CSCI-GA.1170-001 Homework 9

Ankit Sati

TOTAL POINTS

38.5 / 45

QUESTION 1

1 Greedy Ferris Wheel Tycoon 22 / 19

✓ **- 0 pts** *Correct*

✓ **+ 1 pts** *Correct base case in part (d)*

✓ **+ 2 pts** *Partial points to part (d)*

QUESTION 2

2 Who Needs a Nickel? Greedy People! 8.5 / 17

Part-(a)

✓ **- 1 pts** *Incorrect/Missing Definition of DP Matrix*

✓ **- 1 pts** *Incorrect/Missing Base Case i.e. $$DPMatrix[0,j]=\infty$$ (second index denoting amount needed i.e. T) for every $$j>0$$*

✓ **- 1 pts** *Incorrect/Missing Base Case i.e. $$DPMatrix[i,0]=0$$ (first index denoting amount needed i.e. T) for every $$i$$*

Part-(b)

✓ **- 0 pts** *Correct*

Part-(c)

✓ **- 0.5 pts** *New/Updated Input Incorrect/Missing $$5\leq T<10$$ case*

✓ **- 0.5 pts** *New/Updated Input Incorrect/Missing $$10\leq T<25$$ case*

✓ **- 0.5 pts** *New/Updated Input Incorrect/Missing $$25\leq T$$ case*

✓ **- 1 pts** *Improper/Insufficient/Missing Reason for optimality ($$5\leq T<10$$ case). One needs comparison with other solution sets (like choosing all pennies).*

✓ **- 1.5 pts** *Improper/Insufficient/Missing Reason for optimality ($$10\leq T<25$$ case). One needs comparison with other solution sets (i.e. why only using dime is optimal initially).*

You need to argue that the replacement to produce Y does not make it worse. You will argue by saying that the only way to make up for a dime is to use either 10 pennies, 5 pennies and a nickel, or 2 nickels. So at a bare minimum, you will replace TWO coins of a denomination with exactly ONE dime and therefore you have not made matters worse. You need to do a similar analysis for quarters in the next case.

✓ **- 1.5 pts** *Improper/Insufficient/Missing Reason for optimality ($$25\leq T$$ case). One needs comparison with other solution sets (i.e. why only using quarter is optimal initially).*

QUESTION 3

3 Fibonacci Meets Huffman 8 / 9

✓ **- 1 pts** *part c) partially incorrect*

Homework 9-

Problem 9-1

(a) Input I - $c, w_1, \ldots, w_n$

Cost (Strategy) = number of cars used.

Claim: Cost (a) $\leq$ Cost (z)

where. G - Greedy strategy

Z $\in$ only optimal strategy.

Proof.

① Base Case. $n=1$, There is just one rider. Hence only one car will be occupied in both G and Z. .{ Also, $w_1 \leq c$ }

Thus Cost (a) = Cost (z)

② Maintenence: Assuming it is true for riders $< n$, we will show it is true for riders $= n$.

③ For $c_1 = \langle g_1, \ldots, g_n \rangle$ & $Z = \langle z_1, \ldots z_k \rangle$.

g_i and z_i are the indices of the last person stated in car i.

④ If. $z_1 = g_1$, then we are done, following the induction hypothesis. We remove. inputs still w_2. or w_{g_2} and we are left with a smaller input on which we can apply induction hypothesis

Swapping Values:.

When Z does not contain g_1. Then we transform Z to a. strategy Y which makes the local greedy swap. by swapping z_1 for g_1.

Strategy Y - Fill the first car as long as. total weight $\leq c$.. Then fill each car with only 1 driver.

This ensures that z_1 of $Y = g$.

Now, we show that nothing else is affected by the swap.

→ We know that there is ~~nothing else affected by this swap~~. no-line cutting among riders will only sit in car when it's their turn. Hence no other car riders conflict with z_1, and they all come after z_1 riders are seated

→ Since $g_1 = z_1$ of Y. This follows from greedy Algorithms & strategy Y defination.

→ We can also say that none of $z_2 \ldots z_n$ conflict with g. Therefore no other changes need to be made due to the swap.

