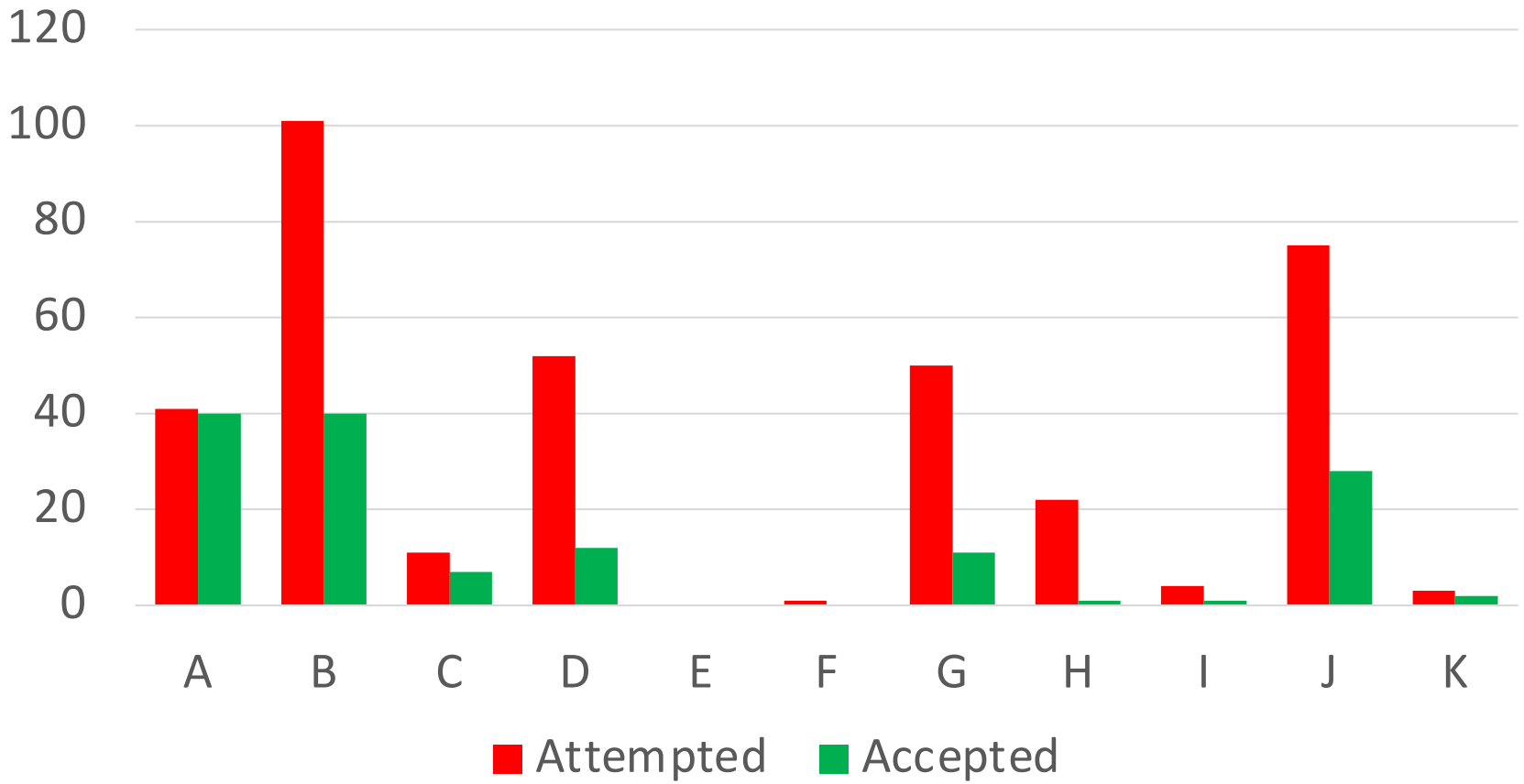
Commentaries on Problems

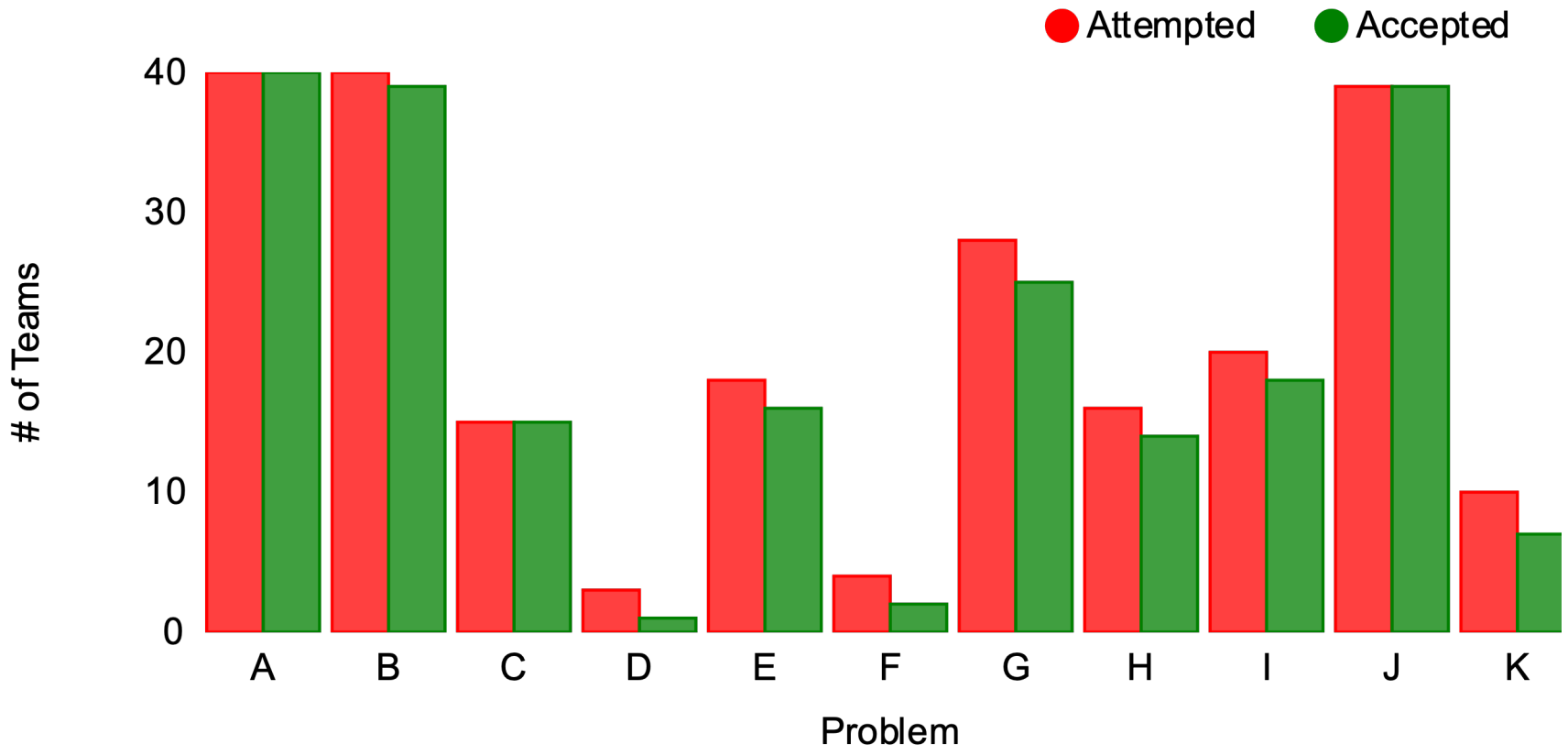
JUDGE TEAM
ICPC 2021 ASIA YOKOHAMA REGIONAL

