\_stop(d, n)$

1 $A = \emptyset$
2 $D = 0$
3 for $i = 0$ to n
4 $D = D + d_{i+1}$
5 If $D > 20$
6 $A = A \cup \{i\}$
7 $D = d_{i+1}$
8 return A