→ Since Y agrees with 1st step of G.

Therefore we can conclude that $cost(Y) \leq cost(z) \ldots (i)$

Recursion Step.

Consider the smaller input I' which has first w_g, drivers removed from I.

Therefore. $I' = c, w_{g_{1+i}}, \ldots w_n$.

Consider G' & Y' on remainder of drivers. They both have the same input I' as the same number of riders were removed. in their respective 1st step.

G' is merely. G on I'.

With the help of induction, $cost(G') \leq cost(Y')$ ——(ii)

$g_1 + cost(G') \leq g_1 + cost(Y)$. — adding g on both sides

$\therefore cost(G) \leq cost(Y)$

$\Rightarrow cost(G) \leq cost(z) \ldots$ from (i)

This concludes the proof //

(b) $F_j(2)$ = Index. of car in which j^{th} person is seated

Claim : $F_j(G_1) \leq F_j(2)$

Proof..

→ Base Case.($j=1$).

G_1 will choose the 1^{st} car itself. according to its defination and hence $F_j(G_1) \leq F_j(2)$. where 2 is any. optimal strategy.

∴ The base case is proved.

→ Maintenance Induction

Assuming it is true for. $j=m$, we need to prove for $j=m+1$.

Now, from Induction hypothesis, we have. $F_m(G_1) \leq F_m(2)$

Since G_1 is greedy., $m+1^{th}$ person will either be in the same car as. m or the next one.

$\therefore F_m(G_1) \leq F_{m+1}(G_1)$ —(1)

And since 2 include all optimal solution with G_1 taking the 1^{st}. possible car, $F_m(2)$ can be greater than or equal. to $F_{m+1}(G_1)$ i.e

$F_{m+i}(G_1) \leq F_m(2)$ —(2).

Now, $F_m(2) \leq F_{m+1}(2)$ —(3) (A_2 defined)

∴ From (2) & (3)

$F_{m+1}(G_1) \leq F_{m+1}(2)$.

⇒ Induction proves Correct.

⇒ Hence proved//.

(c) The cost of optimal solution is l, which means that the n people can be accomodated in l cars.

Now, if all the cars will be filled fully in the worst case then the weight of all people equal to the total weight of l cars.

So, the weight sum of all people will be less than or equal to lc.

So. $$\boxed{\sum_{i=1}^{n} w_i \leq lc} \quad \text{——— (i)}$$

The cost of G_1 strategy is k,

Now, we can say that the total sum of weight of all people will be greater than half the cost of this strategy times the weight of each car.

$$\text{So, } \sum_{i=1}^{n} w_i > \left\lfloor \frac{k}{2} \right\rfloor c \quad \text{—(ii)}$$

On combining (i) and (ii), we get.

$$\left\lfloor \frac{k}{2} \right\rfloor c < \sum_{i=1}^{n} w_i \leq lc$$

$$\left\lfloor \frac{k}{2} \right\rfloor c < lc$$

$$\Rightarrow \boxed{\left\lfloor \frac{k}{2} \right\rfloor < l}$$

(c) Scenarios when G_2 does as good as G_1.

$w_1 = 1$
$w_2 = 3$
$w_3 = 6$.
$w_4 = 7$.

$G_2 = \{w_1, w_2, w_3\}, \{w_4\}$
$G_1 = \{w_1, w_2, w_3\}, \{w_4\}$.

$Cost(G_2) = 2$.
$Cost(G_1) = 2$.

Scenarios when G_2 does better than G_1

$w_1 = 1$
$w_2 = 7$
$w_3 = 3$
$w_4 = 6$
$w_5 = 3$.

Through the above, we can conclude that.

$G_2 = \{w_1, w_3, w_4\}, \{w_2, w_5\}$
$G_1 = \{w_1, w_2\}, \{w_3, w_4\}, \{w_5\}$

$Cost(G_2) = 2$.
$Cost(G_1) = 3$

Hence it is proved.

As G_2 maintain second car as well as the first car it can need both if it is safe and can back to 1st car.

