**Advanced Algorithms**

**Exercise for Lecture 11,12**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** | **周思源** | **Student ID** | 24S151092 |
| **Lecture 5** |  | | |
| **Lecture 6** |  | | |
| **Lecture 7** |  | | |
| **Total Score** |  | | |
| **Notes** | Deadline: **2024-10-24 24:00**  Submission Format: ‘**Lecture11-12\_Name\_ID.docx**’, and please send to: **algo1\_23fall@163.com**.  This assignment is meant to be an evaluation of your **individual** understanding coming into the course and should be completed **without collaboration** or outside help. | | |

**Lecture 11**

**Problem 11.1 [20 points]** Determine an LCS of and using the following table.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
|  | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 |
|  | 0 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
|  | 0 | 1 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 |
|  | 0 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 4 | 4 |
|  | 0 | 1 | 2 | 2 | 3 | 4 | 4 | 4 | 4 | 5 |
|  | 0 | 1 | 2 | 3 | 3 | 4 | 4 | 4 | 5 | 5 |

**Solution:**

BCBAC

BCBDC

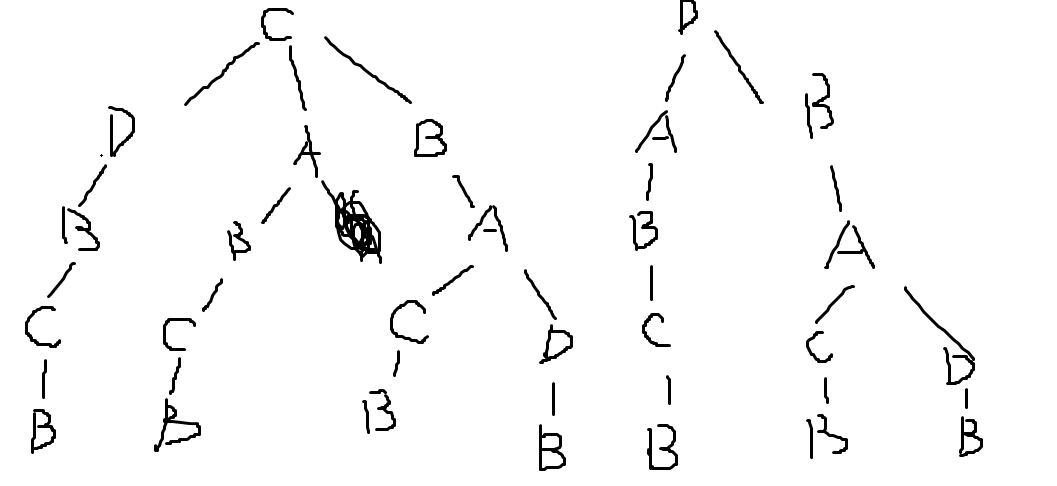
BCABC

BDABC

BCBAD

BCABD

BDABD



**Problem 11.2 [30 points]** A currency has bills in the following denomination: 1, 4, 7, 13, 28, 52, 91, 365. Your goal is to design an algorithm for finding the smallest number of bills that can be used for a given amount of currency.

1. Describe a recursive backtracking algorithm for counting the smallest number of bills that is needed to produce a given amount of currency. E.g., 90 can be represented as one 52 bill, one 28 bill, one 7 bill, and three 1 bills, for a total of 6. Do not make this algorithm efficient or analyze it.
2. Design and analyze a dynamic programming algorithm for finding the smallest number of bills.

**Solution:**

Int min\_bill(int\* denomination, int n){

For(int I = denomination.length() – 1; I >= 0; i--){

If(n >= denomination[i])return f(denomination, n – denomination[i]) + 1;

}

}

Int min\_dp\_bill(int\* denomination, int n){

Int dp[n + 1];

Dp[0] = 0;

For(int I = 1; I <= n; i++)dp[i] = INT\_MAX;

For(int I = 1; I <= n; i++){

For(int j = 0; j < denomination.length() && n >= denomination[j]; j++)

Dp[i] = fmin(dp[i], dp[I – denomation[j]] + 1);

}

Return dp[n];

}

**Lecture 12**

**Problem 12.1 [20 points]** There exist 7 characters and we have the occurrence frequency of them. Please give the huffman code of each character.You need show each step of them coding process. Left branch of huffman tree will be 0 and right branch of huffman tree will be 1.

