SAN FRANCISCO STATE UNIVERSITY
Computer Science Department

CSC510 Analysis of Algorithms–
Algorithm Challenge 3: Dynamic Programming

Instructor: Jose Ortiz

Full Name: Edel Jhon Cenario
Student ID: 921121224

Assignment Instructions. Must read!

Note: Failure to follow the following instructions in detail will impact your grade negatively.

1. This algorithm challenge is worth 9%, and will be graded using a grading point scale where the maximum possible grade is 9 points
2. Handwriting work or screenshots of your work are not allowed. In addition, all the pseudocode done in this algorithm challenge must be done using LaTeX. Students who fail to meet this policy won't get credit for their work. Note that for pseudocode, I only want to see the compiled PDF psudocode, instead of the code to create the pseudocode.
3. Each section of this algorithm challenge is worth 2.25 points
4. Take into account that in this type of assignments, I am more interested in all the different approaches you take to solve the problem rather than on the final solution.

Problem Statement

You are an electrician contractor, and you just finished a very specialized job that used wires with a very specific gauge (amp capacity). Projects that need that type of wire are very rare. Therefore, you have decided to sell all the extra wire to get some profit from it. What is the maximum profit you can get if you sell $N(ft)$ of wire (sold per foot)?

For example: given a list of prices (in dollars) $P = [9, 8, 5, 3]$ per ft, where the $(indexes + 1)$ of those prices represent the number of ft, find the maximum profit for selling 3 ft of wire.

From the above data, we know that 1ft = \$9, 2ft = \$8, 3ft = \$5, and 4ft = \$3 Then, 3ft of wire can be sold in any of the following ways:

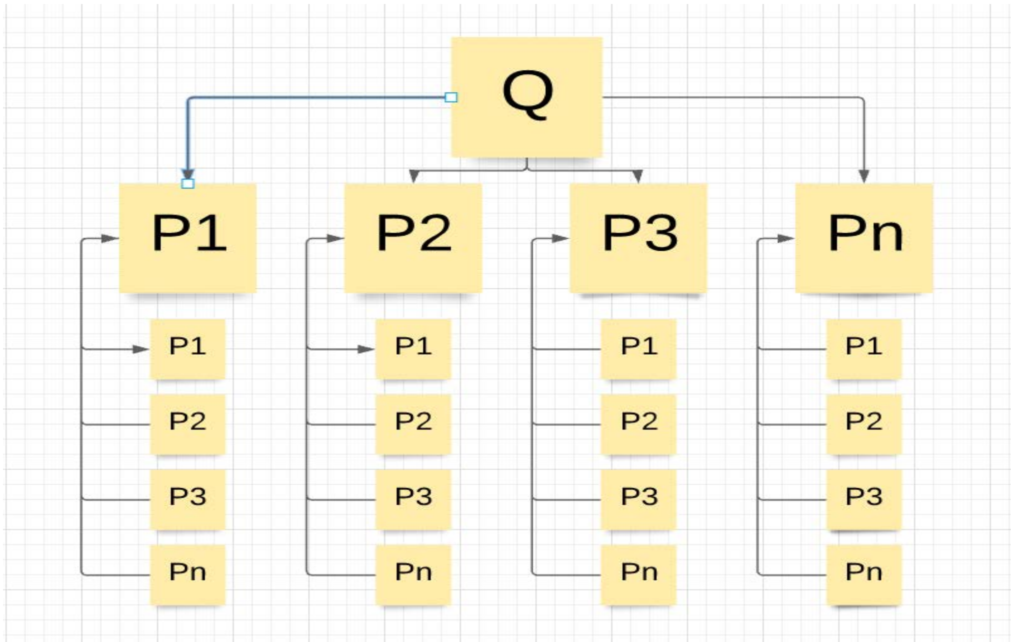
- 3ft = \$5
- 1ft + 2ft = \$9 + \$8 = \$17
- 1ft + 1ft + 1ft = \$9 + \$9 + \$9 = \$27

Therefore you will get the maximum profit from 3ft of wire if you sell them in pieces of 1ft.

Your work starts here

1. Given list of prices $P = [p_1, p_2, p_3, ..., p_n]$ and a random number of ft where $0 < ft \leq n$, create the algorithm to solve the problem using the following two approaches (including an educated guess of their time complexities):

   (a) Brute Force



In this diagram, on the first row of P1, P2, P3, Pn.....
each squares are represented by another set of P1, P2, P3, Pn........
And each set of squares are represented by another set of P1, P2, P3, Pn.....
It is sort of a never ending cycle in which each squares are represented by many more squares.

