INSTITUTE OF TECHNOLOGY

TCSS 343 - Assignment 4

Version 1.0 – Zhihao Yang

May 29, 2018

1 GUIDELINES

Homework should be electronically submitted to the instructor by midnight on the due date. A submission link is provided on the course Canvas Page. The submitted document should be typeset using any common software and submitted as a PDF. We strongly recommend using LaTeX to prepare your solution. You could use any LaTeX tools such as Overleaf, ShareLatex, TexShop etc. Scans of handwritten/hand drawn solutions are acceptable, but you will be docked 1 point per problem if the handwriting is unclear.

Each problem is worth a total of 10 points. Solutions receiving 10 points must be correct (no errors or omissions), clear (stated in a precise and concise way), and have a well organized presentation. Show your work as partial points will be awarded to rough solutions or solutions that make partial progress toward a correct solution.

Remember to cite all sources you use other than the text, course material or your notes.

2 PROBLEMS

2.1 UNDERSTAND

1. (4 points) Let F_i be the i^{th} Fibonacci number such that $F_0 = 0$ and $F_1 = 1$. Use the dynamic programming solution to compute F_{20}. Show your work.

   Answer:

   $$Fib = [0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765]$$

   So F_{20} is the twenty-first term of Fib. Hence $F_{20} = 6765$.

2. (6 points) Consider the Job Selection problem from lecture. Compute the optimal set of jobs and the maximum earnings for the following input.

   $$P = [5,9,12,7,5,13,7,5,4,9,8,7,5,8,4,3,5,10,4,6,8,12,5,6,3,7,16,2,2,16]$$

   Answer: The corresponding optimal set J:

   $J = [5,9,17,17,22,30,30,35,35,44,44,51,51,59,59,62,64,72,72,78,80,90,90,96,96,103,112,112,114,128]$

   We also have this set:

   $$[1,1,1,0,1,1,0,1,0,1,0,1,0,1,0,1,1,1,0,1,1,1,0,1,0,1,1,0,1,1]$$

   The maximal earnings will be:

   $$5+12+13+5+9+7+8+3+10+6+12+6+16+16 = 128$$

   Days: $1,3,6,8,10,12,14,16,18,20,22,24,27,30$

3. On Venus the Venusians use coins of these values $\{1,6,10,19\}$.

   a) (4 point) Select a value v. Make change for v with Venusian coins by always selecting the largest coin. Show that this is not the minimum number of coins possible to make change for v. (Hint: This won't work for all v so select your v carefully.)

   Answer:

   Let $v = 12$, if we used brute force algorithm, we give select three coins: 10 + 1 + 1.

   However, we can select two coins: 6+6. So using brute force algorithm is not the minimum number of coins in this situation.

   b) (6 points) Use the algorithm from class to compute the minimum number of coins needed to make change for 42 on Venus. State what coins are used to satisfy this minimum.

   Answer:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1	2	3	4	5	1	2	3	4	1	2	2	3	4
15	5	2	3	3	1	2	3	3	4	4	2	3	4	4	2
30	3	3	4	5	5	3	4	4	2	3	4	4	5		

So the minimum number of coins needed should be 5: 19 + 10 + 6 + 6 + 1.

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.

2.2 EXPLORE

Consider the Job Selection problem from lecture. You will modify the solution to this problem to work for a related problem. In lecture we had the rule that if you work one day you can't work the next (or previous). In the modified problem you are allowed to work two days in a row but not three days in a row (so if you work 2 days in a row you can't work the day before these 2 days or the day after).

1. (2 point) Express the modified problem formally with input and output conditions.

   Answer:

   JobSelection

   Input: $P[1 \dots n]$, a list of payouts

   Output: m, the maximal earnings and $A \subseteq \{1 \dots n\}$ such that $\sum_{n \in A} P[i] = m$ and if day $i \in A$ and $i+1 \in A$, then $i+2 \notin A$.

2. (4 points) State a self-reduction for your problem. Use the self-reduction from lecture as inspiration.