Problem vs. #Teams @Freeze

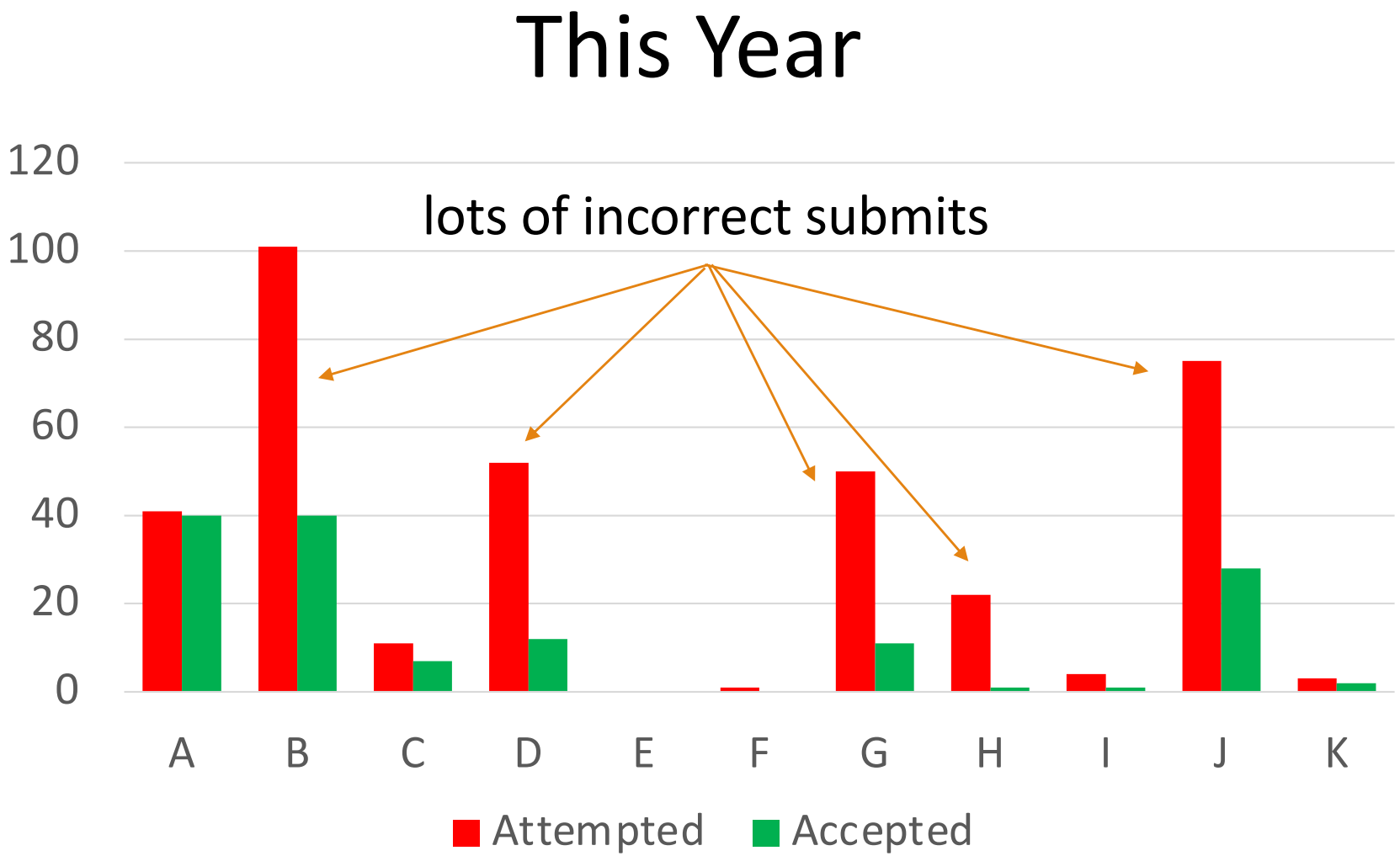


Problem vs. #Teams @Freeze

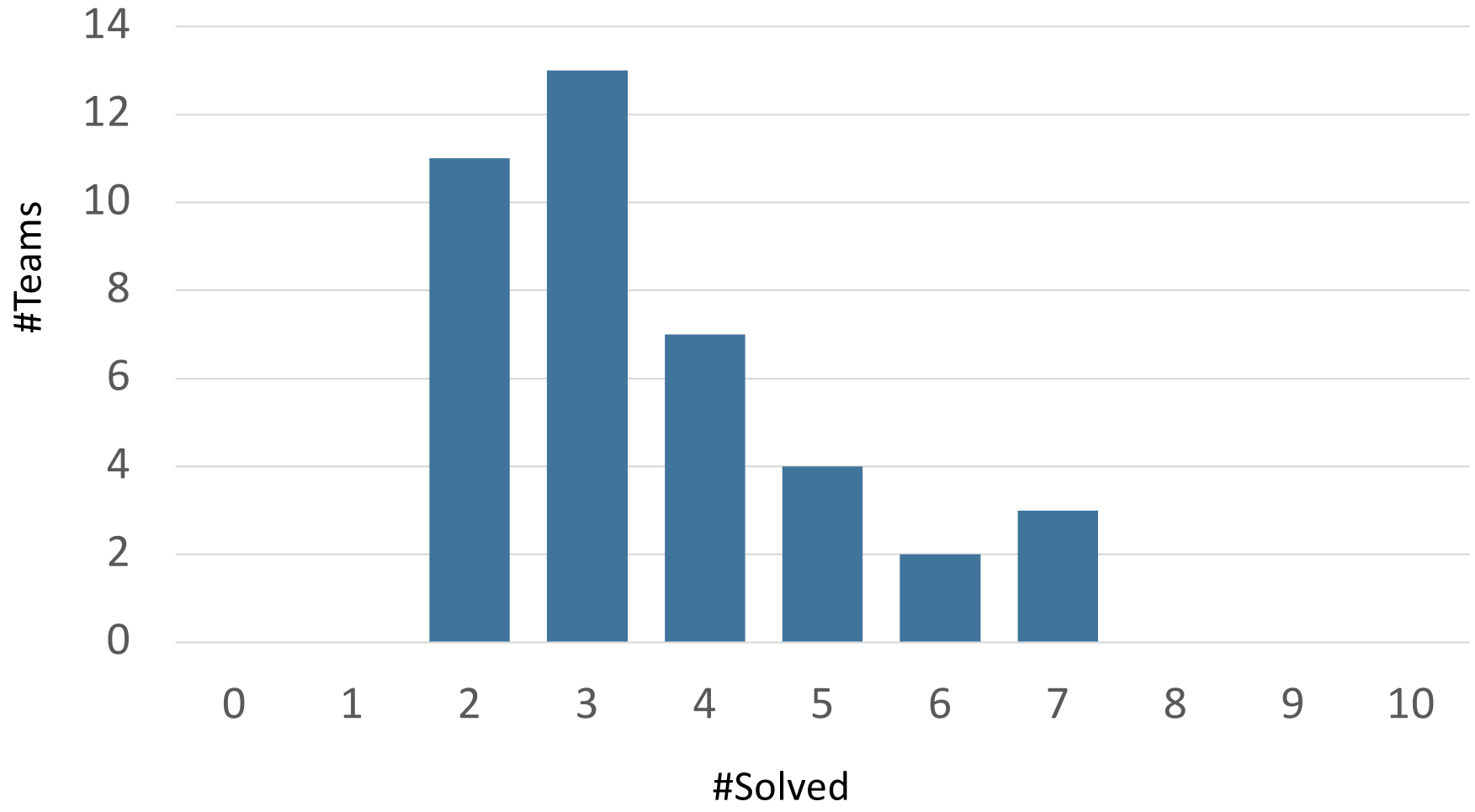
