Romil V. Shah

20008692

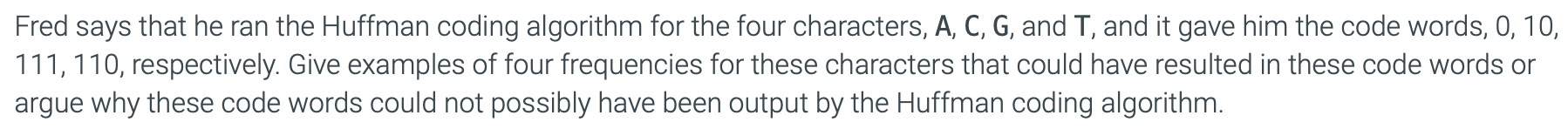
Assignment 5

CS/CPE 600

Prof. Reza Peyrovian

Submission Date: 10/16/2022

Q1. No. 10.5.7



Ans.

Graphical user interface, application, icon

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WORD | Frequency 1 | Frequency 2 | Frequency 3 | Frequency 4 |
| A | 100 | 40 | 130 | 15 |
| C | 45 | 30 | 70 | 7 |
| T | 10 | 10 | 40 | 2 |
| G | 20 | 10 | 30 | 3 |

Example for Frequency 4:

Graphical user interface, application, Teams

Description automatically generated

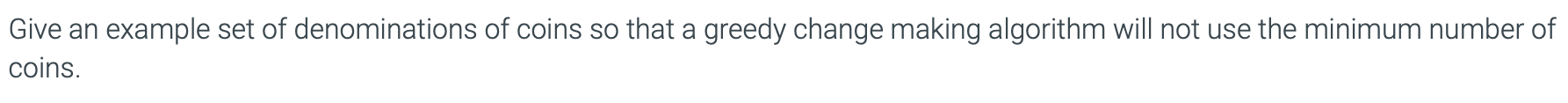
Here, the code of characters, A, C, T, G have different lengths and are assigned in a way that most frequent used character to character which uses fewest number of bits and least used characters to most used. It satisfies the prefix code, that no code is a prefix of any other code.

As a frequency to appear in a word is more so it has been assigned the smallest bit i.e. 0.

As G is the least used word, it is assigned the largest bit, i.e. 111.

This technique helps in achieving the maximum compression.

Q2. No. 10.5.16



Ans.

Greedy Algorithm is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So, the problems where choosing locally optimal also leads to global solution are the best fit for Greedy.

It is an approach for solving a problem by selecting the best option available now. It doesn't worry whether the current best result will bring the overall optimal result. The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach. This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.

So, let us consider the below example:

Suppose value of d1, d2, d3, d4, d5, d6, d7 is 1, 5, 10, 20, 30, 75, 100 and 250.

Now, we calcuclate the minimum numberof coins to get a value of 150 cents.

Using **Greedy Approach**, the minimum coins selected,

150 cents = 100 + 30 + 5 + 1 + 1 + 1 + 1 + 1

But the optimal solution will be,

150 cents = 75 + 75

So, we can say that the greedy algorithm will not use minimum number of coins.

Q3. No. 10.5.27

Graphical user interface, text, application, letter

Description automatically generated

Ans.

Let us consider a string K that is a string of length n and a subsequent string L of length m.

Using a greedy approach, we solve the problem in linear time.

1. We first start with the first character of the message received at the receiver end and finding the same character in the input transmission.
2. Now, we find the second character “O” which is the third character of the first word in the original transmission while ignoring the character which are already matched.
3. We keep calling the above 2 steps until all the characters in the output transmission have been received and matched,
4. Finally, we end the matching if there are no characters left to match with the original string K.

The matching will be done with the original string K of length n and subsequent string L of length m. So, total comparisons made will be O(n+m) which will be the time complexity for this algorithm.

Q4. No. 11.6.1

Text

Description automatically generated

Ans.

Master Theorem is a method for solving divide-and-conquer recurrence equations that is quite general and does not require explicit use of induction to apply correctly. The master theorem is a “cookbook” method for determining the asymptotic characterization of a wide variety of recurrence equations.

Namely, it is used for recurrence equations of the form

Text

Description automatically generated with medium confidence

where d ≥ 1 is an integer constant, a ≥ 1, c > 0, and b > 1 are real constants, and f(n) is a function that is positive for n ≥ d.

