Contest debriefing

Scientific Committee

Result at 4th hour

Total: 57 teams

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

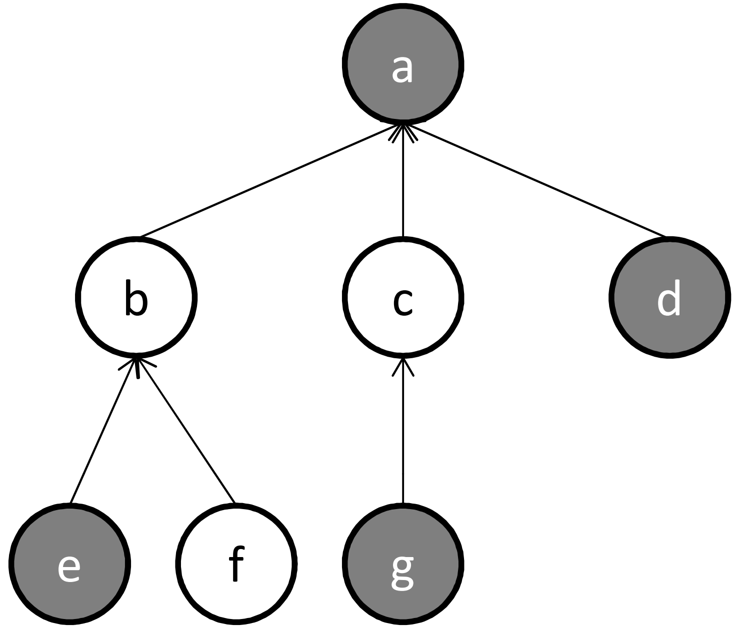
- Problem J: Association of Cats and Magical Lights
- Problem H: Association for Convex Main Office
- Problem D: Association of Computer Maintenance
- Problem B: Association for Cool Machineries (Part 2)
- Problem I: Apples, Cherries, and Mangos
- Problem K: Association of Camera Makers

- (For problems C, E, F, G, J, please listen to the online commentary by Nathan and Jonathan. https://www.youtube.com/watch?v=o7z0IZMvpaQ . Or search "2015 ACM ICPC Singapore Regional Live Commentary" in youtube.com)

Association of Cats and Magical Lights

Problem J

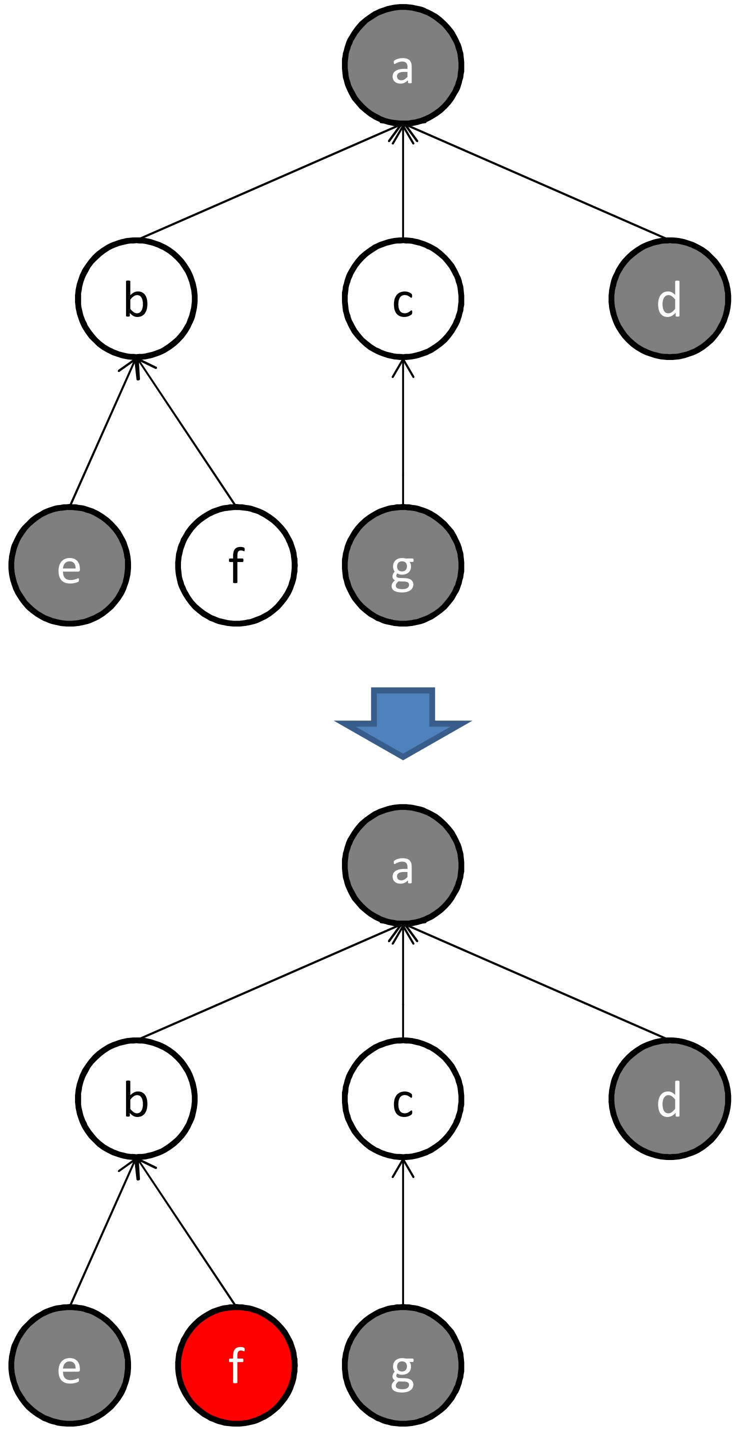
Problem



- 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 α is a magic color if the subtree has odd number of color α
- Query(u): Compute the number of magic colors of a node u
- Update(α, u): Change the color of the node u to α