Last Year



Problem vs. #Teams @Freeze

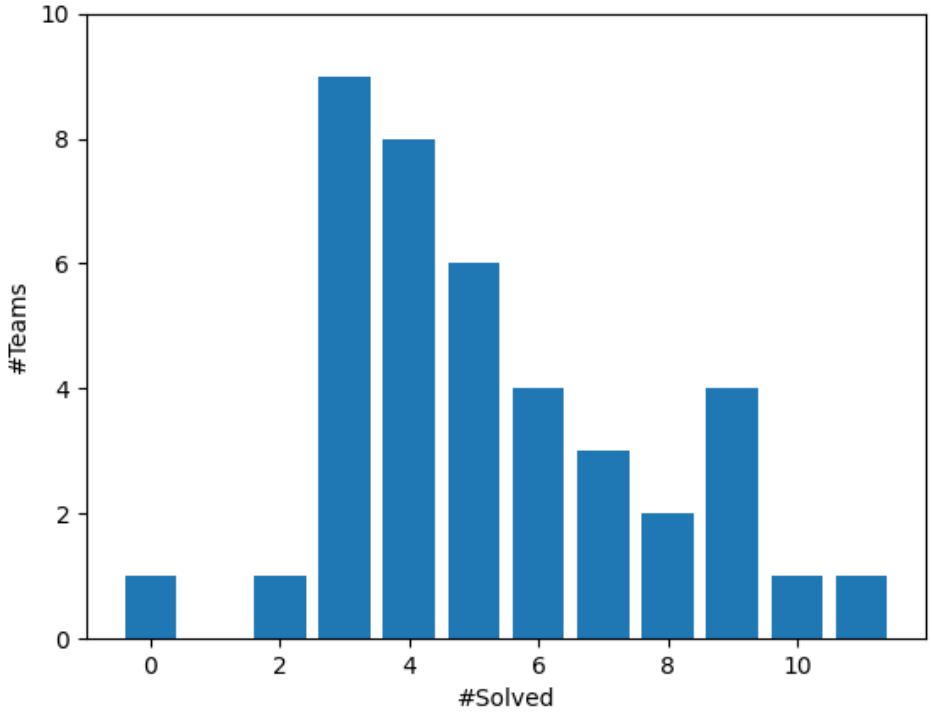


#Solved vs #Teams

