Contest debriefing

Scientific Committee
Result at 4\textsuperscript{th} hour

Total: 57 teams

<table>
<thead>
<tr>
<th>First 4 hours only</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved / Tries</td>
<td>10/44 (22%)</td>
<td>0/3 (0%)</td>
<td>42/93 (45%)</td>
<td>3/19 (15%)</td>
<td>55/105 (52%)</td>
<td>52/103 (50%)</td>
<td>26/166 (15%)</td>
<td>5/27 (18%)</td>
<td>7/18 (38%)</td>
<td>17/57 (29%)</td>
<td>0/1 (0%)</td>
</tr>
<tr>
<td>Average tries</td>
<td>2.32</td>
<td>3</td>
<td>1.82</td>
<td>1.73</td>
<td>1.84</td>
<td>1.81</td>
<td>3.19</td>
<td>1.93</td>
<td>1.5</td>
<td>2.28</td>
<td>1</td>
</tr>
<tr>
<td>Averages tries to solve</td>
<td>3</td>
<td>--</td>
<td>1.79</td>
<td>1.33</td>
<td>1.62</td>
<td>1.71</td>
<td>2.5</td>
<td>1.6</td>
<td>1.29</td>
<td>2.35</td>
<td>--</td>
</tr>
</tbody>
</table>

- Problem J: Association of Cats and Magical Lights
- Problem H: Association for Convex Main Office
- Problem D: Association of Computer Maintenance
- Problem B: Association for Cool Machineries (Part 2)
- Problem I: Apples, Cherries, and Mangos
- Problem K: Association of Camera Makers

(For problems C, E, F, G, J, please listen to the online commentary by Nathan and Jonathan. [https://www.youtube.com/watch?v=07z0lZMvpaQ](https://www.youtube.com/watch?v=07z0lZMvpaQ). Or search “2015 ACM ICPC Singapore Regional Live Commentary” in youtube.com)
Association of Cats and Magical Lights

Problem J
Problem

• Input: A rooted tree of N nodes
  – Color of node $u$ is $C_u$ (1 to 100)
  – Parent of node $u$ is $P_u$

• For a subtree rooted at node $u$, a color $\alpha$ is a magic color if the subtree has odd number of color $\alpha$

• Query($u$): Compute the number of magic colors of a node $u$

• Update($\alpha$, $u$): Change the color of the node $u$ to $\alpha$
Example

• Query(b)=1
  – black is odd & white is even
• Query(c)=2
  – both black and white are odd
• Update(red, f)
• Query(b)=3
  – black, white and red are odd
Simple solution

• For each query Query(u),
  – directly count the number of colors below node u
  – Report the number of colors whose counts are odd

• For each update Update(c, u),
  – Directly update the color of the node u

• This solution is slow
Flatten the tree

• Assign DFS order to the tree

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>a</td>
<td>b</td>
<td>e</td>
<td>f</td>
<td>c</td>
<td>g</td>
<td>d</td>
</tr>
</tbody>
</table>

• Every subtree rooted at some node can be represented as an interval

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>
Store each color as a modified Fenwick tree

- Fenwick tree allows us to find range sum and update in $O(\log N)$ time
- It can be modified to answer range parity
- Since we have 100 colors, the query time and update time is $O(\log N)$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>a</td>
<td>b</td>
<td>e</td>
<td>f</td>
<td>c</td>
<td>g</td>
<td>d</td>
</tr>
<tr>
<td>White</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Black</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Red</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Build modified Fenwick tree
Example

• Query(b) is the sum of range parity of [2 4]
  – White: 1
  – Black: 1
  – Red: 1
  – Ans: 3
Additional note

• Our intended solution is to represent 100 bits as 2 long long (2 * 64bits).
• Then, build a Fenwick tree for the 2 long long.
• Then, we just need to make one Fenwick tree query.

• However, Java version for Fenwick tree of 2 long long is slower than querying 100 Fenwick trees.

• So, we accept both solutions.
Association for Convex Main Office

Problem H
Problem

• Input: An integer \( N \) (\( N \leq 400,000 \))
• Output: \( N \) pairs of 2D coordinates \((x_i, y_i)\) that form a convex hull
  – such that \( 0 \leq x_i, y_i \leq 4 \times 10^7 \).
  – No three points are co-linear

• Example: \( N=4 \)
How to generate a convex office?