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

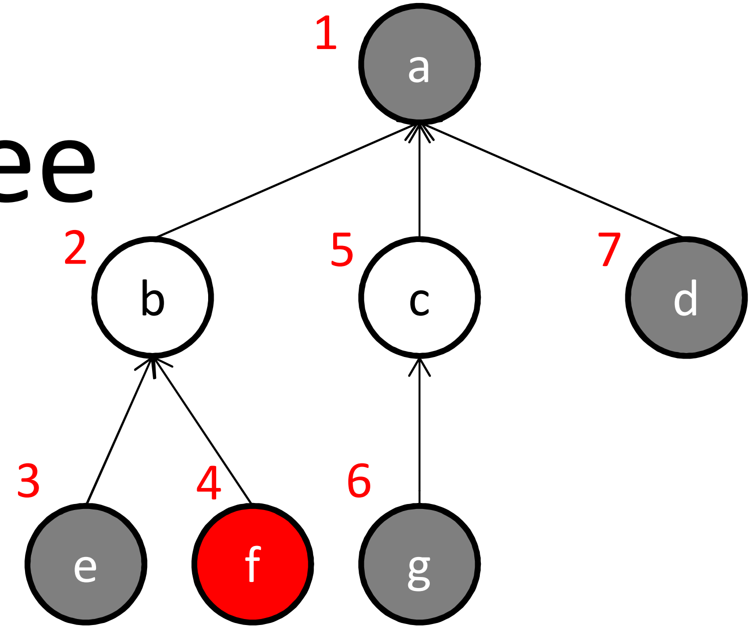


Simple solution

- 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



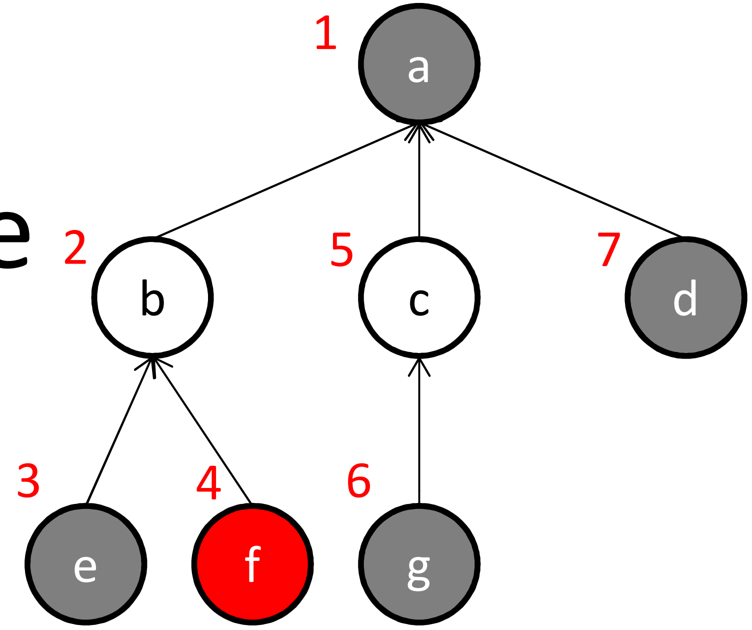
- Assign DFS order to the tree

	1	2	3	4	5	6	7
Node	a	b	e	f	c	g	d

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

	a	b	c	d	e	f	g
S	1	2	5	7	3	4	6
E	7	4	6	7	3	4	6

Store each color as a modified Fenwick tree



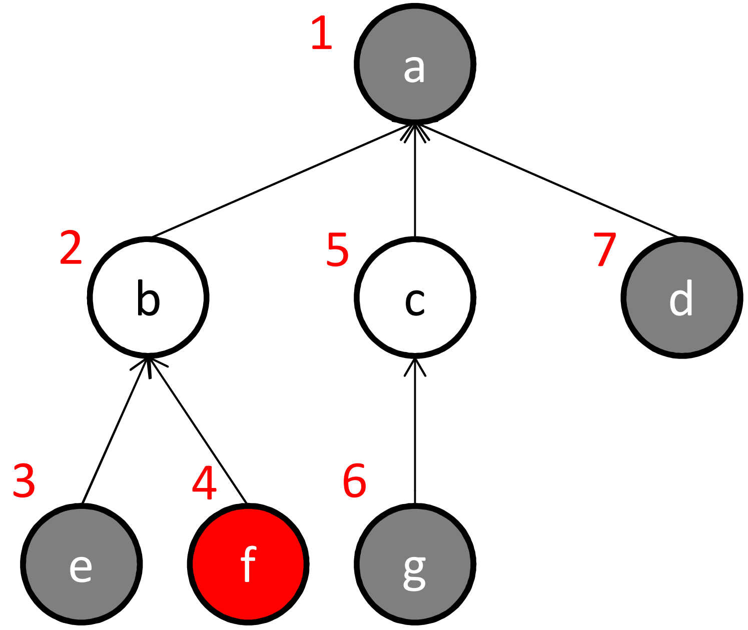
- Fenwick tree allows us to find range sum and update in O(log N) time
- It can be modified to answer range parity
- Since we have 100 colors, the query time and update time is O(log N)

	a	b	c	d	e	f	g
S	1	2	5	7	3	4	6
E	7	4	6	7	3	4	6

	1	2	3	4	5	6	7
Node	a	b	e	f	c	g	d
White	0	1	0	0	1	0	0
Black	1	0	1	0	0	1	1
Red	0	0	0	1	0	0	0

Build modified Fenwick tree

Example



- Query(b) is the sum of range parity of [2 4]
  - White: 1
  - Black: 1
  - Red: 1
  - **Ans: 3**

	a	b	c	d	e	f	g
S	1	2	5	7	3	4	6
E	7	4	6	7	3	4	6

	1	2	3	4	5	6	7
Node	a	b	e	f	c	g	d
White	0	1	0	0	1	0	0
Black	1	0	1	0	0	1	1
Red	0	0	0	1	0	0	0

Build modified Fenwick tree

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.