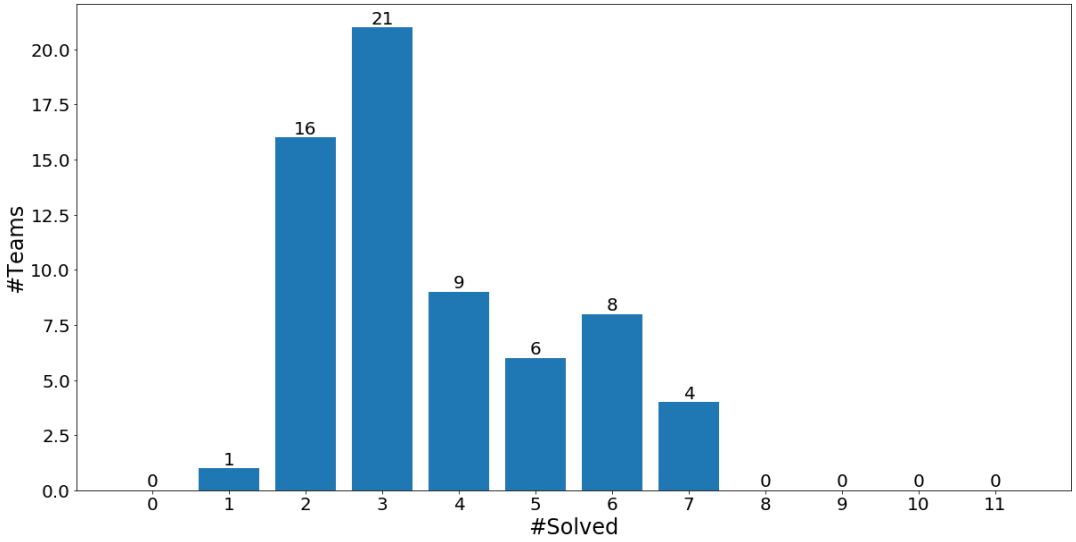


#Solved vs #Teams @Freeze

previous years

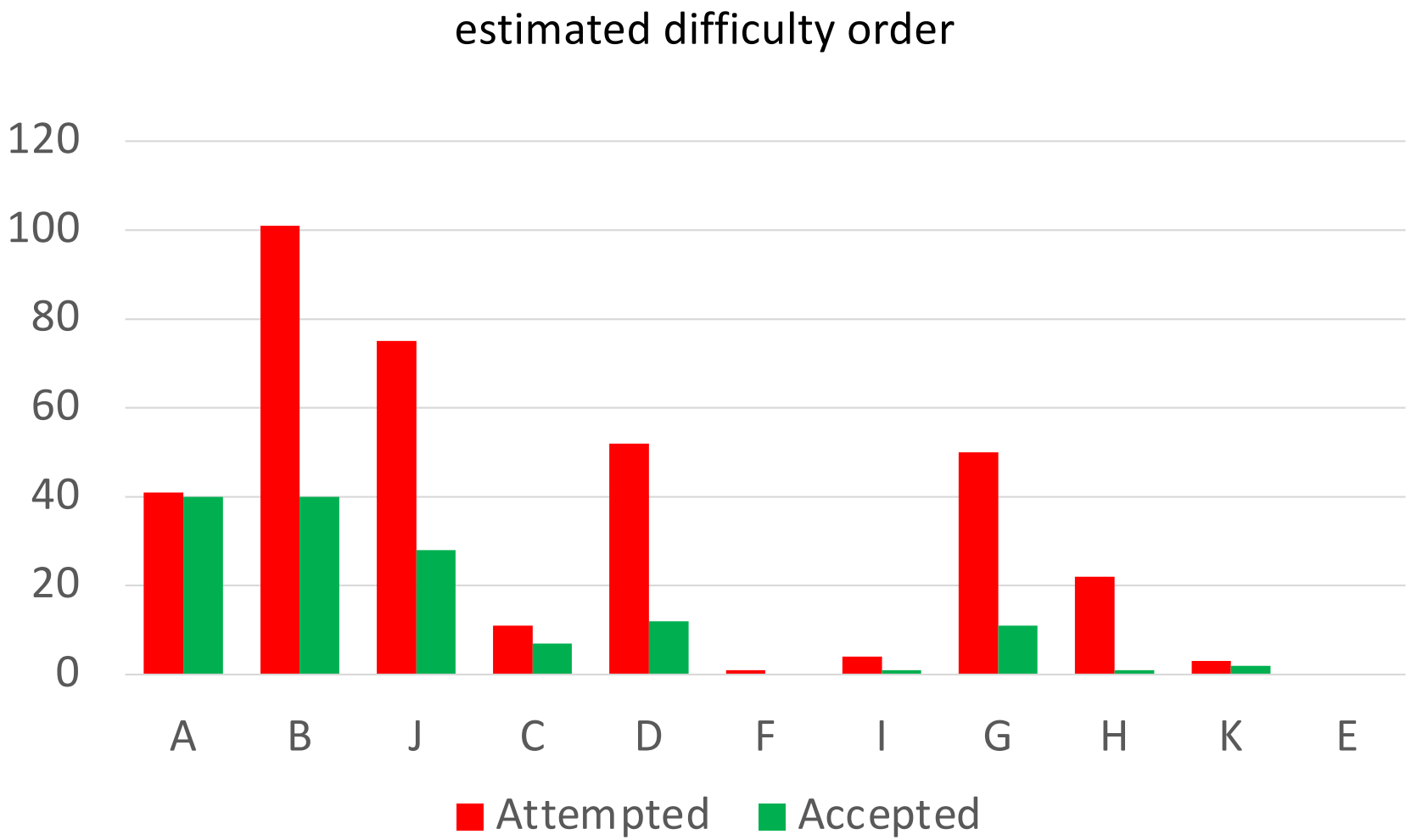


2020