In the first scenario, the weight are increasing and here 1st car is filled. in both cases and only the maximum allowed car.

(d) Claim $Cost(G_2) \leq Cost(G_1)$

Base Case $n=1$.

There is just one element.

So $Cost(G_2) = Cost(G_1)$.

Claim is true.

Induction → Assuming true for orders $< n$, let's prove for orders $= n$.

Local Swap:

We swap r_1 with g from G_2.

Nothing changes in the number of orders queued.

All the orders before g_1 are seated in G_1 & no conflict occurs.

Recursive Step:

G_2 may contain orders which are still in queue as it has r cars.

For Induction

Our orders will be queued in such a way that $G_2 \leq g_1$ does not conflict $r_2 \ldots r_n$.

Here we can conclude induction

$\therefore Cost(G_2') \leq Cost(G_1')$

$\therefore Cost(G_2') \leq g + Cost(G_1')$

$Cost(G_2) \leq Cost(G_1)$ Hence Proved.

1 Greedy Ferris Wheel Tycoon **22 / 19**

✓ **- 0 pts** *Correct*

✓ **+ 1 pts** *Correct base case in part (d)*

✓ **+ 2 pts** *Partial points to part (d)*

Problem 9-2

a) Algorithm:

```
func (T, q, d, n, p):
    if T == 0:
        return 1
    if q == 0 and d == 0 and n == 0 and p == 0
        return 0
    if dp[T][q][d][n][p] != -1
        return dp[T][q][d][n][p]
    if q != 0 : dp[T][q][d][n][p] =
        min(dp[T][q][d][n][p], 1 + func(T-25, q-1, d, n, p)
    if d != 0 : dp[T][q][d][n][p] =
        min(dp[T][q][d][n][p], 1 + func(T-10, q, d-1, n, p)
    if n != 0 : dp[T][q][d][n][p] =
        min(dp[T][q][d][n][p], 1 + func(T-5, q, d, n-1, p)
    if p != 0 :
        dp[T][q][d][n][p] =
        min(dp[T][q][d][n][p], 1 + func(T-1, q, d, n, p-1)
    return dp[T][q][d][n][p]
```

Time Complexity = $O(T(q+d+n+p))$

(a) Let

$q = 1$

$d = 3$ $\quad T = 30$

$n = 0$

$P = 10$

= Invariant

⇒ This is one the best example where greedy will fail to give the Soln.

⇒ If we take the greedy algorithm, then there will be no solution.

⇒ whereas there exists a solution which is to use 3 dimes and 1 penny.

Proof :-

We can conclude, According to greedy, we will select q and will be left with 5. Now we only have 2 pennies to use. Hence this solution will return $c = 6$. Whereas using 3 dimes we can reach $T = 30$.

Hence Proved.

(c) if $T < 5$, it will be optimal to use all the pennies.

(1) If $5 \leq T < 10$, it will be optimal to use a nickel and rest pennies.

(2) If $10 < T < 25$, it will be optimal to use dimes and rest pennies or nickels

(3) Similarly, for $25 \leq T$, it will be optimal to use quaters first and then excess for the remaining value of T accordingly.

$\Rightarrow$ Since we are making a greedy choice every time, so here we can conclude that greedy algorithm works in the case when the coins are embounded.

2 Who Needs a Nickel? Greedy People! 8.5 / 17

Part-(a)

✓ **- 1 pts** *Incorrect/Missing Definition of DP Matrix*

✓ **- 1 pts** *Incorrect/Missing Base Case i.e. $$DPMatrix[0,j]=\infty$$ (second index denoting amount needed i.e. T) for every $$j>0$$*

✓ **- 1 pts** *Incorrect/Missing Base Case i.e. $$DPMatrix[i,0]=0$$ (first index denoting amount needed i.e. T) for every $$i$$*

Part-(b)

✓ **- 0 pts** *Correct*

Part-(c)