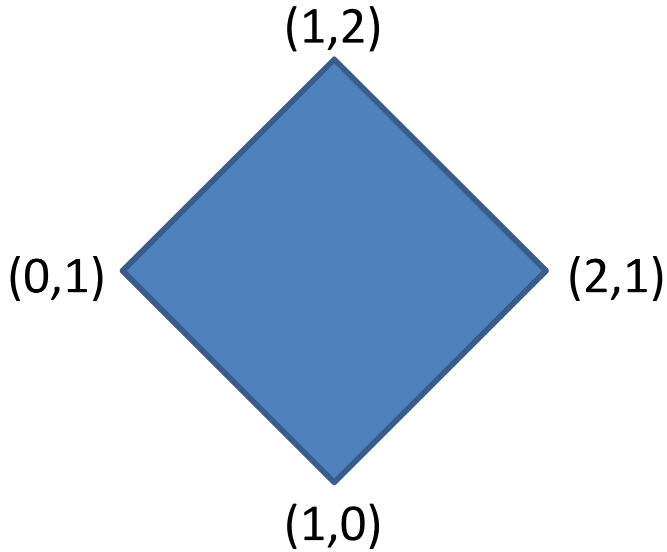
- So, we accept both solutions.

Association for Convex Main Office

Problem H

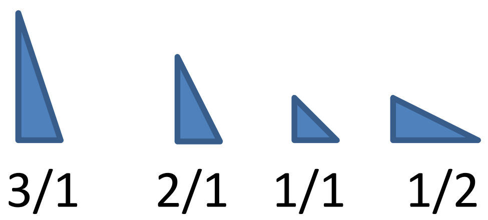
Problem

- 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



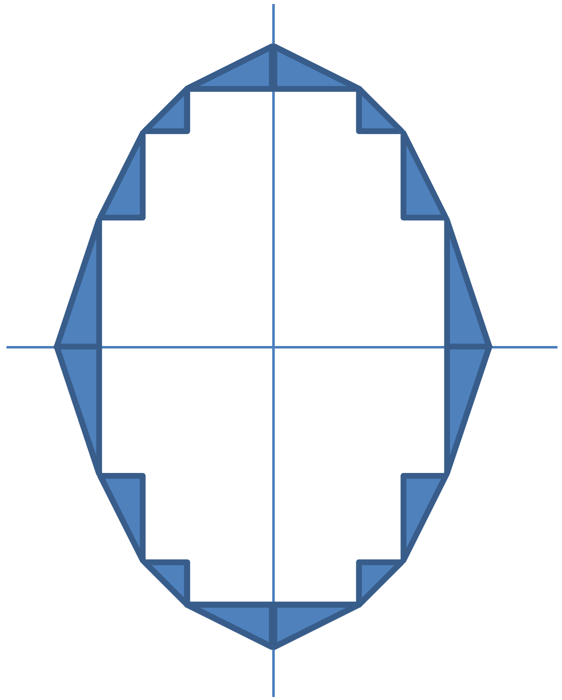
How to generate a convex office?

- Example: N=16
- We form a set of N/4 right-angle triangles, all have different slope.



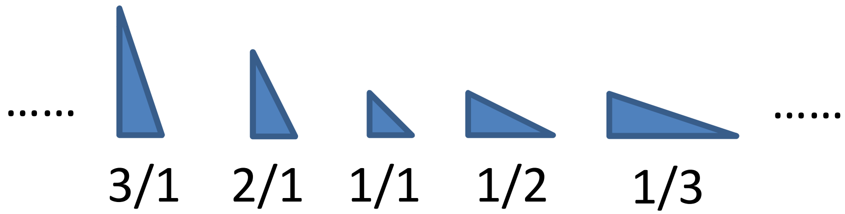
- Arrange the triangles in decreasing slopes.
- Create mirror-image.
- Then, a convex office with N vertices is formed

- Question: How to generate triangles of different slopes?

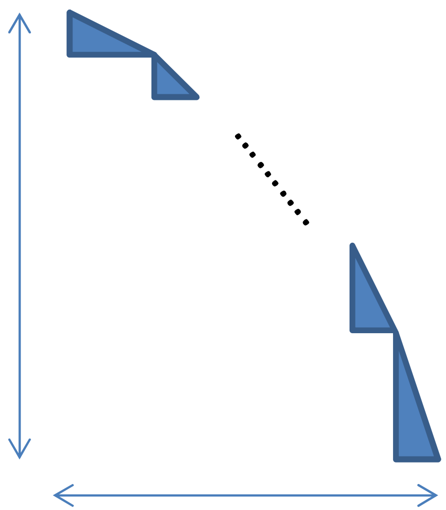


How to generate triangles of different slopes?

- Simple solution:

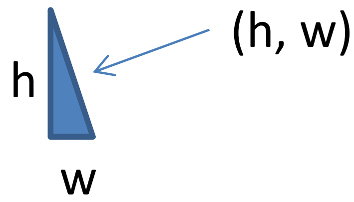


- This solution works for small N.
- When N>20000, the width/height of all triangles > $4x10^7$.