- Example: N=16
- We form a set of N/4 right-angle triangles, all have different slope.
- Arrange the triangles in decreasing slopes.
- Create mirror-image.
- Then, a convex office with N vertices is formed.

- Question: How to generate triangles of different slopes?
How to generate triangles of different slopes?

• Simple solution:

• This solution works for small N.
• When N > 20000, the width/height of all triangles > 4x10^7.
How to generate triangles of different slopes? (II)

- Generate triangle with the shortest height + width first.
  - During the generation, need to ensure the height and the width are co-prime.
  - (This guarantees that the slopes of all triangles are different.)
  - E.g. We will not generate (4, 2) since (4, 2) and (2, 1) have the same slope.

- 2: (1, 1),
- 3: (1, 2), (2, 1),
- 4: (1, 3), (3, 1),
- 5: (1, 4), (4, 1), (2, 3), (3, 2),
- 6: (1, 5), (5, 1),
- 7: (1, 6), (6, 1), (2, 5), (5, 2), (3, 4), (4, 3),
- ......
Note

• This is a rare question in ICPC, which asks for corner case.

• But this type of questions is getting popular in other competitions.

• We hope that the students can develop skill set in this aspect.
Association of Computer Maintenance

Problem D
Problem

• Input:
  – The prime factorization of K
  – (Constraint: Number of divisors of K is \( \sim 10^{10} \)).

• Output: \( f(A) \mod (10^9 + 7) \)
  – such that integer \( A \) minimizes \( f(A) = (A + K/A) \)

• Example: \( K = 2^3 \ast 7 \)
  – \( A=7 \) minimizes \( f(A)=A+K/A=7+8 \)
  – We output \( f(A)=7+8=15 \)
Observation

- To minimize $f(A) = A + K/A$, we choose an integer $A (< \sqrt{K})$ that is closest to $\sqrt{K}$.
- Proof: By differentiation,
  - $f'(A) = 1 - K/A^2 = 0 \Rightarrow A^2 = K$.
  - To minimize $f(A)$, we set $A = \sqrt{K}$ and $f(A) = 2\sqrt{K}$.
- However, $\sqrt{K}$ may not be an integer.
- Then, we need to choose an integer $A$ that is close to $\sqrt{K}$.
Brute-force solution

1. Let A=1;
2. For every divisor P of K,
   - If $A < P \leq \sqrt{K}$ then
     • set $A=P$;
3. Return $(A + K/A) \mod (10^9 + 7)$;

• Example: $K=5^2 \times 7^1$.
  - $\sqrt{K} = 13.23$
  - The list of divisors of $K$ is 1, 5, 7, 25, 35, 175.
  - So, $A$ is 7 $\rightarrow$ $A+K/A = 7+25 = 32$.

• This solution may be slow since there are $10^{10}$ divisors for $K$. 
A technique that requires us to verify $\sim 10^5$ divisors

- Partition $K$ into two halves $K_1$ and $K_2$ such that $K = K_1 \times K_2$ and the number of divisors of $K_1$ and $K_2$ is $\sim 10^5$.
- For each divisor $x_i$ of $K_1$ from small to big,
  - Find the biggest divisor $y_i$ of $K_2$ such that $x_i \times y_i$ just smaller than $\sqrt{K}$
- Set $A$ to be the biggest $x_i \times y_i$
- Report $(A + K/A) \mod (10^9+7)$

- This solution can run in $O(N \log N)$ time where $N=10^5$. (See next slide)
Example

- Initialize $A=1$
- For $x_1=1$, $y_1=343 \rightarrow x_1 \cdot y_1 = 343$
- For $x_2=2$, $y_2=343 \rightarrow x_2 \cdot y_2 = 686$
- For $x_3=3$, $y_3=245 \rightarrow x_3 \cdot y_3 = 735$
- For $x_4=4$, $y_4=49 \rightarrow x_4 \cdot y_4 = 196$
- For $x_5=5$, $y_5=49 \rightarrow x_5 \cdot y_5 = 294$
- For $x_6=6$, $y_6=49 \rightarrow x_6 \cdot y_6 = 392$
- For $x_7=9$, $y_7=49 \rightarrow x_7 \cdot y_7 = 441$
- For $x_8=12$, $y_8=49 \rightarrow x_8 \cdot y_8 = 588$
- For $x_9=18$, $y_9=49 \rightarrow x_9 \cdot y_9 = 882$
- For $x_{10}=24$, $y_{10}=35 \rightarrow x_{10} \cdot y_{10} = 840$
- For $x_{11}=36$, $y_{11}=7 \rightarrow x_{11} \cdot y_{11} = 252$
- For $x_{12}=72$, $y_{12}=7 \rightarrow x_{12} \cdot y_{12} = 504$

The biggest is $A=x_9 \cdot y_9 = 882 = 2 \cdot 3^2 \cdot 7^2$.