1. T(n) = 2T(n/2) + log n

a= 2, b= 2, f(n) = log n

= = n

log n = O(). For

= O(n1-ε) (Using case 1 of master method)

T(n) = θ(

= θ( n)

1. T(n) = 8T(n/2) + n2

a= 8, b= 2, f(n) = n2

= = n3

n2 = O(). For

= O(n3-ε) (Using case 1 of master method)

T(n) = θ(

= θ( n3)

1. T(n) = 16T(n/2) + (n log n)4

a= 16, b= 2, f(n) = (n log n)4

= = n4

(n log n)4 = θ()

T(n) = θ() (Using case 2 of master method)

= θ n4(log n)5

1. T(n) = 7T(n/3) + n

a= 7, b= 3, f(n) = n

= = n1.77

n= O(). For

= O(n1.77-ε) (Using case 1 of master method)

T(n) = θ(

= θ( n1.77)

1. T(n)=9T(n/3) +n3 logn

a= 9, b= 3, f(n) = n3 log n

= = n2

n3 log n = O(). For

Given, a.f(n/b) ≤ δ f(n) for some δ < 1

a.f(n/b) = 9(n/3)3 log(n/3)

= n3/3 log(n/3)

≤ δ f(n). (when δ = 1/3 and n≥ 1)

(Using case 3 of master method)

T(n) = θ (n3 log n)

Q5. No. 11.6.10

Graphical user interface, text, application, email

Description automatically generated

Ans.

**Text, letter

Description automatically generated**

The Stooge-Sort Algorithm says that if the input size, n = 1 or 2, then the algorithm sorts the input immediately. But, for n > 3, we call stooge-sort algorithm recursively into parts each of length ((3(n)) / 4) and in one loop we call stooge-sort 3 times recursively.

Recurrence Relation for the above algorithm will be:

T(n) = 3T ((3(n)) / 4) + cn

Using master method:

a = 3, b = 4, f(n) = n

= = n1.26

n= O(). For

= O(n1.26-ε) (Using case 1 of master method)

T(n) = θ(

= θ( n1.26)

Q6. No. 11.6.17

Text, letter

Description automatically generated

Ans.

The skyline is a collection of rectangular strips, each rectangle has a side on the x-axis, we assume that we are given a set, S = {[a1, b1], [a2, b2], . . . , [an, bn]} of sub- intervals in the interval [0,1], with 0 ≤ ai < bi ≤ 1, for i = 1, 2, … , n, such that there is an associated height, hi, for each interval [ai, bi] in S.

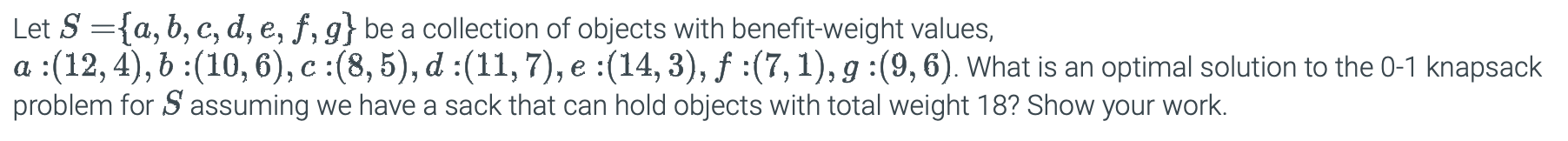
Here, we can use divide and conquer approach to solve the skyline of S.

1. First, we divide the set **S** into two subsets S1 and S2, then we keep dividing further recursively in two halves until a single element is left in the set.
2. Then, we start merging the elements in bottom-up fashion.
3. Merging will be done like merge sort. Start from the first strips of two skyline, compare their ***x*** coordinates. Pick the strip which has smaller ***x*** coordinate and add it to the result.
4. The height of added strip is considered as maximum of current heights from set S1 and S2.

As the algorithm divides the set into two halves every time, it will form the recurrence T(n) = 2 T(n/2) and takes O(log n) time (as the height of the tree). Merging will be done in O(n) time as the work done at the nodes of depth i.

So, the total running time for the above algorithm is O(n log n).

Q7. No. 12.8.5



Ans.