✓ **- 0.5 pts** *New/Updated Input Incorrect/Missing $$5\leq T<10$$ case*

✓ **- 0.5 pts** *New/Updated Input Incorrect/Missing $$10\leq T<25$$ case*

✓ **- 0.5 pts** *New/Updated Input Incorrect/Missing $$25\leq T$$ case*

✓ **- 1 pts** *Improper/Insufficient/Missing Reason for optimality ($$5\leq T<10$$ case). One needs comparison with other solution sets (like choosing all pennies).*

✓ **- 1.5 pts** *Improper/Insufficient/Missing Reason for optimality ($$10\leq T<25$$ case). One needs comparison with other solution sets (i.e. why only using dime is optimal initially).*

You need to argue that the replacement to produce Y does not make it worse. You will argue by saying that the only way to make up for a dime is to use either 10 pennies, 5 pennies and a nickel, or 2 nickels. So at a bare minimum, you will replace TWO coins of a denomination with exactly ONE dime and therefore you have not made matters worse. You need to do a similar analysis for quarters in the next case.

✓ **- 1.5 pts** *Improper/Insufficient/Missing Reason for optimality ($$25\leq T$$ case). One needs comparison with other solution sets (i.e. why only using quarter is optimal initially).*

Problem 3 (q-3)

(a) Fibonacci values.

$f_0 = 1$

$f_1 = 1$

$f_2 = 2$

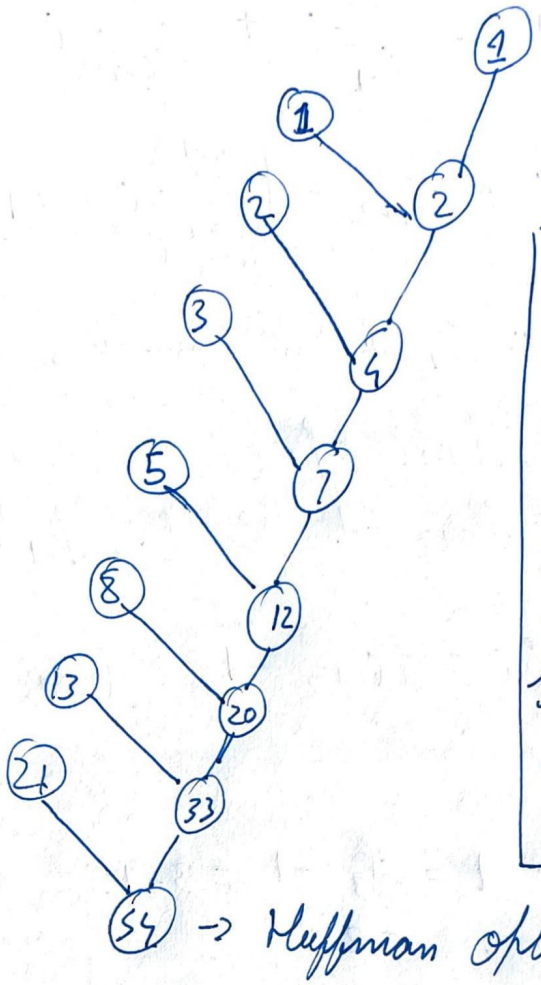
$f_3 = 3$

$f_4 = 5$

$f_5 = 8$

$f_6 = 13$

$f_6 = 21$.



Huffman Code Sequence

$f_0 \rightarrow 0000000$

$f_1 \rightarrow 0000001$

$f_2 \rightarrow 000001$

$f_3 \rightarrow 00001$

$f_4 \rightarrow 0001$

$f_5 \rightarrow 001$

$f_6 \rightarrow 01$

$f_7 \Rightarrow 01$.

54 → Huffman optimal code.

$= 54$

(b) $S_1 = 2 = f_0 + f_1$

$S_i = S_{i-1} + f_i = f_0 + f_1 + \cdots + f_i$ —— (1)

To prove: $S_i = f_{i+2} - 1$, $i \geq 1$

Proof →

Assume that: $F_2 = F_1 + F_0$

$f_3 = f_2 + f_1$

$f_4 = f_3 + f_2$.

