CSPB 3104 - Park - Algorithms

Started on	Tuesday, 2 April 2024, 1:49 PM
State	Finished
Completed on	Tuesday, 2 April 2024, 2:45 PM
Time taken	55 mins 42 secs
Marks	14.00/17.00
Grade	**8.24** out of 10.00 (**82**%)

Question **1**

Complete

Mark 5.00 out of 6.00

A transport ship can accommodate up to 57 tonnes of goods. The following is a table of items that can be shipped along with weights the revenue.

Item	Weight	Revenue (1000s dollars)	Revenue/Ton
Truck	2 Ton	$ 40	20
Machine Part	4 Ton	$ 50	12.5
Prefab Concrete	8 Ton	$ 110	13.75
Medical Equipment	2 Ton	$ 30	15

We wish to maximize the total revenue obtained subject to the constraint that the total weight must remain less than or equal to the weight limit.

Let maxRevenue(W) denote the maximum revenue that can be earned starting with a weight limit of W tons.

The problem below has 6 parts (A)-(F).

(A) What is the value of maxRevenue(W) if W = 0?

If the weight limit W is set to zero, then the ship cannot technically carry any items. So in this case, if the ship is not able to carry any items, it won't have any revenue. Therefore **maxRevenue(0) = 0.**

(B) What is the value of maxRevenue(W) if W < 0?

This is the same idea as part **A,** if our ship has a weight limit that is under 0, then it is not going to be able to carry any items. In fact, the transport ship would be costing the company money in this case and therefore we would then have to say that **maxRevenue(W < 0) = -inf.** I say negative infinity as a baseline (we don't know the operating cost of the ship and therefore it could be a lot, or an infinite cost to the ship to operate in this case).

(C) Write down a recurrence for maxRevenue(W). You may fill in the following template.

maxRevenue(W) = MAX(0 , 40 + maxRevenue(W-2), 50 + maxRevenue(W-4), 110 + maxRevenue(W-8), 30 + maxRevenue(W-2))

(D) What is the "root" call to maxRevenue needed to solve the overall problem above?

The root call to maxRevenue is going to check if the weight of object(s) that are being included on the ship exceed the remaining weight limit of the ship so that the ships carrying weight is not above capacity.

(E) Let us formulate a greedy algorithm that works as follows:

```
W = 57
Repeat until no more choices exist.
   Eliminate all choices whose weights are > W
   Choose the item with largest revenue/ton. Let Wj be its unit weight.
   Choose W//Wj number of items. Note that // is integer division.
    W = W − (W // Wj) ∗ Wj
```

What are the choices made by the greedy algorithm for the problem instance given here?

The largest revenue per ton for this ship is the Truck, then Medical Equipment, Prefab Concrete, and finally the machine part. After selecting the Truck, W gets updated where the final value of W is no longer greater than any of the remaining choices. So all other choices get eliminated and the algorithm finishes execution.

(F) We wish to design a memo table for memoizing the recurrence. How many elements will the memo table have? Write down a pseudocode for filling up this memo table.

The memo table for this solution has to be two dimensional, it would look something similar to this:

memo[weight][revenue] = max of including / excluding choices in what to store on the ship

This is then checked at the beginning of each recursive call so that we aren't calculating the same value more than once and thus reducing runtime of the over all algorithm.

Comment:

(A) What is the value of maxRevenue(W) if W = 0?

0

(B) What is the value of maxRevenue(W) if W < 0?

negative infinity

(C) Write down a recurrence for maxRevenue(W). You may fill in the following template.

maxRevenue(W) = MAX(0 , revenue_1 + maxRevenue(W-weight_1), revenue_2 + maxRevenue(W-weight_2), revenue_3 + maxRevenue(W-weight_3), revenue_4 + maxRevenue(W-weight_4))

(D) What is the "root" call to maxRevenue needed to solve the overall problem above?

maxRevenue(57)

(E) What are the choices made by the greedy algorithm for the problem instance given here?

