Hong Kong Olympiad in Informatics 2007 Senior Question 3 SOS

Hackson Leung

13^{th} January, 2007

Statistics

Disappointing

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

Max : 100

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

$$\text{Max} : 100$$
$$\#\text{Max} : 3$$

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

$$\begin{aligned} \text{Max} &: 100 \\ \#\text{Max} &: 3 \\ \text{Min} &: 0 \end{aligned}$$

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

Max	:	100
#Max	:	3
Min	:	0
Mean	:	28.627

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

Max	:	100
#Max	:	3
Min	:	0
Mean	:	28.627
Mode	:	10

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

Max :	100
#Max :	3
Min :	0
Mean :	28.627
Mode :	10
Std. Dev. :	30.423

Statistics

Disappointing

HKOI 2007 Senior Q3 Statistics

Max	100
#Max	3
Min	0
Mean	28.627
Mode	10
Std. Dev.	30.423

orz

Problem Description

Problem Description

- Given a $N \times M$ characters map, each character can be a brick or a space

Problem Description

- Given a $N \times M$ characters map, each character can be a brick or a space
- You can rotate each row to right or to left

Problem Description

- Given a $N \times M$ characters map, each character can be a brick or a space
- You can rotate each row to right or to left
- Find the minimum number of rotation such that there exists at least one column of N spaces

Naïve Solution

Naïve Solution

- Asking for combinations

Naïve Solution

- Asking for combinations
- Asking for optimal solution

Naïve Solution

- Asking for combinations
- Asking for optimal solution
- Exhuastion!?

Naïve Implementation

Naïve Implementation

Naïve Implementation

Algorithm exhaustion(ROW, STEP)
    Mark this configuration as *visited*
    if ROW = N and *At least one column has N spaces* then
        Answer = Min(Answer, STEP)
    else
        While True do
            Rotate ROW to left
            if current configuration is not *visited* then
                Exhaustion(ROW+1, STEP+Number_of_Left_Rotation)
            else
                break
        Return to original configuration
        {*Do the same for Right Rotation*}

Naïve Implementaion (Continued)

Naïve Implementaion (Continued)

- Clumsy

Naïve Implementaion (Continued)

- Clumsy
- Risk of TLE

Naïve Implementaion (Continued)

- Clumsy
- Risk of TLE
- When you were implementing this, probably you can think of 50% Solution

Observation 1

Observation 1

1. No need to search for all possible combinations

Observation 1

1. No need to search for all possible combinations
2. One row can have at most M different configurations

Observation 1

1. No need to search for all possible combinations
2. One row can have at most M different configurations
3. For each character, we should know the number of minimum rotation made to "pull" a space to that position

Observation 1

1. No need to search for all possible combinations
2. One row can have at most M different configurations
3. For each character, we should know the number of minimum rotation made to "pull" a space to that position
4. Then we can find the minimum roration by comparing the sum

Observation 1

1. No need to search for all possible combinations
2. One row can have at most M different configurations
3. For each character, we should know the number of minimum rotation made to "pull" a space to that position
4. Then we can find the minimum roration by comparing the sum
5. If one level is rotated to left K times, it is equivalent to rotate to right $(M - K)$ mod M times

50% Solution - Implementation

50% Solution - Implementation

50% Solution

Algorithm 50% Solution

MinRotation[][] $\leftarrow \infty$

MinRotation[x][y] $\leftarrow 0$ $//\exists x, y \rightarrow (x, y)$ *is a space*

for each character (r, c) do

if (r, c) is a space then

for $i \leftarrow 1$ to $M - 1$ do

Update MinRotation at (r, c) rotates to left i times with i

Update MinRotation at (r, c) rotates to right i times with i

$$Answer \leftarrow \min_{1 \le i \le M} \{ \sum_{j=1}^{N} MinRotation[j][i] \}$$

50% Solution - Implementation (Continued)

50% Solution - Implementation (Continued)

- Easy to implement

50% Solution - Implementation (Continued)

- Easy to implement
- Reduce memory usage

50% Solution - Implementation (Continued)

- Easy to implement
- Reduce memory usage
- But still, it is not efficient enough to get 100% score

50% Solution - Implementation (Continued)

