Assignment 4

Algorithms & Complexity (CIS 522-01)

Javier Arechalde

March 30, 2018

Part A: Read the solved exercises and Practice

Solved excercise #1 in Chapter 6

In this problem, we want to place billboards in a highway to get maximum revenue. The highway will be M miles long, and will have n locations on which we can locate the different billboards, each one of this locations will give us $r_i > 0$ revenue. There is also a regulation that doesn't allow two billboards to be placed closer than 5 miles away from each other.

The goal of this problem is to find the billboard placements that will give us the maximum revenue, while following all the given regulations.

Algorithm Pseudocode

Algorithm 1 Billboards Pseudocode

```
Initiallize $M[0] = 0$ and $M[1] = r_1$
for $j = 2, 3, ..., n$ do
    if $x_j - x_{j-1} \geq 5$ then
        $M[j] = M[j-1] + r_j$
    else Find the closest possible value $(x_j - x_i \geq 5)$
        if $M[i] + r_j > M[j-1]$ then
            $M[j] = M[i] + r_j$
        else
            $M[j] = M[j-1]$
        end if
    end if
end for

return M[n]
```

Solution for problem instance of size 10

The code for a problem instance of size 10 is as follows. Run it to see the results.

```
x = [1,10,13,14,20,23,28,30,36,40]
r = [10,3,4,20,10,7,6,3,10,20]

n = len(x)

#Initialize M
```

```
M = [0]*(n+1)

#Initializing M[0] and M[1]
M[0] = 0
M[1] = r[0]

for j in range (2,n+1):
  print('j = %i' %j)
  print('Distance to the previous point: %i' %(x[j-1]-x[j-2]))
  if x[j-1]-x[j-2]>=5:
   M[j] = M[j-1]+r[j-1]
   print('M[%i] = %i' %(j,M[j]))
   print('\n')
  else:
   #Look for the eastmost valid
   print('Looking for the eastmost valid value')
   for i in range(j-1,-1,-1):
     print('%i-%i>=5?' %(x[j-1],x[i-1]))
     if x[j-1]-x[i-1]>=5:
      print('YES')
      print('M[%i]+%i>M[%i]?' %(i,r[j-1],j-1))
      if M[i]+r[j-1]>M[j-1]:
       print('YES')
       print('M[%i] = M[%i] + %i' %(j,i,r[j-1]))
       M[j] = M[i]+r[j-1]
       print('M[%i] = %i' %(j,M[j]))
       print('\n')
      else:
       print('NO')
       print('M[%i] = M[%i]' %(j,j-1))
       M[j] = M[j-1]
       print('M[%i] = %i' %(j,M[j]))
       print('\n')
      break
     else:
      print('NO')

print('MAXIMUM REVENUE: %i' %M[n])
```

Time Complexity

As we have to run over all the values in the set, to find the optimal combination of billboard locations, the running time of our implementation will be $O(n)$.

Part B: Problem Solving

Consulting Jobs

We are in a company that does consulting jobs. Each week we have two different jobs to choose from, one thats a low stress job and another one thats a high stress job. Each one of this jobs will give us a different revenue l_i for the low stress job and h_i for the high stress job. The requirement for choosing a high stress job is that the previous week the team should have been resting, then doing no job at all.

The goal is to find the combination of jobs that gives us the maximum revenue, taking into account that if we want to choose a high stress job, the condition stated before has to be met.

Problem Model

We will solve this problem using dynamic programming. In each iteration we will have to choose between doing the high stress job, doing the low stress job, or resting so the next day we can do the high stress job.

We will have a list of low stress jobs, and it's respective revenue for each of them $l = [l_1 = rl_1, l_2 = rl_2, ..., l_n = rl_n]$ and a list of the high stress jobs $h = [h_1 = rh_1, h_2 = rh_2, ..., h_n = rh_n]$ with these jobs, and the revenue expected for each of them.

We will return a list containing the planning, and the expected revenue from this planning.

Pseudocode

Algorithm 2 Consulting Jobs Pseudocode

```
1: for iteration $i = 1$ to $n$ do
2:     if $h_{i+1} > l_i + l_{i+1}$ then
3:         We choose to rest in week $i$
4:         We choose the high stress job on week $i + 1$
5:         Continue on iteration $i + 2$
6:     else
7:         We choose low-stress job
8:         We continue on iteraton $i + 1$
9:     end if
10: end for
```

Implementation

Here is the code for the implementation of the *pseudocode* shown below.

```
#Low stress jobs
l = [10,1,80,10,10]

#High Stress jobs
h = [5,50,160,5,1]

length = len(l)

#Initializing the revenue to 0
rev = 0

#Adding 0 to the array
l.append(0)
h.append(0)

#Rest and skip variable
rest = 0
skip = 0

for i in range(0,length):

 if skip==1:
  skip = 0
  continue

 #If we chose to rest, we skip this iteration
 if rest == 1:
  if h[i+1]>h[i]+l[i+1]:
   rev = rev+h[i+1]+l[i-1]
   rest = 0
   skip = 1
   continue
  else:
   rev = rev+h[i]
   rest = 0
  continue

 print(i)
 print(l[i],l[i+1])
 if h[i+1]>(l[i]+l[i+1]):
  print('Resting and choosing the high stress job')
  rest = 1
  #Else we choose the low stress job
  else:
    print("Choosing the low stress job")
    rev = rev+l[i]

print(rev)
```

Running time

The running time of our algorithm will be $O(n)$, as we have to go over all the n weeks to choose the best possible combination of jobs.

Carrier Selection

We are a consulting company that manufactures PC equipment and ships it all over the country. We have a projected supply s_i for the next weeks, which is measured in pounds. Each week's suppy can be carried by two different companies.

- Company A: Charges per pound r
- Company B: Charges a fixed amount c per week

The only restriction in this problem is that if we select company B, it must be chosen in four continuous weeks at a time.

Our goal is to find the combination of carriers that results in the minimum shipping cost.

Problem Model

In this problem, we will have an array s containing the projected supply for the next n weeks, we will also have r which is the amount that company A charges per pound of the produced PC equipment, and c that is the fixed amount company B charges per week for delivering the equipment.

At the beginning of our algorithm, we will start by only choosing company A, until we reach week 4, once we reach week 4, we will start checking what is more expensive, if choosing company A for the past 3 weeks and that week, or choosing the flat rate of company B. In case using company B is the better choice, we will update the best cost for that week with the best cost four weeks ago plus the flat rate multiplied by the four weeks. Otherwise we will add the cost per pound multiplied for the projected supply for that week to the previous week best cost.

In the end, we will have an array containing the best cost possible for each of the weeks.

Pseudocode

Algorithm 3 Carrier Selection Pseudocode

```
We set $Cost_0 = 0$
for $i$ in range $0 \to n$ do
    if $i < 3$ then
        $Cost_i = Cost_{i-1} + s_i * r$
    else
        if $Cost_{i-1} + s_i * r > 4 * c$ then
            $Cost_i = Cost_{i-4} + 4 * c$
        else
            $Cost_i = Cost_{i-1} + s_i * r$
        end if
    end if
end for
return Best cost possible $Cost_n$
```

Running time

The running time of our implementation will be $O(n)$, as we have to go through the n weeks to find the lowest cost possible to deliver our product supply.