Assignment - 2

Mithilaash Jayakumar
G01206238

1) Consider a modification of rod cutting problem in which addition to a price P_i for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the cost of making the cuts. Give a dynamic programming algorithm to solve this.

Ans:-

```
ALG (p, n, c)
{
1. Let A[0 ... n] is a new array
2. A[0] = 0
3. for j = 1 to n
4.     q = p[j]
5.     for i = 1 to j-1
6.         q = max(q, p[i] + A[j-i] - c)
7.     A[j] = q
8. return A[n]
}
```

The line 6 represents the fixed cost of making the cut, which is deducted from the revenue. The total revenue when we make no cuts (when i equals j) is p[j]. The line 4 represents when we make no cuts. The line 6 runs from i to j-1 instead of j. These changes help to overcome detecting c from total revenue ever when no cuts are made.

2) Give an $O(n^2)$ time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

Ans:- Let us assume that we are given a list of numbers S.
We take a copy of those numbers as S' and then sort that list.

```
Alg (c, X, Y)
n = c[length X, length Y]
Let A[1...n] is a Array
i = length X
j = length Y
while (i>0 and j>0)
    if X[i] == Y[j]
        A[n] = X[i]
        n = n-1
        i = i-1
        j = j-1
    else if c[i-1, j] ≥ c[i, j-1]
        i = i-1
    else j = j-1
for i=1 to A.length
    print A[i]
```

SubProgram_Length_Auxiliary (X, Y, c, a)

m = |X|

n = |Y|

if c[m,n] != 0 or m == 0 or n == 0

return

if X[m] == Y[n]

a[m,n] = ↖

c[m,n] = SubProgram_Length_Auxiliary (X[1...m-1], Y[1...n-1], c, a) + 1

else if SubProgram_Length_Auxiliary (X[1...m-1], Y, c, a) ≥ SubProgram_Length_Auxiliary (X, Y[1....n-1], c, a)

a[m,n] = ↑

c[m,n] = SubProgram_Length_Auxiliary (X[1...m-1], Y, c, a)

else

a[m,n] = ←

c[m,n] = SubProgram_Length_Auxiliary (X, Y[1..n-1], c, a)

SubProgram_Length (X, Y)

let c[1...|X|, 1...|Y|] and a[1...|X|, 1...|Y|] are new tables

SubProgram_Length_Auxiliary (X, Y, c, a)

return c and a

We will run the Lcs algorithm on these two lists.

The resulting Lcs will be a monotone increasing sequence because it is a subsequence of S' which is sorted.

It is also the longest monotone because being a subsequence of S' only adds restriction that the monotone must be increasing.

Since $|S| = |S'| = n$ and the sorting happens in $O(n^2)$

3.) A palindrome is a non empty string over some alphabet that reads same forward and backward. Give an algorithm to find the longest palindrome that is a subsequent of given input string.

Ans: Let us assume that the given word is stored in an array $A[1 \ldots n]$

We must divide at some position i inorder for the palindrome to be a subsequence.

We then solve the problem on $A[1..i]$ and $A[i+1 \ldots n]$, adding an extra letter incase the palindrome has a center letter.

There are n places to split the input, also the Lcs problem takes $O(n^2)$, therefore the palindrome problem takes $O(n^3)$.

4) You are given rectangular piece of cloth with dimensions $X \times Y$, where X and Y are positive integers; and a list of products that can be made from cloth.

Ans: In the optimal solution we can either make an vertical or horizontal cut or take maximum profit from a product of dimensions $X \times Y$, if one exists. The best return in this case is the sum of best returns from two resulting pieces.

Given a cloth of size $X \times Y$, we can make product of size $X \times Y$ or one of $X-1$ or $Y-1$ cuts in case of horizontal / vertical cuts. If multiple products with same dimension exist $i \times j$ or $j \times i$, then we only consider the product which has maximum selling price.

Let $P(i,j)$ indicate the maximum selling price of products that can be made from piece of cloth of $i \times j$. We want $P(X,Y)$. Let $p(i,j)$ indicate the price of product with highest selling price that can be made from a piece of cloth of dimensions $i \times j$ or $j \times i$ if exist or $p(i,j) = 0$.

$$P(i,j) = \max\{\max\{P(a,j) + P(i-a,j)\}, \max\{P(i,j-b) + P(i,b)\}, P(i,j)\}$$ where

$a \geq 1$ and $a < i, j$

gives the recurrence relation of our problem.

Proof:-

The base case is $P(0,0) = 0$

Assume that $P(i,j)$ gives the maximum profit obtainable from a cloth of dimensions $i \times j$ for all $i \leq k$ and for all $j \leq l$

We have following options for computing $P(k+1, j)$, $\forall j \leq l$

1) $P(k+1, j)$ can be the price of the product with the highest selling price that can be made from a cloth of dimensions $k+1 \times j$ or $j \times k+1$ if exists. Otherwise, $P(k+1, j)$ can be 0. This gives us the option $P(k+1, j) = p(k+1, j)$.

2) The cloth can be cut vertically in one of k places. We consider all possible cuts made at some position a, $1 \leq a < k+1$, to find the vertical cut that gives us the maximum selling prices for the two pieces obtained, $P(a, j) + P(k+1-a, j)$.

3.) The cloth can be cut horizontally in one of j-1 places. We consider all possible cuts made at some position b, $1 \leq b < j$, to find the horizontal cut that gives us the maximum selling prices for the two pieces obtained, $P(k+1, j-b) + P(k+1, b)$. These cases give the maximum possible value based on assuptions. of these three cases we choose one that results in the maximum selling price.

The optimality of cases $P(i, l+1), \forall i \leq k$ and $P(k+1, l+1)$ is analogues to the one above. This completes the proof.

Pseudo code:-

```
p(x,y)  // Matrix to store price of single cloth of
          dimension i x j

P(x,y) // Matrix to store maximum return from
          cloth of dimensions i x j

for i from 1 to X:
for j from 1 to Y:
  p(i,j) = -1
  end for
end for // filling in value of p(i,j)
```

```
for i from 1 to n:
if (p(a_i, b_i) < c_i):
 p(a_i, b_i) = c_i
 end of for
for i from 1 to X:
for j from 1 to Y:
p(i,j) = -1
end for
end for
P(0,0) = 0
function maxreturn (i,j) // function to calculate maximum return
                         // from a cloth of dimension i*j
if (P(i,j) != -1):
return P(i,j)
else
P(i,j) = max { max {P(a,j) + P(i-a,j)}, max {P(i,j-b) + P(i,b)}, P(i,j)}
 where 1 ≤ a < i    1 ≤ a < j

 return P(i,j)
 end if
 return P(X,Y)
```

Running time:-

There exist $O(X \cdot Y)$ sub problems of the type $P(i,j)$ as i and j range from 1 to X and 1 to Y. Calculating all the values of $p(i,j)$ initially takes $O(X \cdot Y + n)$ time. Calculating the value of each subproblem $P(i,j)$ takes $O(X+Y)$ to find the maximum selling price over all possible cuts. Therefore, the overall running time of the algorithm is $O(X \cdot Y \cdot (X+Y) + n)$.