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

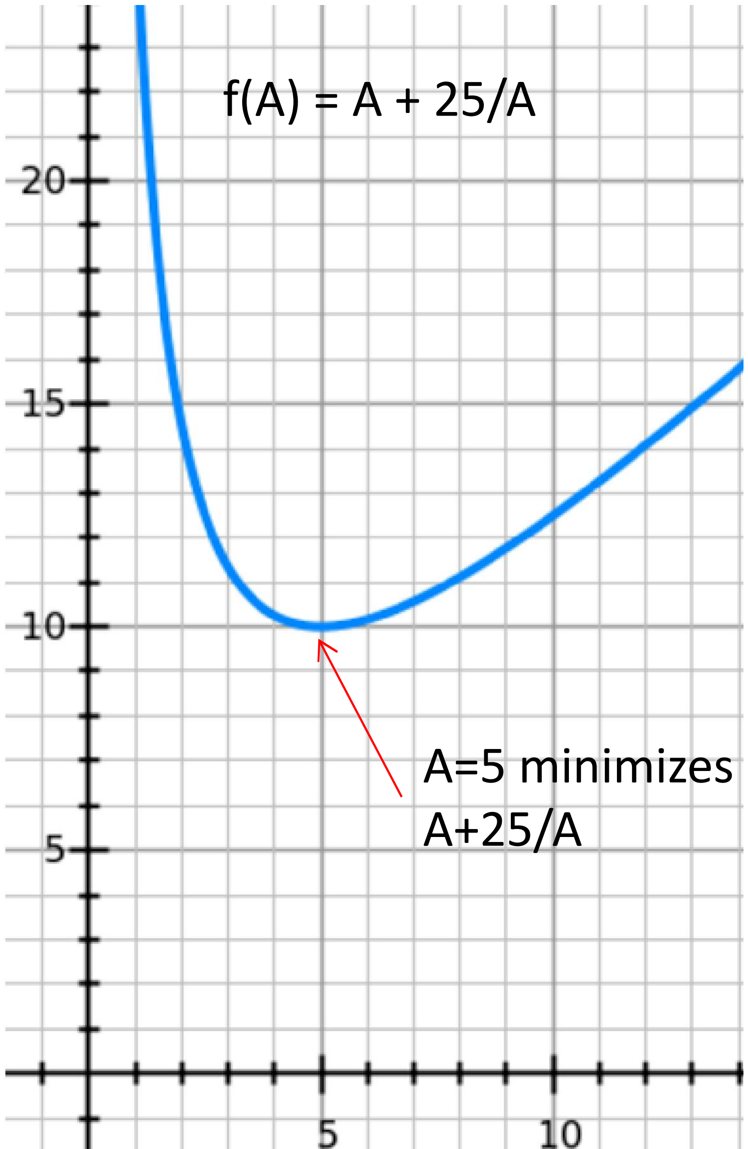
Problem

- Input:
  - The prime factorization of K
  - (Constraint: Number of divisors of K is ~ 10^{10}.)
- Output: f(*A) mod* $(10^9 + 7)$
  - such that integer *A* minimizes $f(A) = (A + K/A)$

- Example: $K = 2^3 * 7$
  - 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*7^1.
  - $\sqrt{K}$ = 13.23
  - The list of divisors of K is 1, 5, 7, 25, 35, 175.
  - So, A is 7 → A+K/A = 7+25 = 32.

- This solution may be slow since there are 10^{10} divisors for K.

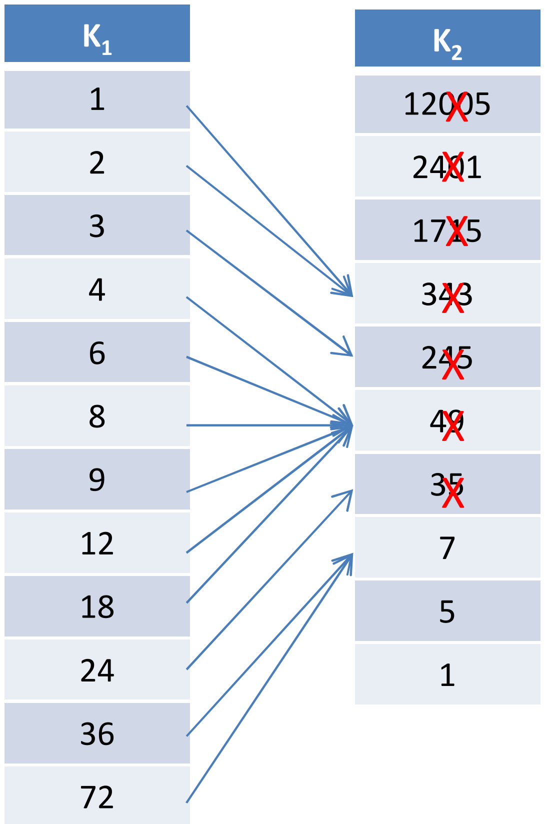
A techique that requires us to verify ~10^5 divisors

- Partition K into two halves K_1 and K_2 such that $K=K_1*K_2$ and the number of divisors of K_1 and K_2 is ~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*y_i just smaller than $\sqrt{K}$
- Set A to be the biggest x_i*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

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



- Initialize A=1
- For $x_1=1$, $y_1=343 \rightarrow x_1*y_1 = 343$
- For $x_2=2$, $y_2=343 \rightarrow x_2*y_2 = 686$
- For $x_3=3$, $y_3=245 \rightarrow x_3*y_3 = 735$
- For $x_4=4$, $y_4=49 \rightarrow x_4*y_4=196$
- For $x_5=6$, $y_5=49 \rightarrow x_5*y_5=294$
- For $x_6=8$, $y_6=49 \rightarrow x_6*y_6=392$
- For $x_7=9$, $y_7=49 \rightarrow x_7*y_7=441$
- For $x_8=12$, $y_8=49 \rightarrow x_8*y_8=588$
- For $x_9=18$, $y_9=49 \rightarrow x_9*y_9=882$
- For $x_{10}=24$, $y_{10}=35 \rightarrow x_{10}*y_{10}=840$
- For $x_{11}=36$, $y_{11}=7 \rightarrow x_{11}*y_{11}=252$
- For $x_{12}=72$, $y_{12}=7 \rightarrow x_{12}*y_{12}=504$

- The biggest is $A=x_9*y_9=882=2*3^2*7^2$. $K/A=2^2*5^1*7^2=980$.
- A + K/A = 882+980=1862.

