CSE 595: Advanced Topics in Computer Science
Presentation 6

Zeeshan Shaikh

Department of Computer Science, Stonybrook University

07/08/2021

Topics for today's presentation

- Painter's Partition Problem.

Step 1: Painter's Partition Problem

- Given: An array where each element represents the length of a board and number of painters for the job (represented by 'k').
- Aim: Find the minimum time required to color the boards under the constraints that any painter can paint only continuous sections of the board.
- Note: Each painter takes 1 unit time to paint 1 unit of the board.
- Example:
  - Input: k = 2, array = [10, 20, 30, 40]
  - Output: 60
  - Explanation: Since a painter can only paint continuous boards, dividing it as [10,20,30] and [40] is the best way to do so. Painter 1 will take 60 units of time and Painter 2 will take 40 units of time.

Step 2: Sub-problem

Approach: We can see from the example that the approach to this problem is to partition the array into $<= k$ partitions such that the maximum sum of elements in the partition is minimum.

- Input: k = 2, array = [10, 20, 30, 40]
- Output: 60
- Possible Partition:
  - No partition: Sum = 100, therefore, time (T) = 100
  - Two partitions:
    - Case 1: [10], [20,30, 40]. Max time needed = 90
    - Case 2: [10, 20], [30, 40], Max time needed = 70
    - Case 3: [10, 20, 30], [40], Max time needed = 60
    - We select the option with the minimum time required of the three options i.e Case 3 with time = 60

Step 2: Sub-problem (continued)

Notations used:

- Array holding the length of boards: a[n]
- Array holding the sum of boards till the current element: s[n] (Why do we need this? This array holds the "continuous" sum till the current element which is a constraint of the problem)
- Matrix/Table holding the time required for 'k' painters for 'n' boards: T[k+1][n+1]

Step 3: Instructions

- Prepare the array of sum of the boards till the given element

Algorithm 1 Time Complexity: O(N)

for i = 1 to N **do**
  $s[i] = s[i-1] + a[i]$
end for

- Entry in the table for the case scenario when we have 1 painter

Algorithm 2 Time Complexity: O(N)

for i = 1 to N **do**
  $T[1][i] = s[i]$
end for

Step 3: Instructions(Continued)

- Entry in the table for the case scenario when we have a single board (1 entry in the array)

Algorithm 3 Time Complexity: O(K)

```
for i = 1 to k do
    T[i][1] = a[0]
end for
```

- Entry in the table for every other case

Algorithm 4 Time Complexity: O(k*N^2)

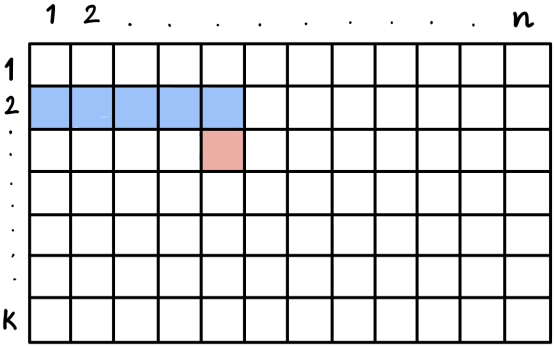
```
for i = 2 to k do
    for j = 2 to N do
        for p = 1 to j do
            best = min(best, max(T[i-1][p], (s[j] - s[p])))
        end for
    end for
end for
```

Step 4: Recurrence

s[a, b] = s[b] - s[a]

$$\text{T[i, j]} = \begin{cases} min \begin{cases} max(T[i-1,1], S[2,j]) \\ max(T[i-1,2], S[3,j]) \\ ... \\ max(T[i-1,j-1], S[j,j]) \\ max(T[i-1,j], S[j+1,j]) \end{cases} \end{cases}$$

Step 5: Dependency



Example: $T[3][5] = \left\{ \min \left\{ \begin{array}{l} \max(T[2,1], s[2,5]) \\ \max(T[2,2], s[3,5]) \\ \vdots \\ \max(T[2,5], s[6,5]) \end{array} \right. \right.$

Figure 1: Dependency

Step 6: Painters Problem Algorithm

Algorithm 5 Painters Problem

```
for i = 1 to N do
    T[1][i] = s[i]
end for
for i = 1 to N do
    T[1][i] = s[i]
end for
for i = 1 to k do
    T[i][1] = a[0]
end for
for i = 2 to k do
    for j = 2 to N do
        best = INTMAX
        for p = 1 to j do
            best = min(best, max(T[i-1][p], (s[j] - s[p])))
        end for
        T[i,j] = result
    end for
```

Step 7: Tables

i	1	2	3	4
a[i]	10	20	30	40
s[i]	10	30	60	100

$k \backslash n$	1	2	3	4
1	10	30	60	100
2	10	20	30	60

Given:

$k = 2$

a[i] such that,

$n = 4$

To find:

T[2][4]

Figure 2: Dependency

Step 8: Complexity Analysis

Complexity Analysis		
Algorithm	Time	Space
Painters Problem	$O(K * N^2)$	$O(K * N)$

where K is the number of painters and n is the number of boards.