CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

Q1. [2 points] What does dynamic programming have in common with divide-and-conquer? What is the principal difference between them?

1. Dynamic programming: It is a concept that divides a given problem and break it into a resonable number of subproblems, so that optimal solutions can be used on subproblems to provide optimal solutions on the bigger problem. Ex: LCS, Knapsack, etc.

Similarities with Divide & Conquer:

→ Both techniques divide a larger sub-proble into smaller sub-problems. and finally integrated to form a final solution.

Differences between Dynamic Programming & Divide and Conquer:

⇒ Problem with multiple overlapping of subproblems are solved using dynamic programming. It uses memoization and tabulation to remember and reuse the subproblem results thereby improving the performance.

Q2. [5 points] Consider a modification of the rod-cutting problem in which, in addition to a price pi 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 costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem. (Ex. 15.1-3)

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

2. Solution is to modify Bottom-up-cut-rod algorithm.

```
Modified-cut-rod (p, n, c)
    let r[0...n] be a new array
    r[0] = 0
    for i = 1 to n
        q = p[i]
        for j = 1 to i-1
            q = max (q, p[j]+r[i-j]-c)
        r[j] = q
    return r[n]
```

We need to account for cost c of every iteration but at last iteration where j=i, we make loop run to i-1 instead of i iterations to make sure that c is subtracted from the candidate revenue.

Q3. [5 points] We say that a problem exhibits the optimal substructure property when optimal solutions to a problem incorporate optimal solutions to related subproblems, which we may solve independently. Suppose that in the rod-cutting problem, we also had limit li on the number of pieces of length I that we are allowed to produce, for i = 1, 2, . . ., n. Show that the optimal-substructure property described no longer holds. (Ex. 15.3-5)

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

3. If the subproblem solutions are optimal implies optimal solution to a problem, which we may solve independently. When a limit l_i is imposed on pieces of size i, the subproblems cannot be solved indepedently.

length i	1	2	3	4
price p_i	15	20	33	36
limit l	2	1	1	1

Here, only 3 instances do not violate the limits
→ length 4 with price 36
→ length 1 & 3 with prize 48
→ length 1, 1, 2 & 2 with prize 50.

* Here the optimal solution is to cut into 1, 1, 2 2.

Now, for subproblem of length 2, the solutions are
→ 2 with price 20
→ 1 & 1 with price 30

* Here the optimal solution is to cut into 1 & 1

But we cannot use this solution for actual problem because it violates limit of two length-rods., as it results in 4 rods of length 1.

Q4. [5 points] Give pseudocode to reconstruct an LCS from the completed c table and the original sequences X = (x1, x2, ; xm) and Y = (y1, y2, ; yn) in O(m+n) time, without using the b table. (Ex. 15.4-2)
(Hint: Try to benefit from the PRINT-LCS procedure)

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

We can also use the below pseudo code:

```
def compute_lcs(C,m,n):

    i = m
    j = n

    x = C[i][j]

    res = [""] * (x+1)

    while i > 0 and j > 0:
        if X[i-1] == Y[j-1]:
            res[x-1] = X[i-1]
            i -= 1
            j -= 1
            x -= 1
        elif C[i-1][j] > C[i][j-1]:
            i -= 1
        else:
            j -= 1

    return res
```

Q4. [5 points] Give pseudocode to reconstruct an LCS from the completed c table and the original sequences X = (x1, x2, ; xm) and Y = (y1, y2, ; yn) in O(m+n) time, without using the b table. (Ex. 15.4-2)

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

4. LCS pseudcode to print solution in $O(m+n)$ times

```
print_lcs (a, X, Y, x, y)
    if a[x,y] == 0
        return
    if X[x] == Y[y]
        print_lcs (a, X, Y, x-1, y-1)
    else if a[x-1, y] > a[x, y-1]
        print_lcs (c, X, Y, x-1, y)
    else
        print_lcs (a, X, Y, x, y-1)
```

Q5. Consider two teams, A and B, playing a series of games until one of the teams wins n games. Assume that the probability of A winning a game is the same for each game and equal to p and the probability of A losing a game is g = 1 -p. (Hence, there are no ties.) Let P(i,j) be the probability of A winning the series if A needs i more games to win the series and B needs j more games to win the series.

a) Set up a recurrence relation for P(i,j) that can be used by a dynamic programming algorithm. [2 points]

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

5 a) Recurrence relation for $P(i,j)$

The probability for A winning a game is 'p'
and probability of A losing game is $q = 1-P$.
If team A wins, still $i-1$ more wins are needed by A. where B will still need j wins. If A loses game, A will still need 'i' wins while B need $j-1$ wins.

Recurrence $(i,j>0)$

$$P(i,j) = pP(i-1,j) + qP(i,j-1)$$

$$P(0,j) = 1 \quad (j>0)$$

$$P(i,0) = 0 \quad (i>0)$$

b) Find the probability of team A winning a seven-game series if the probability of it winning a game is 0.4. [2 points]

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

b) Dynamic programming table

· Using above recurrence equation

input	0	1	2	3	4
	0	1	1	1	1
1	0	0.40	0.64	0.78	0.87
2	0	0.16	0.35	0.52	0.66
3	0	0.06	0.18	0.32	0.46
4	0	0.03	0.09	0.18	0.29

Hence $P[4,4] \simeq 0.29$

c) Write the pseudocode of the dynamic programming algorithm for solving this problem and analyze its running time. [4 points]

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 3 – Dynamic Programming

Neha Goud Baddam
11519516

c) Algorithm World Series (n, p)

```
q ← 1-p
for j ← 1 to n do
    P[0,j] ← 1.0
for i ← 1 to n do
    P[i,0] ← 0.0
    for j ← 1 to n do
        P[i,j] ← p * P[i-1,j] + q*P[i,j-1]

return P[n,n]
```

Input : The number of victories n required to win the series & 'p' probability of one team winning.

Output : The probability of team winning the series

→ Each entry of table is computed in $\Theta(1)$ time

→ Hence, time & space efficiency is $\Theta(n^2)$.