2019

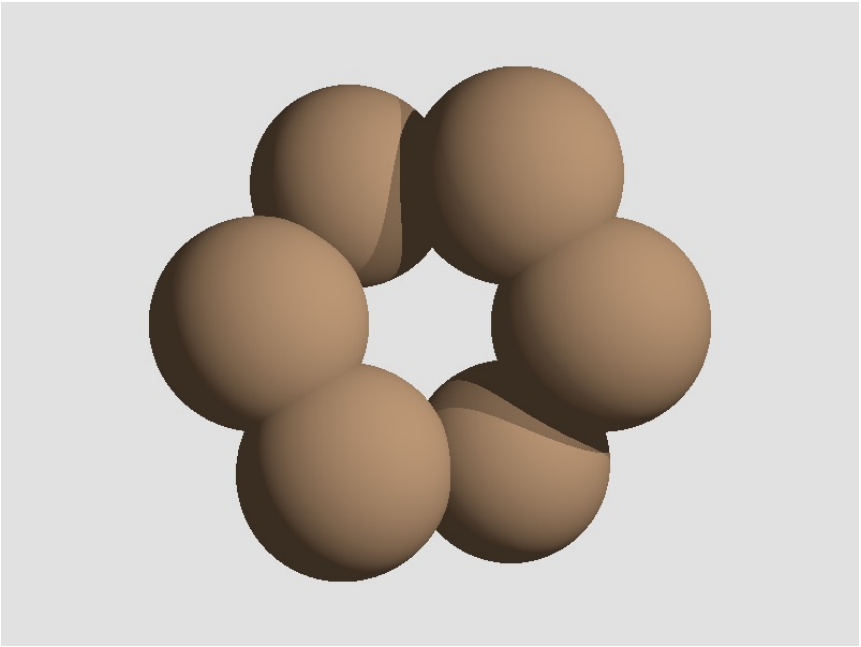
Problem vs. #Teams @Freeze

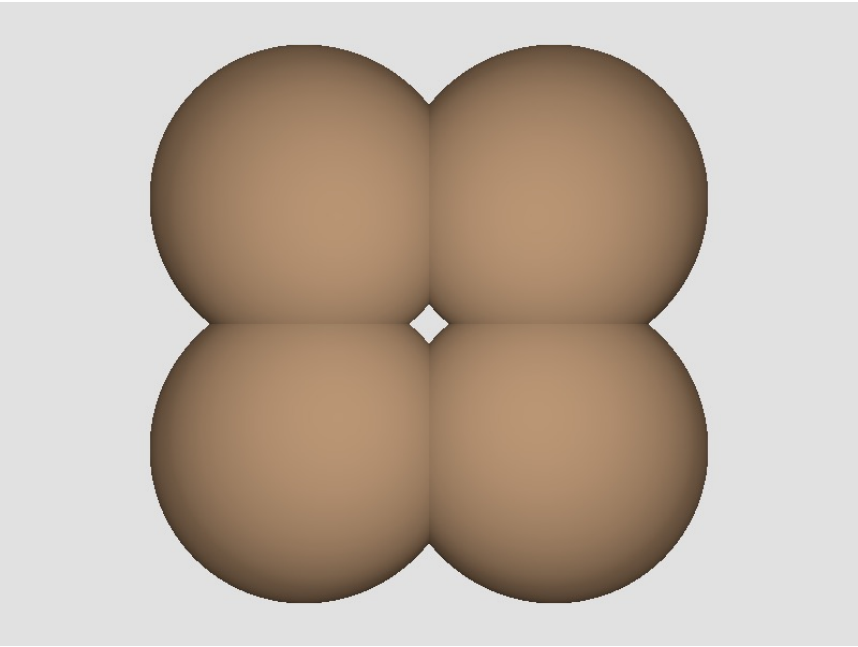


A: Loop of Chocolate