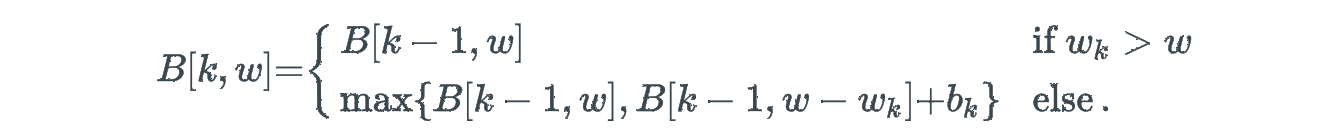
Total Weight object can hold is 18.

Here,

1. a: (12, 4)
2. b: (10, 6)
3. c: (8, 5)
4. d: (11, 7)
5. e: (14, 3)
6. f: (7, 1)
7. g: (9, 6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| B, W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12, 4 | 1 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 10, 6 | 2 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 12 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 |
| 8, 5 | 3 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 20 | 22 | 22 | 22 | 22 | 22 | 30 | 30 | 30 | 30 |
| 11, 7 | 4 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 | 20 | 22 | 23 | 23 | 23 | 23 | 30 | 31 | 33 | 33 |
| 14, 3 | 5 | 0 | 0 | 0 | 14 | 14 | 14 | 14 | 26 | 26 | 26 | 26 | 26 | 34 | 36 | 37 | 37 | 37 | 37 | 44 |
| 7,1 | 6 | 0 | 0 | 0 | 14 | 14 | 14 | 14 | 26 | 26 | 26 | 26 | 26 | 34 | 36 | 37 | 37 | 37 | 37 | 44 |
| 9,6 | 7 | 0 | 0 | 0 | 14 | 14 | 14 | 14 | 26 | 26 | 26 | 26 | 26 | 34 | 36 | 37 | 37 | 37 | 37 | 44 |

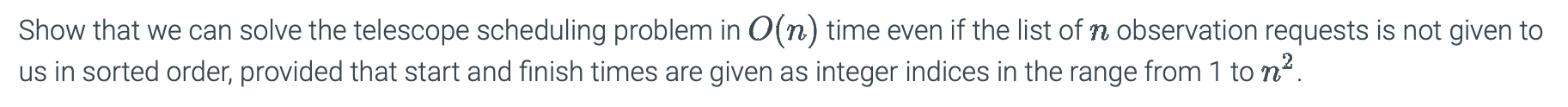
As we fill up the values as per the formula:



We see that the maximum benefit is 44.

We will get the optimal solution from {(12,4), (10,6), (8,5), (14,3)}

Q8. No. 12.8.14



Ans.

Algorithm for telescope scheduling:

Ordering of requests by finish times and an array, P, so that P[i] = pred(i), then we can fill in the array, B

B[0] ← 0  
for i = 1 to n do

B[i] ← max{B[i − 1], B[P [i]] + bi} A

As the list of n observations is not sorted, it will take O(n log n) time to sort and O(n) time for the for loop in the algorithm. So, the total running time of this algorithm will be O(n log n + n).

For large value of n, we consider the value O(n) neglecting log n. Sorting of n observation can be easily done in O(n) time. So, we can compute the entire array B in O(n) time.

For input 1 to n2, we implement dynamic programming to find the optimal solution. But, if the size varies to 2n then, the brute force algorithm will work better.

Q9. No. 12.8.30

Text

Description automatically generated

Ans.

Algorithm to find the longest subsequence of S that is a palindrome:

Input: Palindrome of string S of length [0 … n-1].

Output: The length L[i, j] of longest common sequence of a palindrome S[0 … n-1]

**for** i ← 1 to n **do**

L[i, i] ← 1

**for** i ← 2 to n **do**

L[i, i-1] ← 0

**for** i ← n-1 to 2 **do**

**for** j ← i + 1 to n **do**

**if** L[i] = L[j] **then**

L[i, j] = L [i+1, j-1] + 2

**else**

L[i, j] = max {L[i+1, j], L [i, j-1]}

**return** array L

Here, first, we check the first and the last character of the sequence. This will give us that either both characters are same, or they are not.

If both characters are same, we add 2 to the result and remove both characters and put it in the array L.

If both characters are different, we solve the problem by considering the first character and we compute the remaining subsequence. Then, we repeat the previous step choosing the last character and solve it.

We will record the maximum of both the results.

The algorithm uses 2 dominated nested for loops, with both inner and outer iterating n times. It has an if statement and assignment inside the loop which requires O(1) time.

So, the total running time of the algorithm is O(n2).