Handle big number

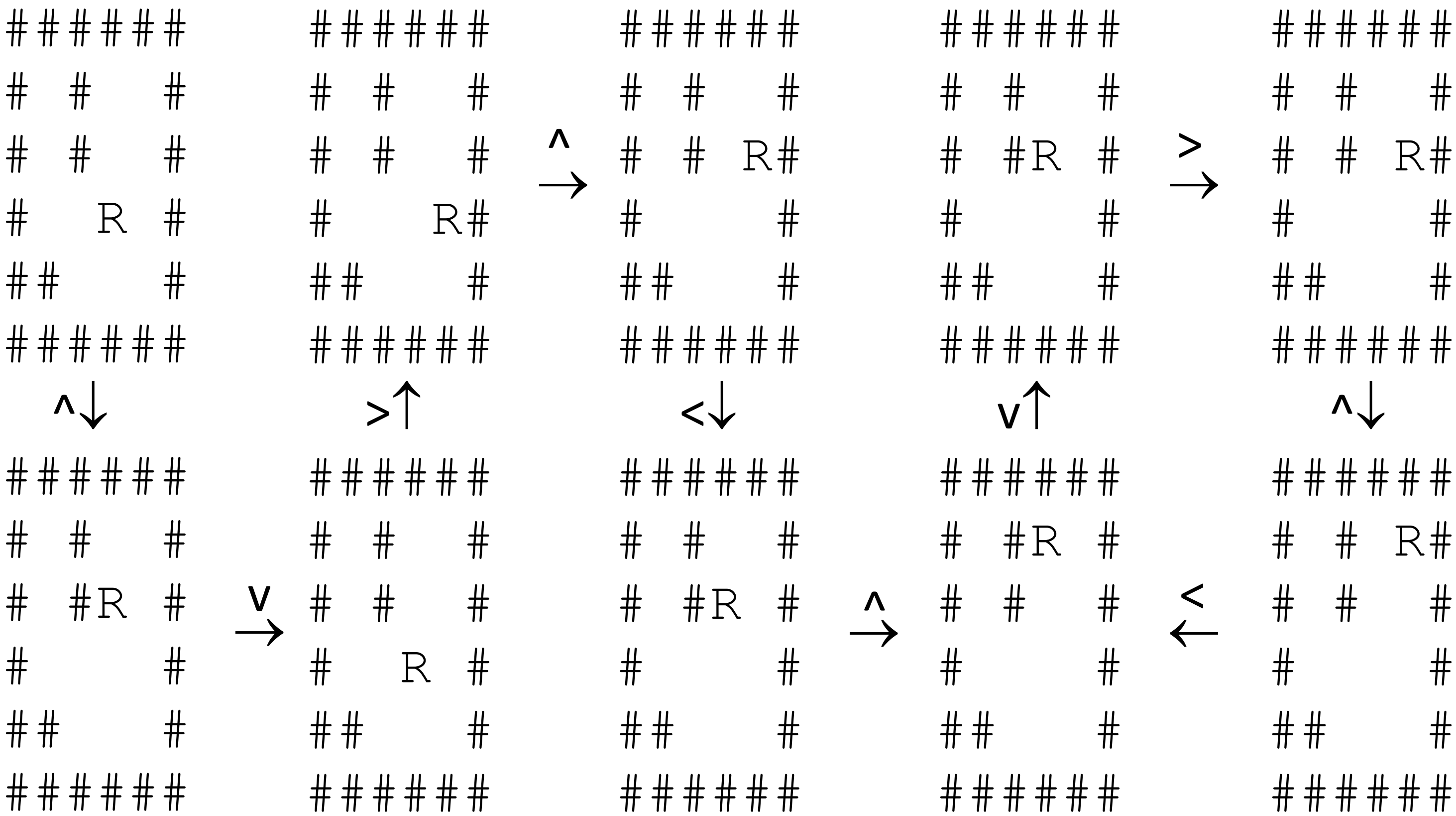
- Multiplication of big number is slow.
- Solution: Use logarithm
  - Replace X * Y by log X + log Y
- It reduces the running time.

Association for Cool Machineries (Part 2)

Problem B

The problem for part 1

- Give a NxN grid and a sequence of <,>,^,v
- Output X, which is the smallest repetition trail
- Example program: ^v>^<



The smallest repetition trail is of length 4

The problem for part 2

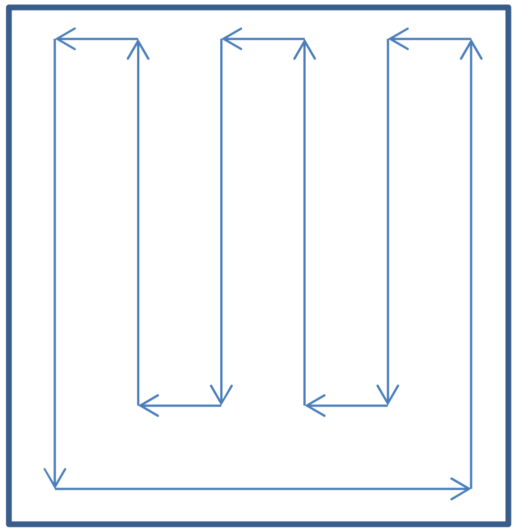
- Design
  - a 200x200 grid and
  - a sequence of <,>,^,v
- such that the smallest repetition trail is of length $> 10^6$

Idea

- Design a sequence (say, `vv<<<^^^>>`) 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`

A difficult map for 12x12 grid

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

  - a→b→…→q→r→a: 5+6+31+10+31+31=114 steps

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

```
12
vv<<<^^^>>v^
############
##l####f####
##   k#   e#
# m#j# g#d##
## #i## #c##
# n# # h# ##
## # ## # ##
# o#    # b#
## ###### ##
#  p q r a##
### # # # ##
############
```