   (b) Dynamic programming

I created this table using the Maximization formula:
A[i][L] = MAX(A[i-1] [L], A[i] [L-1])

Pi ($)	Li (ft)	i	0 (ft)	1 (ft)	2 (ft)	3 (ft)
0	0	0	0	0	0	0
9	1	0	0	9	18	27
8	2	1	0	9	18	27
5	3	2	0	9	18	27

To further check the validity of the table, I used the values of i and L, and plugged it in my equation to calculate the values. Using i = 0 and L = 0, if we use the Maximization formula, we will come up with the answers 0 for every single column. That would make sense because as we use the values i = 0, our entire formula would equal to 0, thus, it would not matter whether we plug in 0 (ft), 1(ft), etc.

2. Write the pseudocode that represents your algorithm from problem (1.b). **Note that in this problem, I am asking for the compiled LaTeX pseudocode (PDF format) instead of the LaTeX code that creates this pseudocode**

```
function MAXPROFIT( amount, wire )
    INITIALIZE: p = amount of profit, L = length of wire, i = 0,
                y = 0, A = profit and length of wire 2D matrix
    Assert: ( index + 1 ) is the number of feet, where index = 0 is
equals to 1ft
            p is a list of prices, and L is the length of wire
    if L = 1 then
        return 1
    end if
    for i to p,     inclusive do
        for y to L,     inclusive do
            A[x][L] = MAX( A[i - 1] [L] , A[i][L - 1] + P_i )
        end for
    end for
    return A[i][L]
```

3. Based on your pseudocode from part 2, state the complexity function and based on that function compute the time complexity of the dynamic programming algorithm that you created. Note that if you implemented a recursive algorithm, you must apply the substitution method to compute the time complexity of your algorithm, and then check it with the Master Theorem. If you, however, implemented an iterative algorithm for this problem, then you must use a step counting approach to compute the time complexity. **Credit for this problem will be given only to the students that show ALL the work step by step including how to solve the summations or recurrences. Incomplete or poor work won't get credit**

In this table, P = profit, and L = length of wire.

one thing at a time

DATE: ....

Time Complexity

$$T(n) = \sum_{i=0}^{P} \sum_{j=0}^{L} 1$$

$$T(n) = \sum_{i=1}^{P+1} \sum_{j=1}^{L+1} 1$$

$$T(n) = \sum_{i=1}^{P+1} L+1$$

$$T(n) = (p+1)\,(L+1)$$

Therefore: $= O(pL)$

Time Complexity $= O(pL)$

# times ran	i	j
L	0	0, 1, 2, 3, ... L
L-1	1	1, 2, 3, 4, ... L
L-2	2	2, 3, 4, 5, ... L
L-3	3	3, 4, 5, 6, ... L
...	...	...
1	P	L

4. Modify your algorithm from problem 1.b (Dynamic Programming) so now you must compute the maximum profit you can get if you sell **all the wire**. Will this modification, in the dynamic programming algorithm, change the time complexity you got from part 3? Explain in detail to get credit.

To achieve the maximum profit we get if we were to sell all the wire, we can apply dynamic programming to our algorithm to improve it as such:

The idea is that we will be trying to get maximum profit by applying our maximization formula for each wire sold.

#	P1	P2	P3
0	0	0	0
1	P1	P1	P1
2	2P1	MAX(2P1, P2)	MAX(P1+P2)
3	3P1	MAX(P1 + P2 + P3)	MAX(P1+P2, P3, 3P1)
4	4P1	MAX(P3P1, 2P2, P3+P2)	MAX(4P1, 2P2, P3 + P1, P4)

With this change, the time complexity will still be O(pL).

To further understand this modification to maximize our profit, let us discuss the process that is taking place in here.

Let us take our max profit value of $36. Our max profit is achievable if we were to do 4ft. What that means is that when we are maximizing our profit, we are iterating over the 2D matrix which will result in the time complexity of O(pL).

If we are to apply the same idea, with another value, per se, 3ft, 2ft, or 1ft. Our time complexity will stay the same because the values we are iterating through are technically still "the same".