Apple Garden

So Pak Yeung
17 - 1 - 2015

Statistics

Mean: 43.27

Standard Deviation: 25.34

Max: 100

100, 98, 95, 70.......

Problem Description

- *N* x *N* grid
- K of the cells contain an apple each
- Other cells contain no apples
- Find the maximum number of apples covered by a single *M* x *M* square

Solution 1(Intuitive Solution)

- For each cell, try to let that cell be the top left corner of the M x M square
- Count how many apples there are in the square
- Choose the maximum one

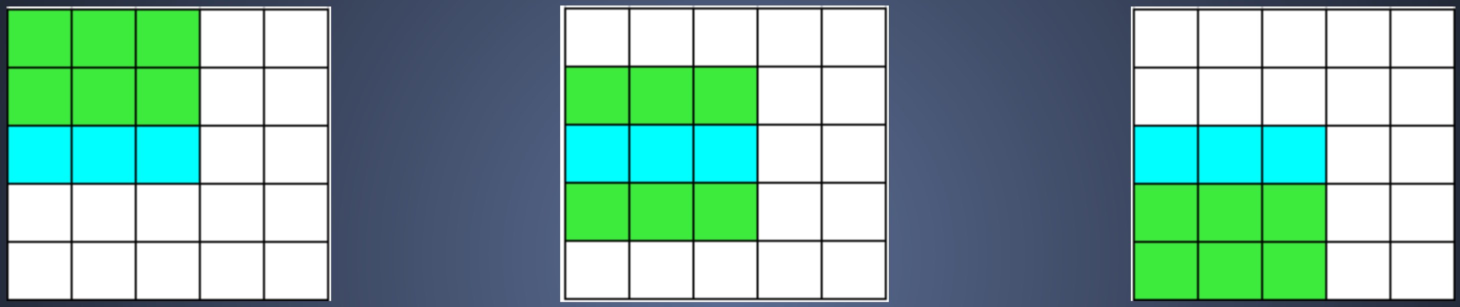
Time Complexity: $O(N^2M^2)$

Expected score: 40

Solution 1(Intuitive Solution)

```
for i = 1 -> n-m+1
  for j = 1 -> n-m+1
      for u = 0 -> m-1
        for v = 0 -> m-1
           if Garden[i+u][j+v] has apple
             ++count
```

Solution 2 - Observation



3 squares share row 3!

Solution 2 - Idea

- Precompute RowSum[i][j] = Garden[i][j] + Garden[i][j+1] + ... + Garden[i][j+m-1]
- Perform Solution 1 optimized with RowSum[][]

```
for i = 1 -> n-m+1
  for j = 1 -> n-m+1
      for u = 0 -> m-1
         count = count + Rowsum[i+u][j]
```

Time Complexity: $O(N^2M)$

Expected Score: 55

Solution 3 - Observation

- Most of the M x M Squares contain very few apples
- Most of the cells are empty
- Comparing with N^2 or M^2, K is relatively small

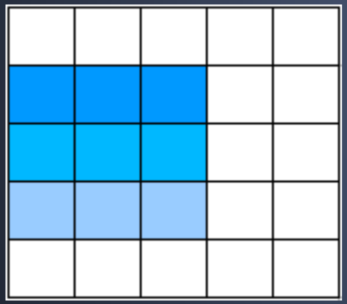
Solution 3 - Idea

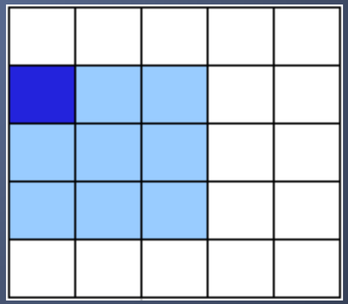
- Perform Solution 1
- Determine whether each apple is in the *M* x *M* square rather than check every cell

```
for i = 1 -> n-m+1
   for j = 1 -> n-m+1
        for x = 1 -> k
           if the x^th apple is inside (i,j)..(i+m-1,j+m-1)
              ++count;
```

Solution 4 - Observation

- Back to Solution 2...
  - Can it be faster?