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.

```
22
vv<<<^^^>>v^v^v^v^v^v^
######################
## #### #### #### ####
##    #    #    #    #
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  # #  # #  # #  # ##
## # ## # ## # ## # ##
#  #    #    #    #  #
## ################ ##
#                  R##
### # # # # # # # # ##
######################
```

Note

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

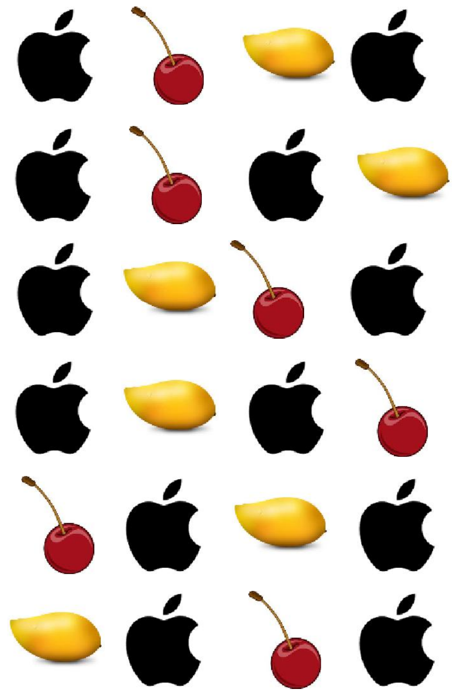
- This is similar to convex office.
- The question asks for designing a corner test case.
- This is an important problem solving technique that is rarely tested in ICPC.

Apples, Cherries, and Mangos

Problem I

Problem

- WLOG, assume A ≥ C ≥ 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

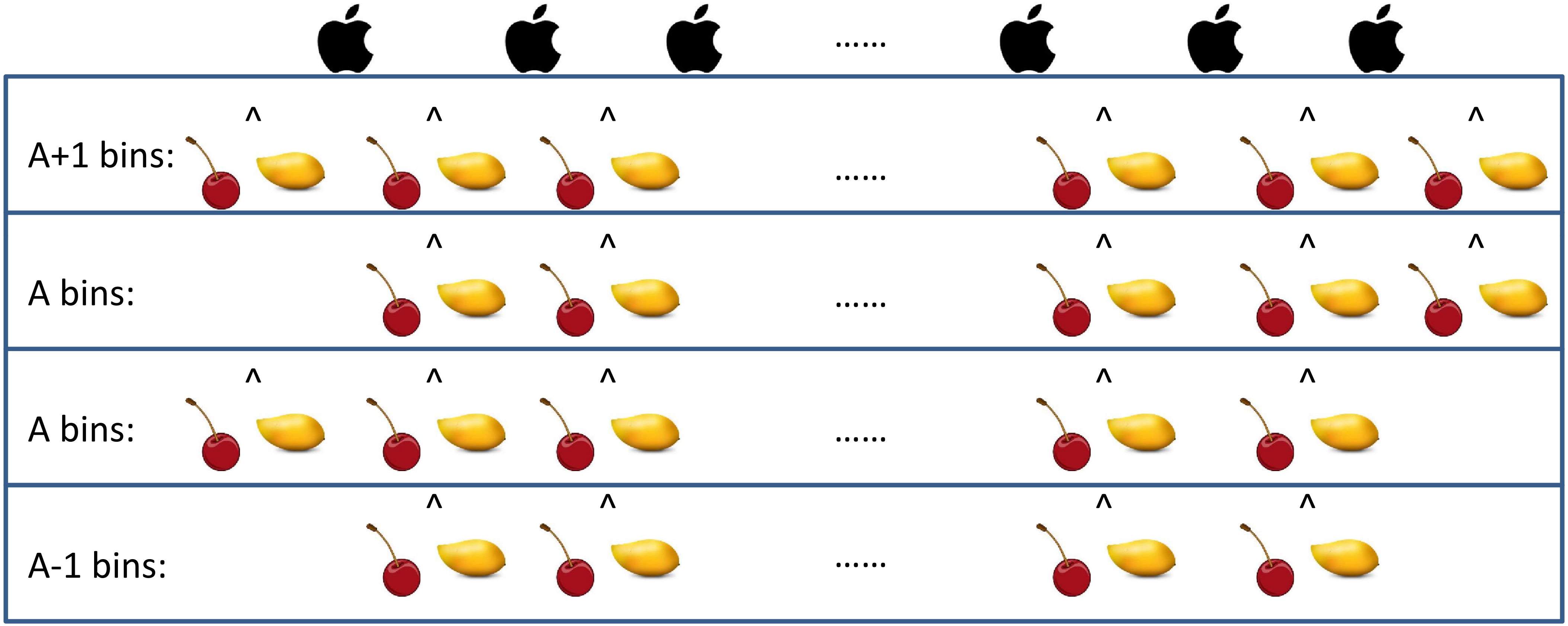
- Base cases:
  - $V_A(1, 0, 0) = 1$, $V_C(0, 1, 0) = 1$, $V_M(0, 0, 1) = 1$
  - $V_w(x, y, z) = 0$ if $x<0$ or $y<0$ or $z<0$