Story

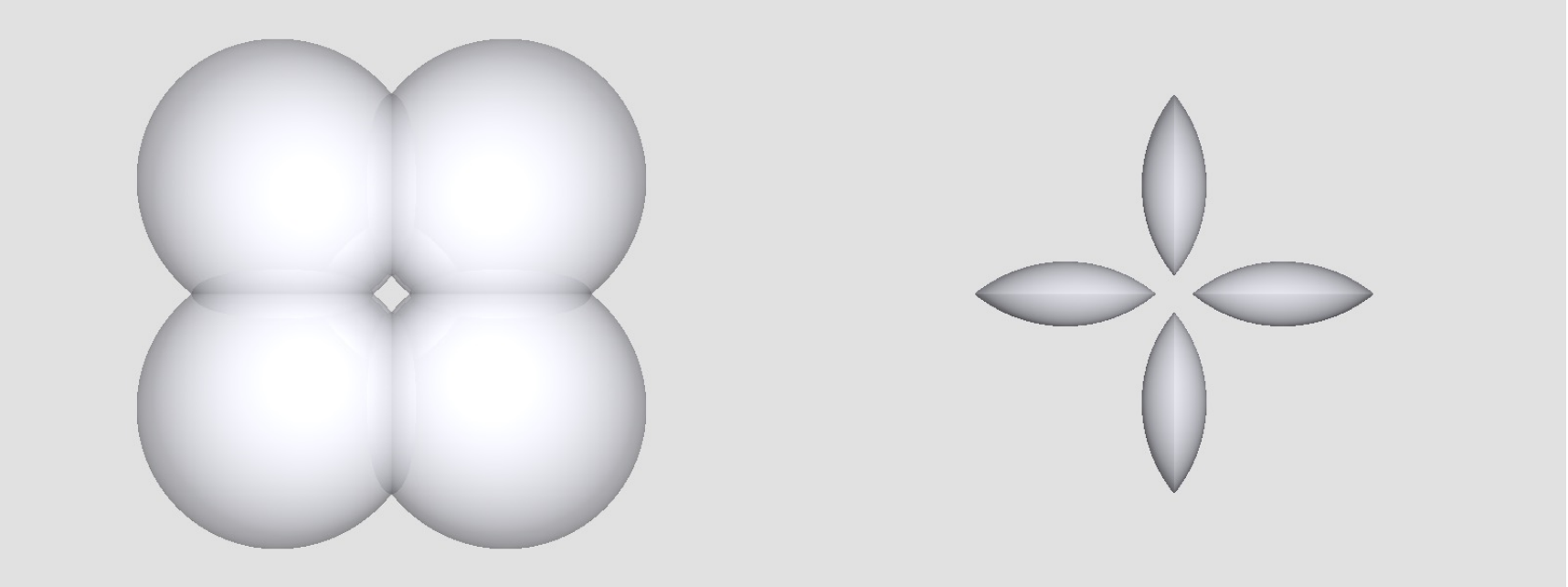
Let's make sweets of a fancy shape that is a loop of chocolate.