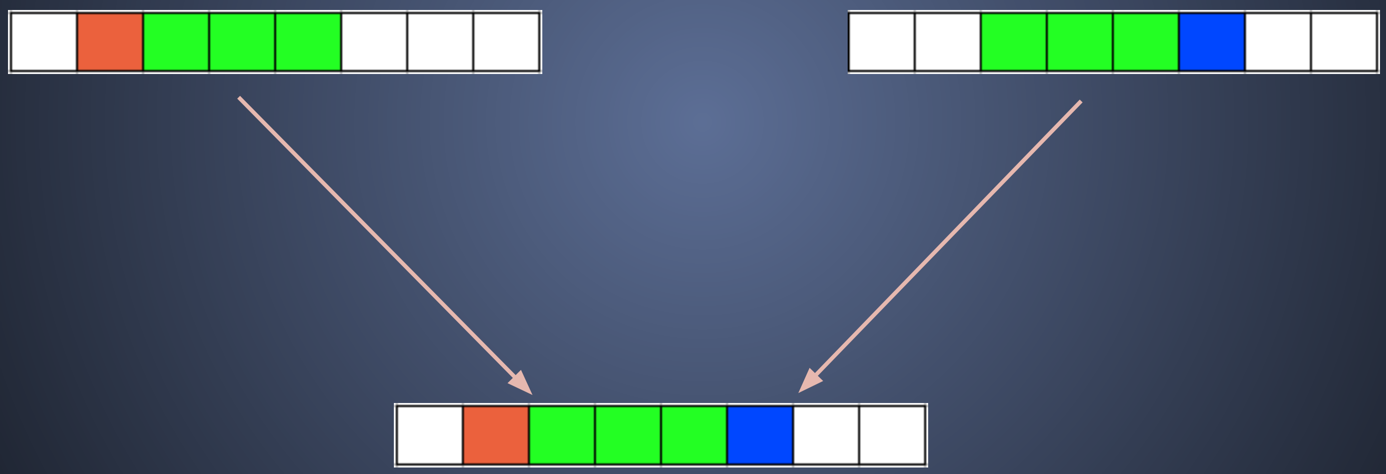




- Precompute SqrSum[][] by summing up Rowsum[][]

Solution 4 - Observation

- To achieve higher score, we need to speed up the precompute process



Rowsum[i][j] = Rowsum[i][j-1] - Garden[i][j-1] + Garden[i][j+m-1]

Solution 4 - Idea

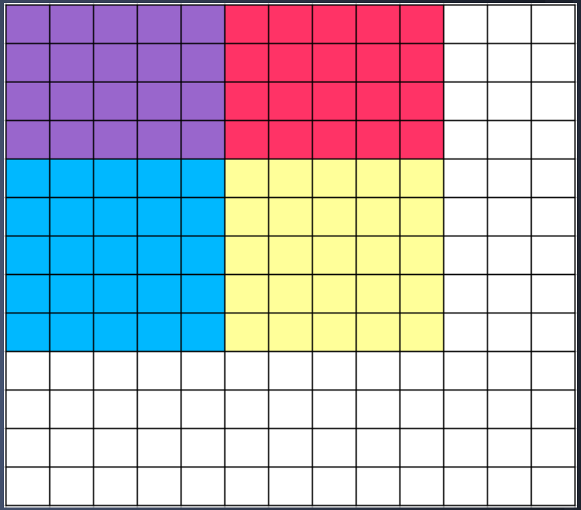
- Precompute Rowsum[][]
- Precompute Sqrsum[][]
  - Sqrsum[i][j] = Sqrsum[i-1][j] - Rowsum[i-1][j] + Rowsum[i+m-1][j]
- Find the maximum in Sqrsum[][]

Time complexity: $O(N^2)$

Expected Score: 70

Solution 5(Out of Syllabus)

- Inclusion-exclusion principle



Sqrsum[i][j] = Sqrsum[i-1][j] + Sqrsum[i][j-1] - Sqr[i-1][j-1] + Garden[i][j]

Ans = Sqrsum[i][j] - Sqrsum[i-m][j] - Sqr[i][j-m] + Sqrsum[i-m][j-m]