- Easy to implement
- Reduce memory usage
- But still, it is not efficient enough to get 100% score
- Runtime Complexity: $\mathcal{O}(NM^2)$

Observation 2

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently

Observation 2

1. The bottleneck is “How to find the nearest space to an element” efficiently
2. Consider no rotating stuff first

Observation 2

1. The bottleneck is “How to find the nearest space to an element” efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`

Observation 2

1. The bottleneck is “How to find the nearest space to an element” efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`
   - $P = \{3, 5, 8, 10\}$

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`
   - $P = \{3, 5, 8, 10\}$
   - I want to know where is the nearest space with respect to the 6^{th} element

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`
   - $P = \{3, 5, 8, 10\}$
   - I want to know where is the nearest space with respect to the 6^{th} element
   - 3? 5? 8? 10?

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`
   - $P = \{3, 5, 8, 10\}$
   - I want to know where is the nearest space with respect to the 6^{th} element
   - 3? 5? 8? 10?
   - Obviously P is sorted

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`
   - $P = \{3, 5, 8, 10\}$
   - I want to know where is the nearest space with respect to the 6^{th} element
   - 3? 5? 8? 10?
   - Obviously P is sorted
3. *Binary Search* can be applied

Observation 2

1. The bottleneck is "How to find the nearest space to an element" efficiently
2. Consider no rotating stuff first
   - Let P be a list storing the position of spaces in a row
   - e.g. `bbsbsbbsbs`
   - $P = \{3, 5, 8, 10\}$
   - I want to know where is the nearest space with respect to the 6^{th} element
   - 3? 5? 8? 10?
   - Obviously P is sorted
3. *Binary Search* can be applied
4. How to deal with rotating stuff?

70% Solution - Implementation

70% Solution - Implementation

70% Solution - Implementation

Algorithm 70% Solution

MinRotation[][] $\leftarrow \infty$

MinRotation[x][y] $\leftarrow 0$ $\quad //\exists x, y \rightarrow (x, y)$ *is a space*

for each character (r, c) do

Update MinRotation[r][c] $\leftarrow lower\_bound(P, c)$

$$Answer \leftarrow \min_{1 \le i \le M} \{\sum_{j=1}^{N} MinRotation[j][i]\}$$

70% Solution - Implementation

70% Solution - Implementation

Algorithm 70% Solution
  MinRotation[][] $\leftarrow \infty$
  MinRotation[x][y] $\leftarrow 0$ $// \exists x, y \rightarrow (x, y)$ *is a space*
  for each character (r, c) do
    Update MinRotation[r][c] $\leftarrow$ *lower_bound*(P, c)

$$Answer \leftarrow \min_{1 \le i \le M} \{ \sum_{j=1}^{N} MinRotation[j][i] \}$$

lower_bound is a binary search function that returns the first position where value could be inserted without violating the ordering. (Ref.: http://www.sgi.com/tech/stl/lower_bound.html)

70% Solution - Implementation (Continued)

70% Solution - Implementation (Continued)

- A bit easier to implement

70% Solution - Implementation (Continued)

- A bit easier to implement
- Much efficient

70% Solution - Implementation (Continued)

- A bit easier to implement
- Much efficient
- Runtime Complexity: $\mathcal{O}(NMlgM)$
  Note: $2^x = y \rightarrow x = lgy$

70% Solution - Implementation (Continued)

- A bit easier to implement
- Much efficient
- Runtime Complexity: $\mathcal{O}(NMlgM)$
  Note: $2^x = y \to x = lgy$
- However, if M is sufficiently large, it will still risk TLE

Observation 3

Observation 3

1. Still considering no rotation stuff

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation[α], we can consider MinRotation[$\alpha + 1$] immediately.

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation[α], we can consider MinRotation[$\alpha + 1$] immediately.
   - Suppose we know MinRotation[α] corresponds to the space located at $P[\beta]$ which is on its left, then MinRotation[$\alpha + 1$] can be determined by one of the two "neighbouring" spaces.