G: 4 D: 8 C: 12 E: 11 B: 16 F: 20 A:25

**Solution:**

As the following picture

G:0000

D:0001

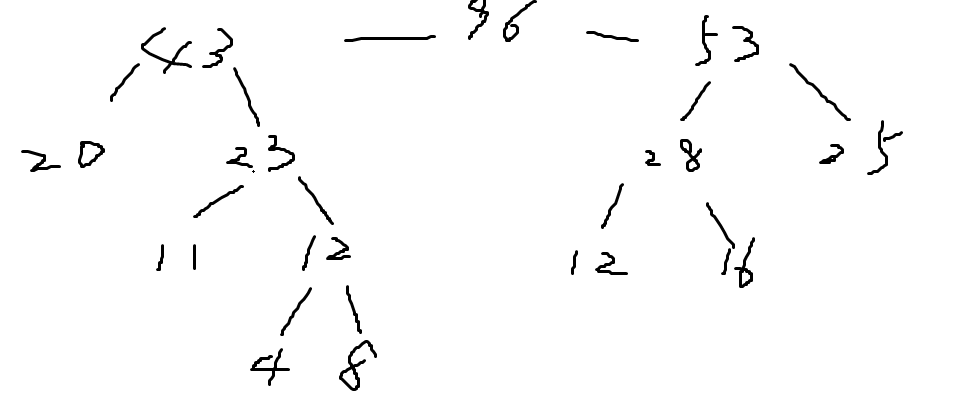
E:0010

F:0100

C:1100

B:1110

A:1000



**Problem 12.2 [30 points]** Cookie assignment: Consider the following problem: You are baby-sitting n children and have m > n cookies to divide between them. You must give each child exactly one cookie (of course, you cannot give the same cookie to two different children). Each child has a greed factor Gi , n≥i≥1 which is the minimum size of a cookie that the child will be content with; and each cookie has a size Sj , m≥j≥1. Your goal is to maximize the number of content children, i.e.., children i assigned a cookie j with Sj≥ Gi .

a. Define pseudo-code a greedy algorithm to solve the cookie assignment problem and define the complexity of your algorithm.

b. Is the algorithm optimal? Prove this, or give a counter-example to show sub-optimality.

**Solution:**

**int cmp(const void \*a, const void \*b){**

**return \*(int \*)a - \*(int \*)b;**

**}**

**int findContentChildren(int\* g, int gSize, int\* s, int sSize) {**

**qsort(g, gSize, sizeof(int), cmp);**

**qsort(s, sSize, sizeof(int), cmp);**

**int cnt = 0;**

**for(int i = 0; i < sSize; i++){**

**if(cnt >= gSize)break;**

**if(s[i] >= g[cnt]){**

**cnt++;**

**continue;**

**}**

**}**

**return cnt;**

**}**

**假设有 m 个孩子，胃口值分别是 g 1​到 g m，有 n 块饼干，尺寸分别是 s 1到 s n，满足 g i≤g i+1和 s j≤s j+1，其中 1≤i<m，1≤j<n。**

**假设在对前 i−1 个孩子分配饼干之后，可以满足第 i 个孩子的胃口的最小的饼干是第 j 块饼干，即 s j是剩下的饼干中满足 g i≤s j的最小值，最优解是将第 j 块饼干分配给第 i 个孩子。如果不这样分配，考虑如下两种情形：**

**如果 i<m 且 g i+1≤s j也成立，则如果将第 j 块饼干分配给第 i+1 个孩子，且还有剩余的饼干，则可以将第 j+1 块饼干分配给第 i 个孩子，分配的结果不会让更多的孩子被满足；**

**如果 j<n，则如果将第 j+1 块饼干分配给第 i 个孩子，当 g i+1≤s j时，可以将第 j 块饼干分配给第 i+1 个孩子，分配的结果不会让更多的孩子被满足；当 g i+1>s j时，第 j 块饼干无法分配给任何孩子，因此剩下的可用的饼干少了一块，因此分配的结果不会让更多的孩子被满足，甚至可能因为少了一块可用的饼干而导致更少的孩子被满足。**

**基于上述分析，可以使用贪心的方法尽可能满足最多数量的孩子。**