Time Complexity: $O(N^2)$

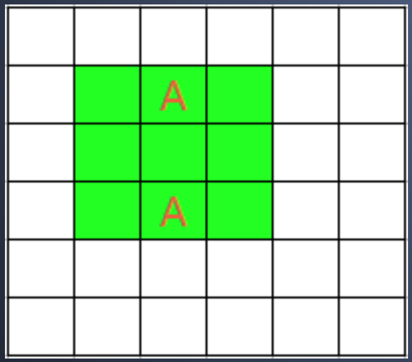
Expected Score: 70

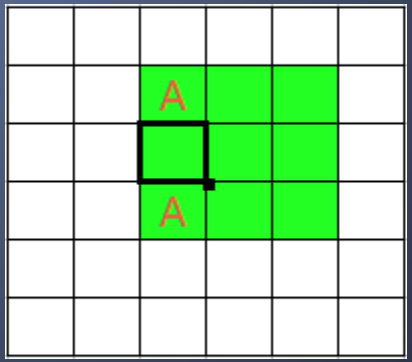
Solution 6 - Observation

- Cannot obtain full mark using solution related to N and M
- Try to think of some solutions related to K

Solution 6 - Observation

- One of the optimal ways to select the square:
  - at least one apples on the leftmost column
  - at least one apples on the top row





Solution 6 - Idea

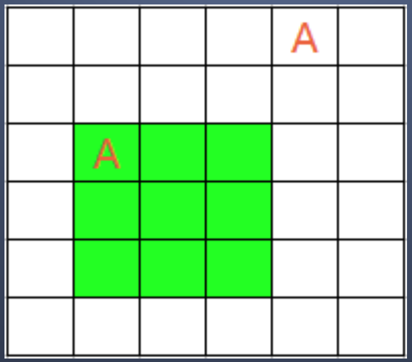
- Try all possible leftmost columns
- Try all possible top rows
- Determine whether each apple is inside the square

Time Complexity: O(K^3)

Expected Score: 70

Solution 7 - Observation

- Exhausting 2 edges is time consuming
  - some combinations are impossible
- Just exhaust the top rows which are possible for the leftmost column being tried from top to bottom

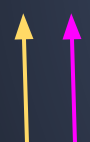


Solution 7 - Observation

Consider the following 1-D case

$M = 6$

1	5	6	8	12	13	15



Diff < 6

Solution 7 - Observation

Consider the following 1-D case

M = 6

1	5	6	8	12	13	15

Diff < 6

Solution 7 - Observation

Consider the following 1-D case

$M = 6$

1	5	6	8	12	13	15

Diff < 6

Solution 7 - Observation

Consider the following 1-D case

$M = 6$

1	5	6	8	12	13	15

Diff > 6

count = 3

Solution 7 - Observation

Consider the following 1-D case

M = 6

1	5	6	8	12	13	15

Diff < 6

Solution 7 - Observation

Consider the following 1-D case

M = 6

1	5	6	8	12	13	15

Diff > 6

count = 3

Solution 7 - Observation

Consider the following 1-D case

M = 6

1	5	6	8	12	13	15

Diff > 6

count = 2

Solution 7 - Observation

Consider the following 1-D case

M = 6

1	5	6	8	12	13	15

Diff < 6

And so on....

Solution 7 - Idea

- When we fix the leftmost column, the problem is reduced to 1-D case
- Only consider apples planted between the leftmost column and the rightmost column
- Apples should be arranged from the bottom to the top
  - That's why input data are sorted :)

Solution 7 - Idea

- Exhaust leftmost columns
- For each column exhausted, screen out the apples needed to consider
- Apply 1-D case method(Greedy)
- Find the maximum

Solution 7 - Time Complexity

- Exhaust leftmost column - O(K)
- Screening and Greedy - O(K)
  - each of the 2 pointers only goes through each apple once

Time complexity: O(K^2)

Expected Score: 100

Other Solutions

- Inclusion-exclusion Principle with discretization - $O(K^2)$
- Segment Tree - $O(K \lg K)$
- Other reasonable solutions

Expected Score: 100

Thank You