Design

The shape of a loop is formed by a union of a number of spheres.



Input and Output

1. Input: Given the radius and positions of spheres.
2. Output: The total volume of the union of the spheres, i.e., the amount of chocolate for filling the loop.

This problem was solved by all of the teams.

Solution

There are n spheres.

$S_k \equiv k$ th Sphere with the center (x_k, y_k, z_k)

$(S_{n+1} \equiv S_1)$

Volume of the Union $=$

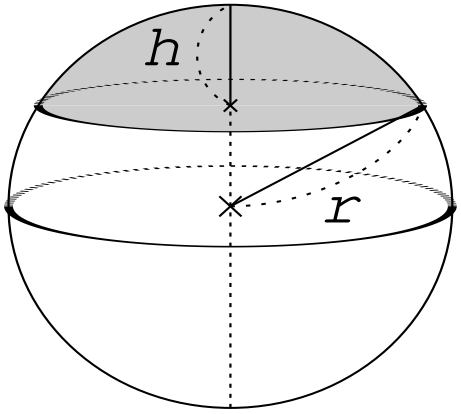
$$n \cdot \sum_{k=1}^{n} V(S_k) - \sum_{k=1}^{n} V(S_k \cap S_{k+1}),$$

$$V(S_k) \equiv \frac{4}{3}\pi r^3$$

$$V(S_k \cap S_{k+1}) \equiv \frac{2}{3}\pi \left(r - \frac{d_{k,k+1}}{2}\right)^2 \left(2r + \frac{d_{k,k+1}}{2}\right)$$

$$d_{k,k+1} = \text{distance}(\ (x_k, y_k, z_k), (x_{k+1}, y_{k+1}, z_{k+1})\ )$$

Spherical cap

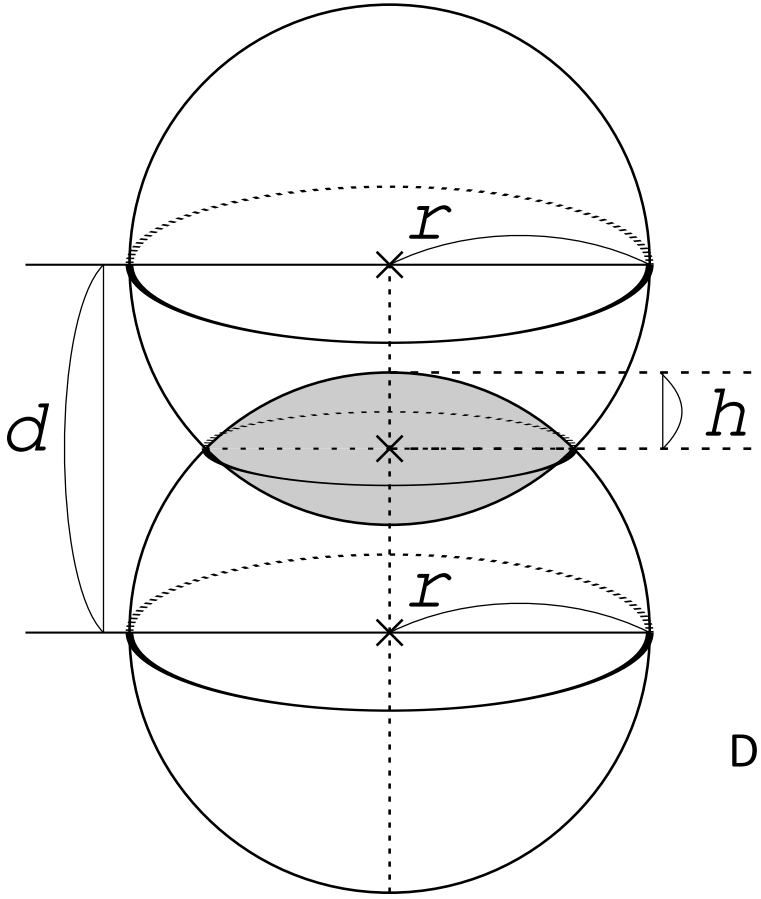


$$\int_0^h \pi(r^2 - (r-x)^2)dx = \pi h^2\left(r - \frac{h}{3}\right)$$

Volume of the spherical cap

Half of the volume of the intersection of two spheres with radius *r*

Intersection of two spheres



$$h = r - \frac{d}{2}$$

$$2\pi h^2\left(r - \frac{h}{3}\right) = \frac{2}{3}\pi\left(r - \frac{d}{2}\right)^2\left(2r + \frac{d}{2}\right)$$

Doubled

B:Lottery Fun Time

Problem

Calculate the maximum possible amount you may win by your tickets with different 6-digit numbers.