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

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

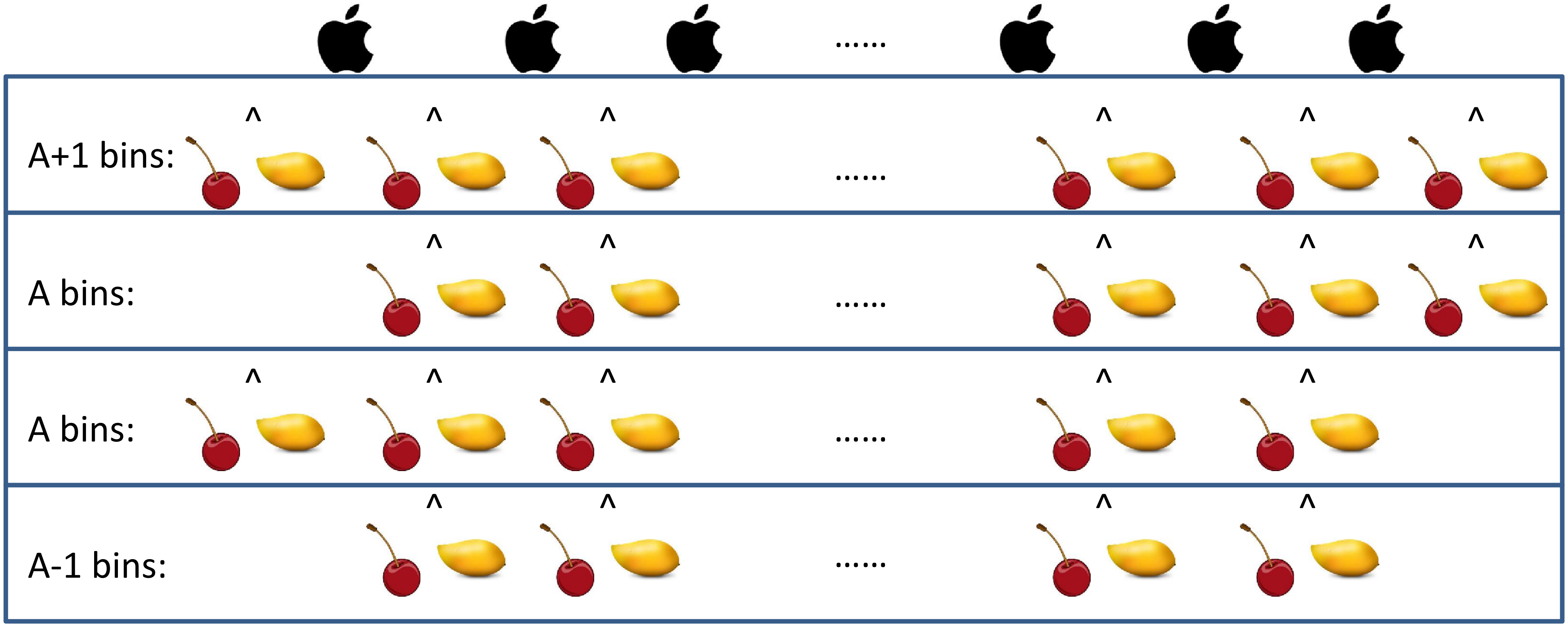
Valid arrangement

- WLOG, assume A ≥ C ≥ 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



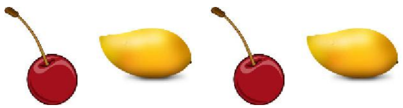
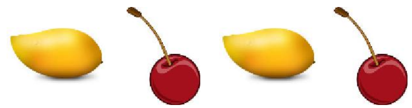
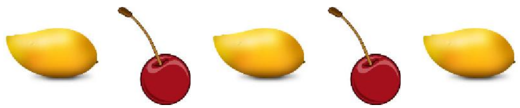
Number of valid arrangements of A,C,M

- Denote $count_{C,M}(k)$ 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:
  - $count_{C,M}(A-1) + 2count_{C,M}(A) + count_{C,M}(A+1)$



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

Same:	c=m	 or 	2 arrangements
C_major:	c=m+1		1 arrangement
M_major:	c=m-1		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 $count_{C,M}(k) =$
- $\sum_{t_1=0}^{C}\left\{\binom{k}{t_1,t_2,t_3}\binom{C+t_2-1}{C-t_1-t_3}2^{t_3}\middle| t_2 = M-C+t_1, t_3 = k-t_1-t_2\right\}$

- Proof: Skip

Final algorithm

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

Algorithm ValidArrangment(A, C, M)

Input: Assume A > C > M

Return $count_{C,M}(\mathrm{A}-1)+2count_{C,M}(\mathrm{A})+count_{C,M}(\mathrm{A}+1)$;

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

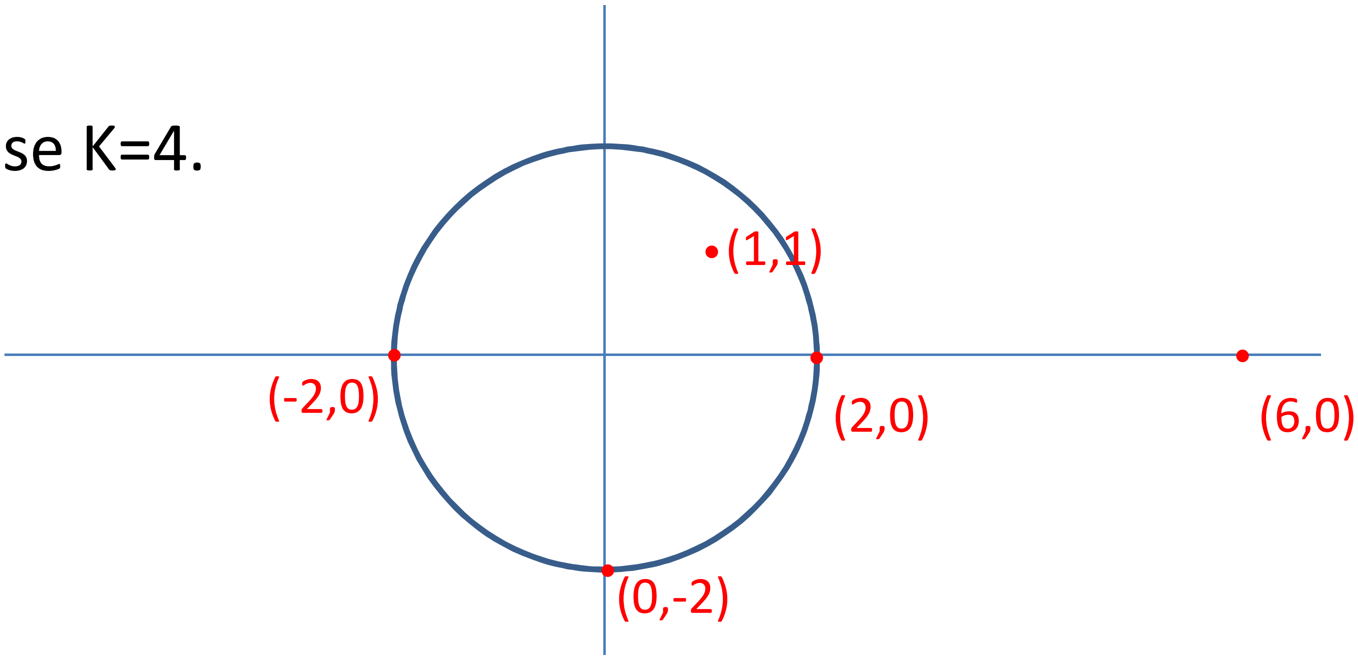
Return $\sum_{t_1=0}^{C}\left\{\binom{k}{t_1,t_2,t_3}\binom{C+t_2-1}{C-t_1-t_3}2^{t_3}\middle| t_2=M-C+t_1, t_3=k-t_1-t_2\right\}$;