$K/A = 2^2 \cdot 5^1 \cdot 7^2 = 980$.

$A + K/A = 882 + 980 = 1862$. 

- $K = 2^3 \cdot 3^2 \cdot 5^1 \cdot 7^4$ 
  - there are $(3+1)(2+1)(1+1)(4+1)=120$ (distinct) divisors.
- Set $K_1=2^3 \cdot 3^2$ and $K_2=5^1 \cdot 7^4$.
- $K_1$ has $(3+1)(2+1)=12$ divisors.
- $K_2$ has $(1+1)(4+1)=10$ divisors.
- $\sqrt{K} = 929.71$
Handle big number

- Multiplication of big number is slow.
- Solution: Use logarithm
  - Replace $X \times Y$ by $\log X + \log Y$
- It reduces the running time.
Association for Cool Machineries (Part 2)

Problem B
The problem for part 1

• Give a $N \times N$ grid and a sequence of $<, >, ^, \downarrow$
• Output $X$, which is the smallest repetition trail
• Example program: $^\uparrow \downarrow > ^\uparrow <$

The smallest repetition trail is of length 4
The problem for part 2

• Design
  – a 200x200 grid and
  – a sequence of $<,>,^,\downarrow$

• such that the smallest repetition trail is of length $> 10^6$
Idea

• Design a sequence (say, $vv<<<<^v^v^v^v^v^v^v^v$) and walls that allows the robot to move up, down, left and right.
• E.g.

```
#c#   ##  #
#b#   #  a#
#a#   ##  #
  #    #  b#
  #    ##  #
   #    #  c#
```

• To make the robot move many steps, we design a difficult map.
• To make the robot move more, append $^v^v...^v$ to the end of the sequence.
  – E.g. $vv<<<<^v^v^v^v^v^v^v^v...^v$
A difficult map for 12x12 grid

- For the 12x12 grid,
  - \(a \rightarrow b\): 5 steps
  - \(b \rightarrow c\): 6 (=n-6) steps
  - \(c \rightarrow h\): \(7+6*4 = 31\) steps
    - \(c \rightarrow d\): 7 (=n-5) steps \(\leftarrow (n-11)\) times
    - \(d \rightarrow e\): 6 (=n-6) steps
    - \(e \rightarrow f\): 6 steps
    - \(f \rightarrow g, g \rightarrow h\): 6 (=n-6) steps \(\leftarrow (n-8)/2\) times
  - \(h \rightarrow i\): 10 (=n-2) steps
  - \(i \rightarrow n\): 31 steps
  - \(n \rightarrow a\): \(6+7+3*6 = 31\) steps
    - \(n \rightarrow o\): 6 (=n-6) steps
    - \(o \rightarrow p\): 7 (=n-5) steps
    - \(p \rightarrow q, q \rightarrow r, r \rightarrow a\): 6 (=n-6) steps \(\leftarrow (n-6)/2\) times
  - \(a \rightarrow b \rightarrow \ldots \rightarrow q \rightarrow r \rightarrow a: 5+6+31+10+31+31=114\) steps

- In general, the number of steps is
  - \(5+(n-6)+\left[ (n-11)(n-5)+(n-6)+6+(n-6)(n-8)/2 \right]*(n-2)/5 + (n-2)(n-7)/5 + [ (n-6) + (n-5) + (n-6)*(n-6)/2 ],
  - which is \(O(n^3)\).
Generalize the nxn grid

• For the 22x22 grid,
  – by the previous formula, the robot needs to use 1,526 steps.

• For the 192x192 grid,
  – by the previous formula, the robot needs to use 1,968,630 steps.
Note

• This is just one solution.
• You may find another solution.