   Answer:

$$JS(n) = \begin{cases} \text{P[1]} & \text{if } n = 1 \\ \max\{\text{p[1], P[2]}\} & \text{if } n = 2 \\ \max\{\text{JS[n-1], P[n]+JS[n-2], P[n]+P[n-1]+JS[n-3] }\} & \text{if } n \geq 3 \end{cases}$$

3. (4 points) State a dynamic programming algorithm based off of your self reduction that computes the maximum earnings.

   Answer:

   JobSelection($P[1 \dots n]$)

   Let J be 1-d array of size n;

   $J[1] = P[1]$;

   $J[2] = max\{P[1], P[2]\}$;

   for $i = 3$ to n

   $J[i] = max\{J[n-1], P[n] + J[n-2], P[n] + P[n-1] + J[n-3]\}$;

endFor

return $J[n]$;

End JobSelection.

4. (4 points) Design a function that recovers the days that must be worked to achieve the maximum earnings.

   Answer:

   Recover(n)

   If J[n] == J[n-1]

   return Recover(n-1);

   If J[n] == J[n-2] + P[n]

   return Recover(n-2) and add $\{n\}$ into the day list;

   If J[n] == J[n-3] + P[n] + P[n-1]

   return Recover(n-3) and add $\{n, n-1\}$ into the day list;

   End Recover.

5. (2 point) Compute the maximum earnings and the days to work for this input (same as above).

   $$P = [5,9,12,7,5,13,7,5,4,9,8,7,5,8,4,3,5,10,4,6,8,12,5,6,3,7,16,2,2,16]$$

   Answer:

   J= [5,14,21,24,26,39,44,44,48,57,61,64,69,74,76,77,82,91,91,97, 105,111,

   114,117,120,124,140,140,142,158]

   Days: [2,3,5,6,8,10,11,13,14,16,18,19,21,22,24,26,27,29,30]

6. (4 points) What are the worst case time and space requirements of your complete solution?

   Answer:

   Runnint time should be $O(n)$.

   Space is $O(n)$, but it can be improved to $O(1)$ since we don't need all the previous numbers.

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.

2.3 EXPAND

Consider the Subset Sum problem from lecture. You will modify the solution to this problem to work for a related problem. In the modified problem we will count the number of different subsets that sum to our target number.

1. (2 point) Express the modified problem formally with input and output conditions.

   Answer:

   Subset Sum(SS)

   Input: $S[1 \dots n]$, a set of n positive integers and t, a positive target integer.

   Output: k, the number of different subsets of S that sum to the target number, t and A, a set of subsets which each subset is $A_i \subseteq \{1 \dots n\}$ such that $t = \sum_{i \in A_i} S[i]$.

2. (4 points) State a self-reduction for your problem.

   Answer:

$$SS(n,t) = \begin{cases} 1 & \text{if } t = 0 \\ 0 & \text{if } t > 0, n = 0 \\ \text{SS(n-1, t) + SS(n-1, t-S[n])} & \text{if } t > 0, n > 0 \end{cases}$$

3. (4 points) State a dynamic programming algorithm based off of your self reduction that computes the number of subsets that sum to our target number.

   Answer:

```
SubsetSum(S[1...n], t)
    Let M be a 2-D array of size (n+1)(t+1);
    M[0][0] = 1;
    for j = 1 to t
        M[1][j] = 0;
    endFor
    for i = 1 to n
        M[i][1] = 0;
    endFor
    for i = 1 to n
        for j = 1 to t
            M[i][j] = M[i-1][j] + M[i-1][j-S[i]];
        endFor
    endFor
    return M[n][t]
End SubsetSum.
```

4. (4 points) Design a function that recovers **every subset** that sums to our target number. (Hint: Recursion will be helpful.)

   Answer:

   Recover(n, t)

   if t <= 0, return an empty set of empty sets;

   if n <= 0, t > 0, return an empty set;

   if M[n][t] = 0, return an empty set;

   Creat a new set K = Recover(n-1, t);

   Creat a new set H = Recover(n-1, t-S[n]);

   for each subset, a, in H

   Add the current index, n, to each set, a;

   endFor

   Let G be the union of K and H;

   return G

   End Recover.