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation[α], we can consider MinRotation[$\alpha + 1$] immediately.
   - Suppose we know MinRotation[α] corresponds to the space located at $P[\beta]$ which is on its left, then MinRotation[$\alpha + 1$] can be determined by one of the two "neighbouring" spaces.
   - $\alpha + 1 > P[\beta + 1] \Rightarrow$MinRotation[$\alpha + 1$] $=$ $\min(|\alpha + 1 - P[\beta + 1]|, |\alpha + 1 - P[\beta + 2]|)$
   - Otherwise, MinRotation[$\alpha+1$] $= \min(|\alpha+1-P[\beta]|, |\alpha+1-P[\beta+1]|)$

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation[α], we can consider MinRotation[$\alpha + 1$] immediately.
   - Suppose we know MinRotation[α] corresponds to the space located at $P[\beta]$ which is on its left, then MinRotation[$\alpha + 1$] can be determined by one of the two "neighbouring" spaces.
   - $\alpha + 1 > P[\beta + 1] \Rightarrow$MinRotation[$\alpha + 1$] $=$ $\min(|\alpha + 1 - P[\beta + 1]|, |\alpha + 1 - P[\beta + 2]|)$
   - Otherwise, MinRotation[$\alpha+1$] $= \min(|\alpha+1-P[\beta]|, |\alpha+1-P[\beta+1]|)$
2. How about when $P[\beta] > \alpha$?

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation$[\alpha]$, we can consider MinRotation$[\alpha+1]$ immediately.
   - Suppose we know MinRotation$[\alpha]$ corresponds to the space located at $P[\beta]$ which is on its left, then MinRotation$[\alpha+1]$ can be determined by one of the two "neighbouring" spaces.
   - $\alpha + 1 > P[\beta+1] \Rightarrow$MinRotation$[\alpha+1] = \min(|\alpha+1-P[\beta+1]|, |\alpha+1-P[\beta+2]|)$
   - Otherwise, MinRotation$[\alpha+1] = \min(|\alpha+1-P[\beta]|, |\alpha+1-P[\beta+1]|)$
2. How about when $P[\beta] > \alpha$?
3. Rotation stuff?!?!

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation[α], we can consider MinRotation[$\alpha + 1$] immediately.
   - Suppose we know MinRotation[α] corresponds to the space located at $P[\beta]$ which is on its left, then MinRotation[$\alpha + 1$] can be determined by one of the two "neighbouring" spaces.
   - $\alpha + 1 > P[\beta + 1] \Rightarrow$MinRotation[$\alpha + 1$] $=$ $\min(|\alpha + 1 - P[\beta + 1]|, |\alpha + 1 - P[\beta + 2]|)$
   - Otherwise, MinRotation[$\alpha + 1$] $= \min(|\alpha + 1 - P[\beta]|, |\alpha + 1 - P[\beta + 1]|)$
2. How about when $P[\beta] > \alpha$?
3. Rotation stuff?!?!
4. Much much faster

Observation 3

1. Still considering no rotation stuff
   - If we know MinRotation[α], we can consider MinRotation[$\alpha + 1$] immediately.
   - Suppose we know MinRotation[α] corresponds to the space located at $P[\beta]$ which is on its left, then MinRotation[$\alpha + 1$] can be determined by one of the two "neighbouring" spaces.
   - $\alpha + 1 > P[\beta + 1] \Rightarrow$MinRotation[$\alpha + 1$] $=$ $\min(|\alpha + 1 - P[\beta + 1]|, |\alpha + 1 - P[\beta + 2]|)$
   - Otherwise, MinRotation[$\alpha+1$] $= \min(|\alpha+1-P[\beta]|, |\alpha+1-P[\beta+1]|)$
2. How about when $P[\beta] > \alpha$?
3. Rotation stuff?!?!
4. Much much faster
5. Runtime Coimplexity?

Special Note

Special Note

During Competition:

1. Don't use `cin` and `cout`!
2. Avoid using standard `string` and single character input.

Special Note

During Competition:

1. Don't use `cin` and `cout`!
2. Avoid using standard `string` and single character input.

In HKOI Online Judge:

1. Time Limit is changed to 1 second
2. Input may critically affecting the runtime if you use an improper way.
3. Never use too much memory (Why?)
4. HKOI Online Judge only accept the best solution. Don't try to submit every solution you've written.