• This is similar to convex office.
• The question asks for designing a corner test case.
• This is an important problem solving technique that is rarely tested in ICPC.
Apples, Cherries, and Mangos

Problem I
Problem

• WLOG, assume $A \geq C \geq M$
• We need to arrange them so that adjacent fruits are different
• Example: $A=2$, $C=1$, $M=1$
Solution: DP

- \( V(A, C, M) = \) no of valid ways to allocate all fruits
- \( V_A(A, C, M) = \) no of valid ways to allocate all fruits given that the first fruit is Apple
- \( V_C(A, C, M) = \) no of valid ways to allocate all fruits given that the first fruit is Cherry
- \( V_M(A, C, M) = \) no of valid ways to allocate all fruits given that the first fruit is Mango

- Base cases:
  - \( V_A(1, 0, 0) = 1, V_C(0, 1, 0) = 1, V_M(0, 0, 1) = 1 \)
  - \( V_w(x, y, z) = 0 \) if \( x<0 \) or \( y<0 \) or \( z<0 \)

- Recursive cases:
  - \( V_A(A, C, M) = V_C(A-1, C, M) + V_M(A-1, C, M) \)
  - \( V_C(A, C, M) = V_A(A, C-1, M) + V_M(A, C-1, M) \)
  - \( V_M(A, C, M) = V_A(A, C, M-1) + V_C(A, C, M-1) \)

- This solution runs in \( O(A * C * M) \)
- It is too slow when the number of fruits is close to 200,000
Valid arrangement

- WLOG, assume $A \geq C \geq M$
- For any valid arrangement, apples partitions the sequence into $A+1$ bins
- Every bin must be some cherries or mangos
  - Except for the first and the last bins
- Depending on whether first and/or last bins are empty, we have 4 cases

<table>
<thead>
<tr>
<th>A+1 bins:</th>
<th></th>
<th></th>
<th></th>
<th>A bins:</th>
<th></th>
<th></th>
<th>A bins:</th>
<th>A-1 bins:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
<td>^</td>
</tr>
<tr>
<td></td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
<td>......</td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
</tr>
<tr>
<td></td>
<td>🍑</td>
<td>🍑</td>
<td>🍑</td>
<td>......</td>
<td>🍑</td>
<td>🍑</td>
<td>🍑</td>
<td>🍑</td>
</tr>
<tr>
<td></td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
<td>......</td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
</tr>
<tr>
<td></td>
<td>🍑</td>
<td>🍑</td>
<td>🍑</td>
<td>......</td>
<td>🍑</td>
<td>🍑</td>
<td>🍑</td>
<td>🍑</td>
</tr>
<tr>
<td></td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
<td>......</td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
<td>🍎</td>
</tr>
</tbody>
</table>
Number of valid arrangements of A,C,M

- Denote $\text{count}_{c,m}(k)$ is the number of ways to arrange C cherries and M mangos into k bins such that adjacent fruits are different.

- **Theorem 1**: The number of valid arrangements of A apples, C cherries and M mangos is:
  
  \[ \text{count}_{c,m}(A - 1) + 2\text{count}_{c,m}(A) + \text{count}_{c,m}(A + 1) \]

<table>
<thead>
<tr>
<th>A+1 bins:</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A bins:</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A bins:</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A-1 bins:</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
<th>(^)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
<td>🍏</td>
</tr>
</tbody>
</table>
Valid arrangement for cherries and mangos in each bin

- Suppose we don’t have apple
- Assume we have c cherries and m mangos
- To have a valid arrangement, we need $c=m$ or $c=m+1$ or $c=m-1$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Same:</td>
<td>$c=m$</td>
<td>or</td>
<td>$c=m$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_major:</td>
<td>$c=m+1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_major:</td>
<td>$c=m-1$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 arrangements
1 arrangement
1 arrangement
How to distribute cherries and mangos into $k$ bins?

- **Theorem 2:** Assume $C < M$. The number of ways to distribute cherries and mangos into $k$ bins is $\text{count}_{C,M}(k) = \sum_{t_1=0}^{C} \left( \binom{k}{t_1, t_2, t_3} \left( \frac{C + t_2 - 1}{C - t_1 - t_3} \right) 2^{t_3} \right) \left| t_2 = M - C + t_1, t_3 = k - t_1 - t_2 \right.$

- Proof: Skip
Final algorithm

• By Theorems 1 and 2, we have the following algorithm

**Algorithm** ValidArrangement(A, C, M)

Input: Assume A > C > M

Return \( \text{count}_{C,M}(A - 1) + 2 \text{count}_{C,M}(A) + \text{count}_{C,M}(A + 1) \);