28 * 20 (28 Trucks), giving 560

1 remaining ton is wasted, giving 0

(F) We wish to design a memo table for memoizing the recurrence. How many elements will the memo table have? Write down a pseudocode for filling up this memo table.

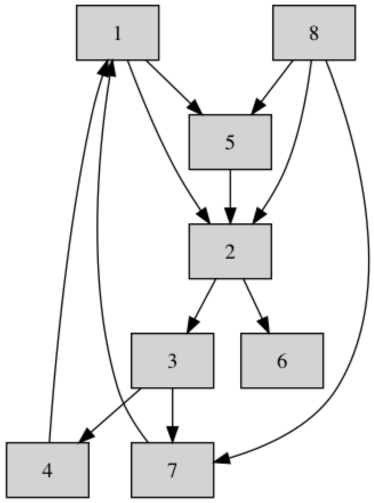
```
initialize a table of size W+1
table[0] = 0
for j = 1 to W:
   q = - infty
   for i = 1 to size(wieghts):
      if j - weights[i] >= 0:
         if revenue[i] + table[j - weights[i] ] > q:
            q = revenue[i] + table[j - weights[i]]
   table[j] = q
```

Question **2**

Partially correct

Mark 5.00 out of 7.00

Consider a directed graph G below.



We perform a DFS visit starting from the node id 8.

The discovery time of 8 is taken to be 1.

Whenever a node is visited, its adjacent nodes are considered in increasing order of their node ids.

Write down the ID of the parent node for the following nodes in the resulting DFS tree. Your answer in each case should be a number. Make sure that you do not type unnecessary spaces in the answer box.

(A) Node 1

4 ✔

(B) Node 5

8 ✖

(C) Node 2

8 ✔

(D) Node 6

2 ✔

(E) What is the classification of the edge (1,2) in the DFS tree?

- Tree Edge
- Back Edge ✔ Correct
- Cross Edge
- Forward Edge

Mark 1.00 out of 1.00

The correct answer is: Back Edge

(F) What is the classification of the edge (8,5) in the DFS tree?

- Tree Edge ✖ Wrong

- ○ Back Edge
- ○ Cross Edge
- ○ Forward Edge

Mark 0.00 out of 1.00

The correct answer is: Forward Edge

(G) What is the classification of the edge (7,1) in the DFS tree?

- ○ Tree Edge
- ○ Back Edge
- ◉ Cross Edge ✔ Correct
- ○ Forward Edge

Mark 1.00 out of 1.00

The correct answer is: Cross Edge

Question **3**

Correct

Mark 4.00 out of 4.00

Consider the following recursive procedure for calculating the Fibonacci numbers.

```
def fib(n):
   if  n <= 1:
        return 1
   return fib(n-1) + fib(n-2)
```

Consider the memoized version:

```
def fib(n):
    tbl = [1] * (n+1) # Create a table of n+1 entries
    for i in range(2, n+1):
          tbl[i] = tbl[i-1] + tbl[i-2]
    return tbl[n]
```

Select all the correct answers from the list below.

Select one or more:

- [x] a. The memoized procedure runs in time polynomial in the number of bits used to represent n. ✖
- [ ] b. The running time of the recursive procedure is $\Theta(n)$.
- [ ] c. The memoized procedure runs in time exponential in the number of bits used to represent n.
- [ ] d. The recursive procedure requires just as much space (upto constant factor) when compared with the memoized procedure.
- [x] e. The two procedures compute the same result for all natural numbers n. ✔
- [ ] f. The recursive procedure uses constant amount of space.
- [x] g. The recursive procedure uses up space that is exponential in n. ✖
- [x] h. The running time of the recursive procedure is $O(2^n)$. ✔

Your answer is correct.

The correct answers are: The running time of the recursive procedure is $O(2^n)$.

, The two procedures compute the same result for all natural numbers n., The recursive procedure requires just as much space (upto constant factor) when compared with the memoized procedure.,
The memoized procedure runs in time exponential in the number of bits used to represent n.

Comment: