**COT 5405 ANALYSIS OF ALGORITHMS**

**PROGRAMMING PROJECT**

**Spring 2022**

|  |  |
| --- | --- |
| **Team Members** | |
| Mandar Palkar | 2140-6740 |
| Siddhi Wadgaonkar | 9544-2212 |

Table of Contents

* Problem Definition
* Design and Analysis of Algorithms

(For each alg)

* Algorithm
* Correctness
* Mathematical Recursive Formulation expressing optimal substructure property
* Time Complexity
* Space Complexity
* Experimental Comparative Study
* Graphs etc.
* Conclusion

(For each prog task)

* Ease of implementation
* Other potential technical challenges

**Problem 1**

**ALG1 – Brute Force**

Pseudo-code

**MAXIMIZE-AIR-QUALITY-INDEX(A)**

1 n = A.Length

2 maxSumSoFar = -INF

3 for left = 0 to n-1

4 for right = left to n – 1

5 currSum = 0

6 for temp = left to right

7 currSum = currSum + A[temp]

8 if currSum > maxSumSoFar

9 maxSumSoFar = currSum

10 L = left

11 R = right

12 return maxSumSoFar, L, R

**ALG2 – Dynamic Programming ( O(n2) )**

Pseudo-code

**MAXIMIZE-AIR-QUALITY-INDEX(A)**

1. maxSumSoFar = -INF
2. n = A.Length
3. temp = 0
4. Array DP[0…n+1]
5. DP[0] = 0
6. L = -1
7. R = -1
8. for i = 1 to n
9. DP[i] = DP[i-1] + A[i-1]
10. for left= 1 to n
11. for right= left to n
12. temp = DP[right] - DP[left-1]
13. if temp> maxSumSoFar
14. maxSumSoFar = currSum
15. L = left - 1
16. R = right - 1
17. return maxSumSoFar, L, R

**ALG3 TASK 3A – Dynamic Programming ( O(n) )**

Pseudo-code

**HELPER(DP,index)**

1. if index >= A.Length
2. return 0
3. if DP[index] > 0
4. return A[index]
5. curr = DP[index]
6. sum = DP[index]+ helper(A, index+1)
7. DP[index] = max(curr, sum)
8. return DP[index]

**MAXIMIZE-AIR-QUALITY-INDEX(A)**

1. n = A.Length
2. maxSumSoFar = -INF
3. x = -INF
4. s=0
5. e=0
6. DP [0…n+1]
7. for i = 0 to n
8. x = HELPER(DP,i)
9. if x > maxSumSoFar
10. maxSumSoFar = x
11. s = i
12. x = maxSumSoFar
13. e = s
14. While x > 0
15. x = x - A[e]
16. if x == 0
17. Break
18. e = e + 1
20. return maxSumSoFar, L, R

**ALG3 TASK3B – Dynamic Programming ( O(n) )**

Pseudo-code

**MAXIMIZE-AIR-QUALITY-INDEX(A)**

1. maxSumSoFar = -INF
2. n = A.Length
3. L = -1
4. R = -1
5. currSum = 0
6. currStart = 0
7. For i = 0 to n
8. currSum = currSum + A[i]
9. if maxSumSoFar < currSum
10. maxSumSoFar = currSum
11. l = currStart
12. r = i
13. if currSum < 0
14. currSum = 0
15. currStart = i+1
16. return maxSumSoFar, L, R

**ALG4 – Brute Force ( O(n6) )**

Pseudo-code

**MAXIMIZE-AIR-QUALITY-INDEX(MAT)**

1. maxSumSoFar = -INF
2. rows = MAT.Length
3. cols = MAT[0].Length
4. xLeft = -1
5. yLeft = -1
6. xRight = -1
7. yRight = -1
8. For r1 = 0 to rows
9. For c1 = 0 to cols
10. For r2 = r1 to rows
11. For c2 = c1 to cols
12. currSum = 0
13. For m = r1 to r2
14. For n = c1 to c2
15. currSum += MAT[m][n]
16. if maxSumSoFar < currSum
17. maxSumSoFar = currSum
18. xLeft = r1
19. yLeft = c1
20. xRight = r2
21. yRight = c2
22. return maxSumSoFar, xLeft, yLeft, xRight, yRight

**ALG5 – DYNAMIC PROGRAMMING ( O(n4) )**

Pseudo-code

**MAXIMIZE-AIR-QUALITY-INDEX(MAT)**

1. rows = MAT.Length
2. cols = MAT[0].Length
3. SUB [0…rows][0..cols]
4. xLeft = -1
5. yLeft = -1
6. xRight = -1
7. yRight = -1
8. maxSum = -INF
9. For r1 = 0 to rows
10. For r2 = 0 to cols
11. If r1 == 0 || r2 == 0
12. SUB[r1][r2] = 0
13. Else
14. SUB[r1][r2] = SUB[r1-1][r2] + SUB[r1][r2-1]

-SUB[r1-1][r2-1]+SUB[r1-1][r2-1]

2. For r1 = 0 to rows
3. For r2 = r1 to rows
4. For c1 = 0 to cols
5. For c2 = c1 to cols
6. submatrix\_sum = SUB[r2+1][c2+1] - SUB[r2+1][c1]- SUB[r1][c2+1]+ sub[r1][c1]
7. If submatrix\_sum > maxSum
8. maxSum = submatrix\_sum
9. xLeft = r1
10. yLeft = c1
11. xRight = r2
12. yRight = c2
13. return maxSum, xLeft, yLeft, xRight, yRight

**ALG6 – DYNAMIC PROGRAMMING ( O(n3) )**

Pseudo-code

**HELPER-ALG3A(A)**

1. maxSumSoFar = -INF
2. n = A.Length
3. currSum = 0
4. currStart = 0
5. For i = 0 to n
6. currSum = currSum + A[i]
7. if maxSumSoFar < currSum
8. maxSumSoFar = currSum
9. l = currStart
10. r = i
11. if currSum < 0
12. currSum = 0
13. currStart = i+1
14. return maxSumSoFar

**MAXIMIZE-AIR-QUALITY-INDEX(MAT)**

1. maxSumSoFar = -INF
2. rows = MAT.Length
3. cols = MAT[0].Length
4. prefix [0…rows+1][0…cols+1]
5. xLeft = -1
6. yLeft = -1
7. xRight = -1
8. yRight = -1
9. For i = 0 to rows
10. For j = 0 to cols
11. if j == 0
12. prefix[i][j] = MAT[i][j]
13. else
14. prefix[i][j] = prefix[i][j] + prefix[i][j-1]
16. maxSum = -INF
17. for i = 0 to n
18. for j = i to n
19. V[0…rows]
20. for k = 0 to m
21. ele = 0
22. if i == 0
23. ele = prefix[k][j]
24. else
25. ele = prefix[k][j] – prefix[k][i-1]
26. V[k] = ele
28. maxTempSum = HELPER-ALG3A(V)
30. If maxSum < maxTempSum
31. maxSum = temp
32. tt = i
33. bb = j
34. xLeft = ll
35. xRight = rr
37. yLeft = tt
38. yRight = bb
39. return maxSum, xLeft, yLeft, xRight, yRight