**Advanced Algorithms**

**Exercise for Lecture 11,12**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** |  | **Student ID** |  |
| **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: **3459996503@qq.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 |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |  |
|  | 0 |  |  |  |  |  |  |  |  |  |

**Solution:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  | 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↑ |

Hence, the LCS is <B,D,A,B,D>.

**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:**

a)

def make\_change\_rec(target):

if target == 0:

return 0

else:

#try all possible choices for the next bill and return the minimum #the last clause skips values that would result in a negative target return min(1+make\_change\_rec2(target-bill)

for bill in bills

if target-bill >= 0)

b)

Our memoization array is going to be best[i] representing the smallest number of bills that are needed to get the amount i. It follows the recurrence defined above, also stated as:

We can see that elements of best depend only on smaller elements, so evaluating the recurrence starting at 1 suffices. This gives us the solution: def make\_change\_dyn2(target):

best = [0 for n in range(target+1)]

for value in range(1,target+1):

best[value] = min(1+best[value bill] for bill in bills if value-bill >= 0)

return best[target]

This problem takes time since the min operation is done over a constant number of bills.

**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.

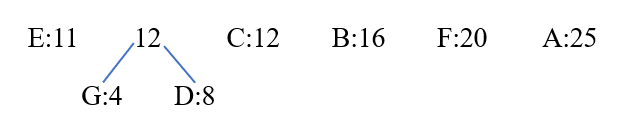
G: 4 D: 8 C: 12 E: 11 B: 16 F: 20 A:25

**Solution:**

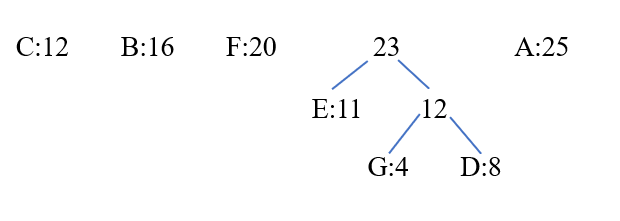
Step 1:



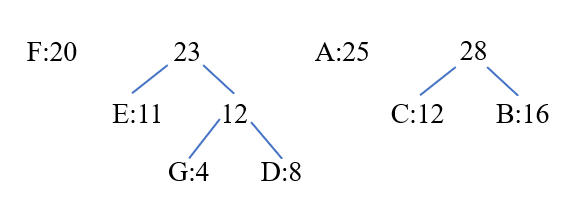
Step 2:



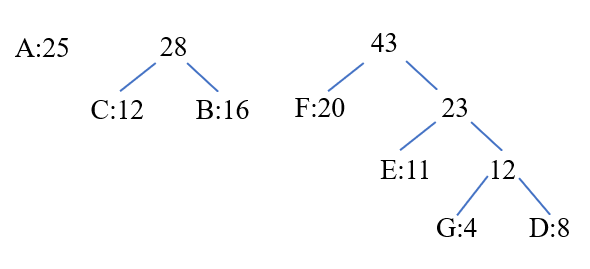
Step 3:



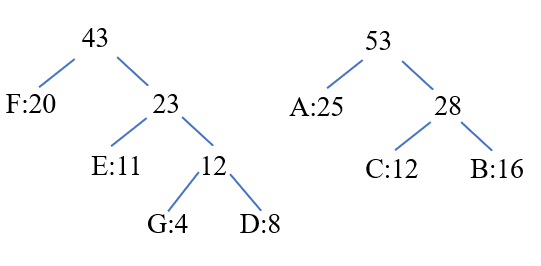
Step 4:



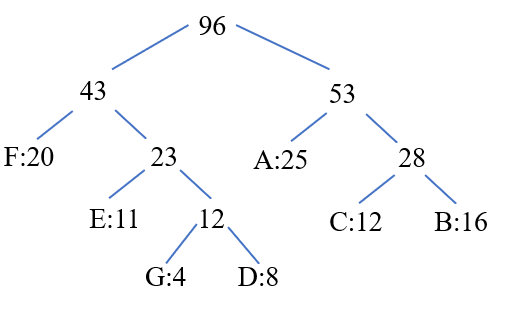
Step 5:



Step 6:



Step 7:



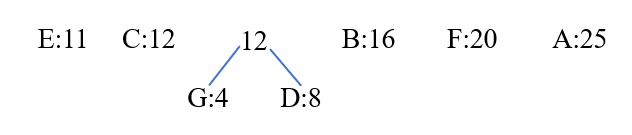
So the coding is F:00, E:010, G:0110, D:0111, A:10, C:110, B:111.

Or:

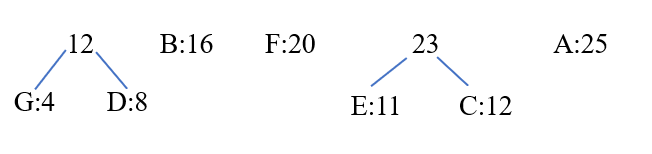
Step 1:



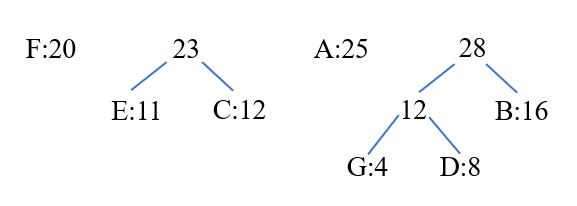
Step 2:



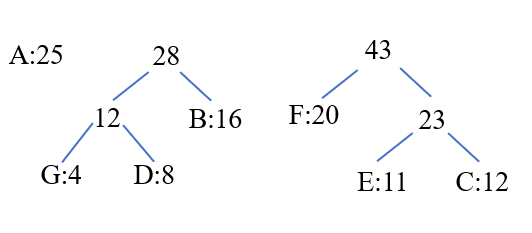
Step 3:



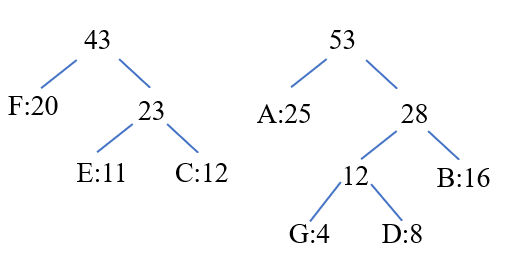
Step 4:



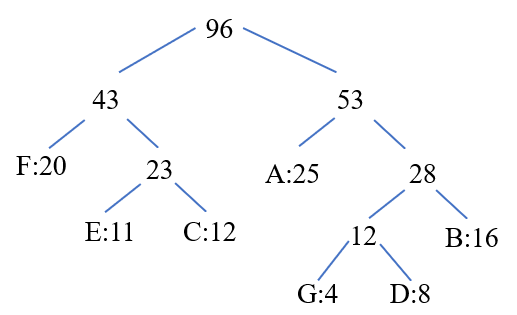
Step 5:



Step 6:



Step 7:



So the coding is F:00, E:010, C:011, A:10, G:1100, D:1101, B:111.

**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:**

a.

Greedy algorithm (children, cookies):

Sort both the children by greed and the cookies by size, in descending order.

C←children; T ← cookies; //sorted lists//

Do for i=1 to n

Assign the largest cookie to the greediest child.

//Repeat until all children have a cookie//

End

b.

The complexity is O(mlogm), which is for sorting the cookies. The main loop has complexity O(n), with n < m.

The algorithm is not optimal.

Counter-example: Consider two children, with greeds 2 and 3, and two cookies of sizes 1 and 2. If we give the cookie of size 2 to the greediest child, we must give the cookie of size 1 to the other child, and neither are content. But if we give the largest cookie to the less greedy child, at least that one small child is content, minimizing malcontentedness.