5. (2 point) Use your algorithm to compute and recover every subset that sums to the target number $t = 6$ for the multiset S.

$$S = \{1,2,1,3,1,4,1,5\}$$

   Answer:

	-	1	2	1	3	1	4	1	5
0	1	1	1	1	1	1	1	1	1
1	0	1	1	2	2	3	3	4	4
2	0	0	1	2	2	4	4	7	7
3	0	0	1	2	3	5	5	9	9
4	0	0	0	1	3	6	7	12	12
5	0	0	0	0	2	5	8	15	16
6	0	0	0	0	2	4	8	16	20

   $\{1,5\},\{1,1,4\},\{1,1,1,3\},\{1,1,1,1,2\},\{1,2,3\},\{2,4\}$

   I omit some duplicates. For example, $\{1,1,1,3\}$ and $\{1,1,3,1\}$ are the same.

6. (4 points) What are the worst case time and space requirements of your complete solution?

   Answer:

Running time should be $O(n \cdot t)$; Recover takes about $O(n \cdot k)$ where k is the number of different subsets that sum to the target number.

Space is $O(n \cdot t)$ as well since we have this 2d array; Recover also takes about $O(n \cdot k)$ where k is the number of different subsets that sum to the target number.

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.

2.4 CHALLENGE

You and your friends are driving to Tijuana for spring break. You are saving your money for the trip and so you want to minimize the cost of gas on the way. In order to minimize your gas costs you and your friends have compiled the following information.

First your car can reliably travel m miles on a tank of gas (but no further). One of your friends has mined gas-station data off the web and has plotted every gas station along your route along with the price of gas at that gas station. Specifically they have created a list of $n+1$ gas station prices from closest to furthest and a list of n distances between two adjacent gas stations. Tacoma is gas station number 0 and Tijuana is gas station number n. For convenience they have converted the cost of gas into price per mile traveled in your car. In addition the distance in miles between two adjacent gas-stations has also been calculated. You will begin your journey with a full tank of gas and when you get to Tijuana you **will fill up for the return trip**.

You need to determine which gas stations to stop at to minimize the cost of gas on your trip.

1. (1 point) Express this problem formally with input and output conditions.

   Answer:

   Trip Gas Cost:

   Input: $G[0 \ldots n]$, a list of gas station prices from closest to furthest and, $D[1 \ldots n]$, a list of distances between two adjacent gas station, and m miles that you can travl with a tank of gas.

   Output: c, the minimum cost of gas for the trip, and A, a subset of $[1 \ldots n]$, which for any two adjacent elements a, b in A, $d(a,b) <= m$ and also $c = \sum_{i \in A}^{n} d(i,j)G[i]$ such that j is the next gas station after i.

2. (2 points) State a self-reduction for your problem.

   Answer:

$$TGC(n) = \begin{cases} 0 & \text{if } n = 0 \\ \min\{\text{d(i,j)G[i]}\} + \text{TGC(n)} & \text{if } n > 0 \end{cases}$$

3. (2 points) State a dynamic programming algorithm based off of your self reduction that computes the minimum gas cost.

4. (2 points) Design a function that recovers the gas stations that should be stopped at to achieve the minimum cost.

5. (1 point) Use your algorithm to compute minimum cost and the right gas stations for the following lists of gas stations, distances and gas tank capacity.

   Prices (cents per mile)

   $$[12,14,21,14,17,22,11,16,17,12,30,25,27,24,22,15,24,23,15,21]$$

   Distances (miles)

   $$[31,42,31,33,12,34,55,25,34,64,24,13,52,33,23,64,43,25,15]$$

   Your car can travel 170 miles on a tank of gas.

6. (2 points) What are the worst case time and space requirements of your complete solution?

   Answer:

   Even I can't finish the algorithm. We definitely need a double-loop to loop the available gas station prices for the distances we've already traveled. So I will say running time should be $O(n^2)$.

   We need two array to store the data: one for distance, one for the prices. So space should be $O(n)$.

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.