**Algorithm** Count\(_{C,M}(k)\)

Return \( \sum_{t_1=0}^{C} \left\{ \binom{k}{t_1, t_2, t_3} \left( \frac{C + t_2 - 1}{C - t_1 - t_3} \right) 2^{t_3} \mid t_2 = M - C + t_1, t_3 = k - t_1 - t_2 \right\} \);
Association of Camera Makers

Problem K
Association of Camera Makers

• **Input:**
  - A set of points \((X_1, Y_1), \ldots, (X_N, Y_N)\)
  - A threshold \(K\)

• **Output:**
  - The minimum radius \(R\) such that a circle of radius \(R\) that covers \(K\) points

• **Example:** Suppose \(K=4\).
  - Ans: \(R=2\)
Can we verify if a radius-R circle cover K points?

- VerifyRadius(R, K) is a function that returns true if a radius-R circle exists that covers K points
- Suppose there exists a radius-R circle that contains K points
  - Then, the radius-R circles of the K points should overlap
  - Any point in the overlapping region can be the center of the radius-R circle.
    - In particular, we can set any intersecting point as the center of the radius-R circle.

Example: R=4, K=4
Idea for VerifyRadius(R,K)

• Let \((X_i, Y_i)\) and \((X_j, Y_j)\) be any two points
• Let \(Q\) and \(Q'\) be the intersecting points of the radius-\(R\) circles of \((X_i, Y_i)\) and \((X_j, Y_j)\)
• If there exist \((K-2)\) other points whose distances from \(Q\) (or \(Q'\)) are less than \(R\), then
  – VerifyRadius(R, K) returns true.

Example: \(R=4, K=4\)
VerifyRadius(R, K)

Function VerifyRadius(R, K)
• For every pair of points (X_i, Y_i) and (X_j, Y_j),
  – If the radius-R circles of (X_i, Y_i) and (X_j, Y_j) overlap,
    • Let the intersecting points be Q and Q’
    • Check if there are (K-2) points whose distances from Q (or Q’) are less than R;
      • If yes, return true;
  • Return false;
• The running time is O(N^3);
Solution

• Note that 0 and $10^6$ are the lower bound and upper bound, respectively, of the radius $R$

• This problem can be solved by binary search using $\text{FindRadius}(0, 10^6)$

• $\text{FindRadius}(L, U)$
  – If (L and U are the same up to 2 decimal place) report L;
  – $M = (L + U)/2$;
  – If $\text{VerifyRadius}(M, K)$ is true,
    • $\text{FindRadius}(M, U)$;
  – Else
    • $\text{FindRadius}(L, M)$;
Still not good enough

• Previous solution runs in $O(N^3 \log 10^8) = O(27 \, N^3)$ time
• It can handle cases where $N<1000$
• Hence, it can solve 10 out of 16 test cases

• To solve all 16 test cases, please read the paper:
  – Jiri Matousek. On enclosing k points by a circle, 1995
  – Implementing this algorithm without an accelerating grid gives an O($N^2 \log^2 N$) solution. The full algorithm with the grid takes O($NK \log^2 K$) time
Acknowledgement
(related to question setting)

• Problem setters
  – Hubert Teo Hua Kian (Stanford University),
  – Irvan Jahja,
  – Mark Theng Kwang Hui (SG IOI),
  – Ranald Lam Yun Shao (SG IOI),
  – Dr Steven Halim,
  – Professor Sung Wing Kin, Ken,
  – Victor Loh Bo Huai (Facebook),
  – William Gan Wei Liang (SG IOI)

• Scientific committee (Tester)
  – Harta Wijaya (Garena),
  – Jonathan Irvin Gunawan,
  – Nathan Azaria

• Scientific committee (NUS staffs)
  – Associate Professor Chang Ee-Chien,
  – Associate Professor Hugh Anderson,
  – Dr Seth Lewis Gilbert,
  – Professor Frank Christian Stephan,
  – Associate Professor Leong Hon Wai

• Honorary Judges
  – Dr Felix Halim (Google),
  – Suhendry Effendy (ACM ICPC Jakarta Regional chief judge),
  – Trinh Tuan Phuong (Quantcast),
  – Fredrik Niemelä, Per Austrin, & Greg Hamerly (Kattis)