Association of Camera Makers

Problem K

Association of Camera Makers

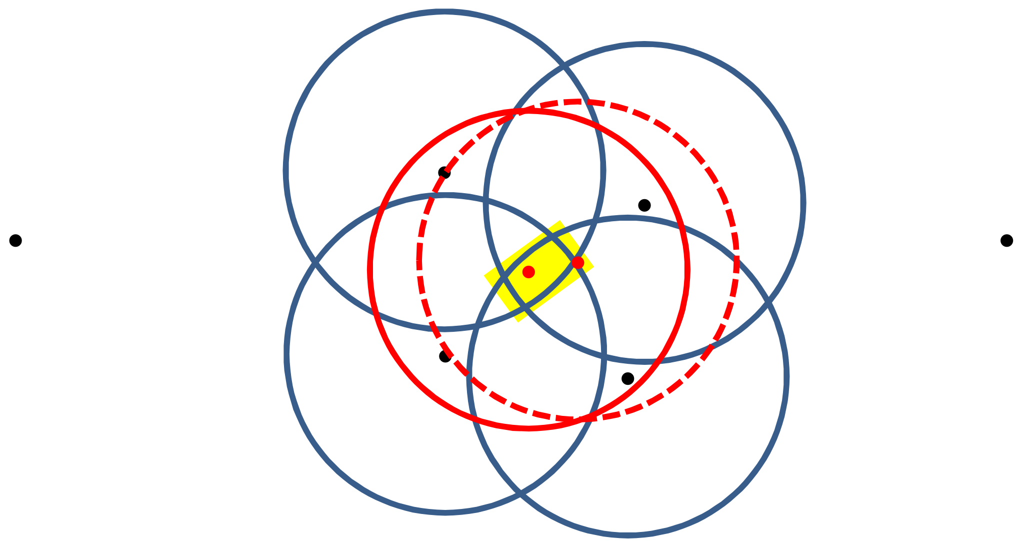
- Input:
  - A set of points (X_1,Y_1), …, (X_N, Y_N)
  - A threshold K
- Output:
  - The minimum radius R such that a circle of radius R that covers K points
- Example: Suppose K=4.
  - Ans: R=2



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.

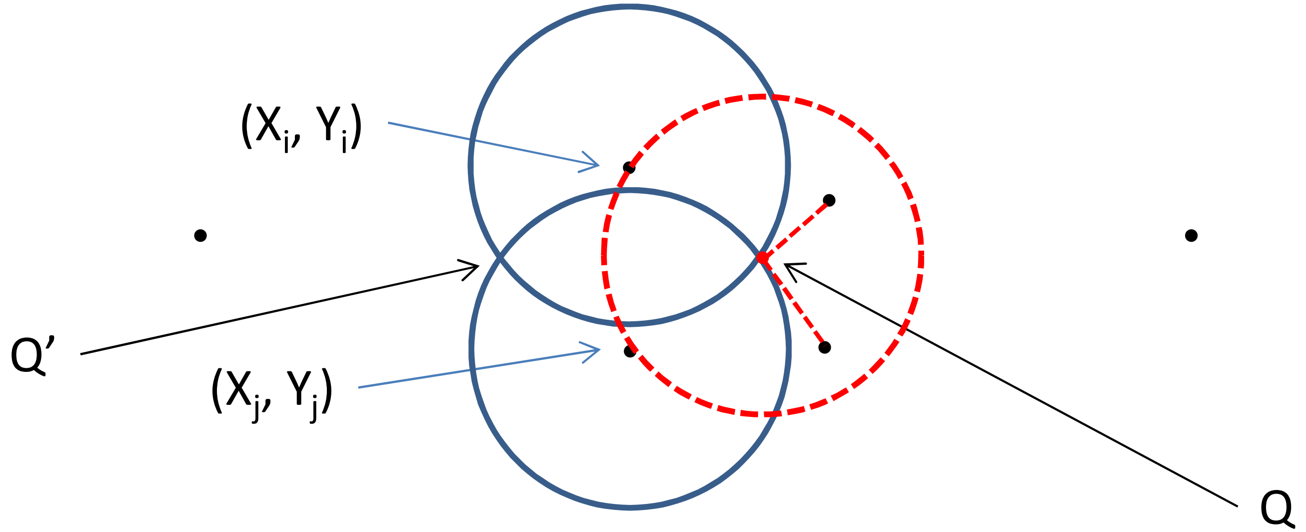
Example: R=4, K=4



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 FindRadius(0, 10^6)

- FindRadius(L, U)
  - If (L and U are the same up to 2 decimal place) report L;
  - M=(L+U)/2;
  - If **VerifyRadius**(M, K) is true,
    - FindRadius(M, U);
  - Else
    - 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)