$\vdots$

$f_{(n)} = f_{n+1} + f_{n-2}$

$f_{(n+1)} = f_n + f_{n-1}$

$f_{(n+2)} = f_{n+1} + f_{n+2}$.

Adding all of them,

$$\sum_{i=2}^{n+2} f(i) = \sum_{i=1}^{n+1} f_i + \sum_{i=0}^{n} f_i$$

$$= \sum_{i=2}^{n+1} f_i + f_{n+2} = \left(\sum_{i=2}^{n+1} f_i + f_1\right) + \sum_{i=0}^{n} f_i$$

$$= f_{n+2} - f_1 + S_n \quad \text{— From equation 1}$$

$$= S_n = f_{n+2} - f_1$$

$$= S_n = f_{n+2} - 1. \qquad (\because f_1 = 1)$$

or

$$S_i = f_{i+2} - 1 \qquad (\text{for } i > 1)$$

For $i = 1$,

$$S_1 = f_{1+2} - 1 = f_3 - 1 = 3 - 1.$$
$$= 2$$

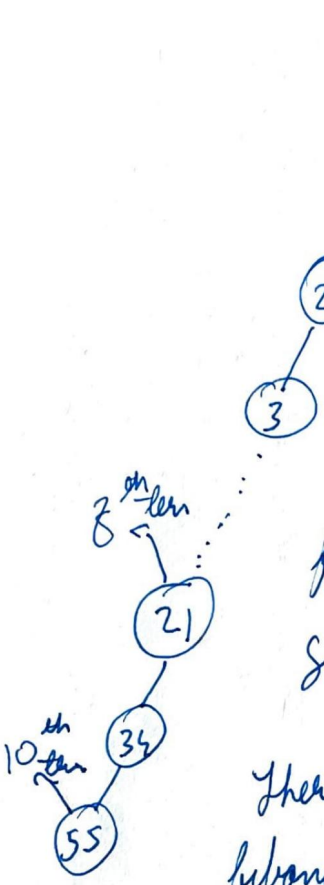
Since,

$S_i = f_{i+2} - 1$ hold true for.

$i \geq 1$

Therefore this is proved.

(c) In our solution to part (a), we notice that the optimal huffman code is actually the next to next term after 8 term - 1.



⇒ optimal huffman code is actually the sum of all the fibonacci numbers in the frequencies.

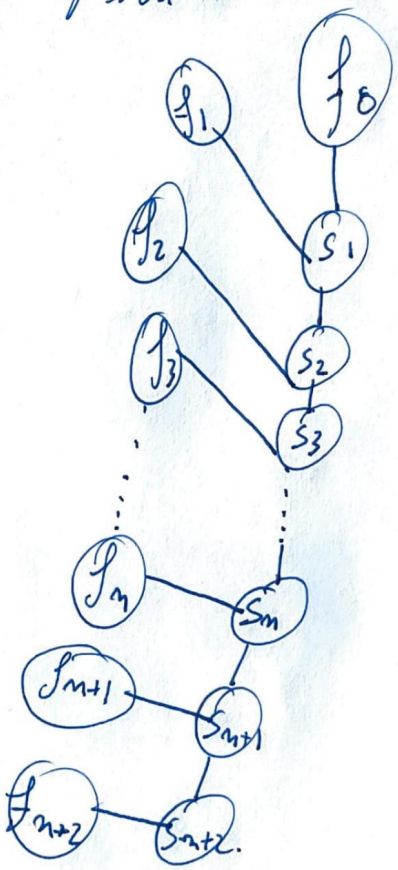
As we can see that the tree atleast level are always merged. and that the sum is always one of the least for the next merging. In this way. each of the fibonacci gets added.

Therefore we can say that, optimal huffman code of n fibonacci numbers is equals to sum of those n numbers.

Also, part a tells us that

$$S_n = f_{n+2} - 1$$

⇒ The tree we can see that optimal huffman code for n fib numbers, is S_n which is actually $(f_{n+2} - 1)$



⇒ Thus correctness is justified

3 Fibonacci Meets Huffman **8 / 9**

✓ **- 1 pts** *part c) partially incorrect*