Winning matches and prizes are:

All 6 digits: 300,000 yen,

Lowest 4 digits: 4,000 yen, and

Lowest 2 digits: 500 yen.

- **Three winning numbers for the 3rd prizes.**
- **Winning numbers differ with the lowest 2 digits.**

Exhaustive Search!

1. **Find frequencies of last two digits and last four digits among the tickets at hand: $O(n)$**
2. **Find top 5 frequencies of last two-digits**
3. **Choose up to 3 out of the top 5 two-digits for the third prize.**
4. **Choose the most frequent four digit number with last two digits different from them, if any, for the second prize.**
5. **Any cards remain with last two digits unused, make one of them win the first prize.**

Pitfalls

All teams solved this problem but after many failures.

- **Top 5 rather than top 3 for candidates of third prizes. Two of them might better be used for the first and the second prizes.**
- **The variety of the last two digits in the tickets at hand can be quite poor.**
  - **The tickets may have only few kinds of last two digits.**
  - **Choosing less than 3 third prize numbers may be better, to allow winning the first and/or the second prizes.**
  - **Choosing one from the first and the second priezes may be required.**

C: Reversible Compression

Story: Data Compression

You are given the decoder of the data compression.

The set of code words is {**00**, **01**, …, **99**}

0X –> output **X**

X0 –> no output

AL –> do "output the A-th last one" for L times

Example: (Compression ratio = 8/15)

Code string: **00 01 25 48**

Decoded string: **0 1 01010 10101010**

Story: Data Compression

Some code strings decoded into the same string.

E.g., these are all decoded into **`0101010101010`**:

`00012548`, **`000122288210000`**, **`000128822210000`**

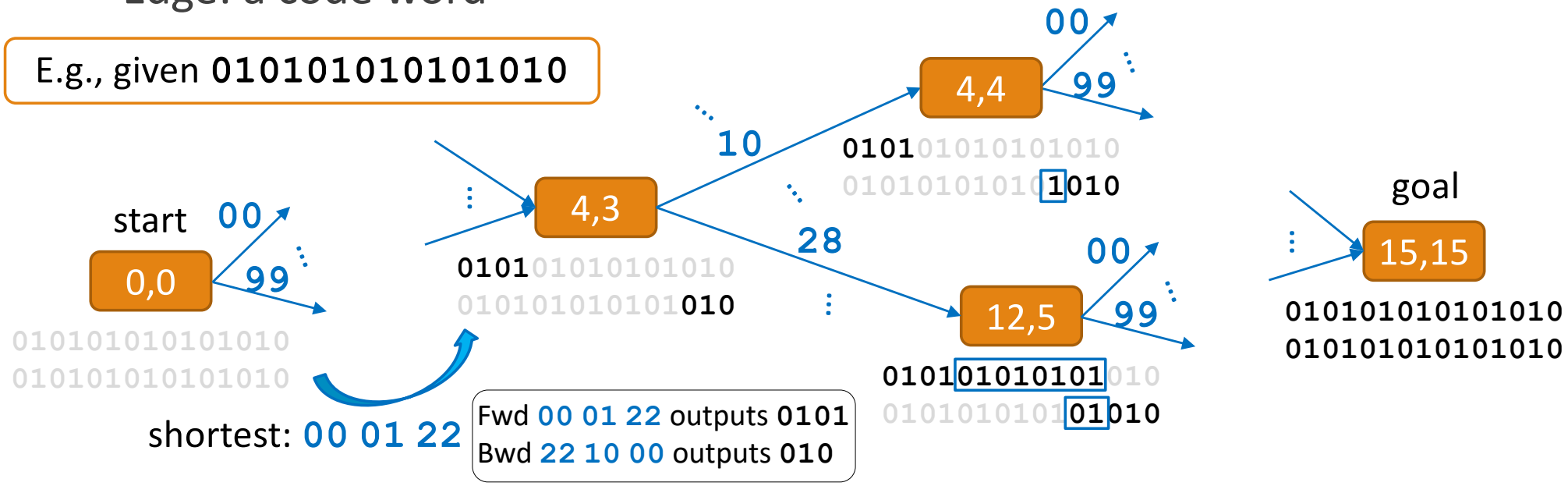
Reversible: it and its reverse are both decoded into the same string.

Your task is to find the lexicographically earliest shortest *reversible* code string decoded into the given string.

Solution

Shortest path search in a DAG of $(n+1)^2$ vertices :
Vertex: a pair (i, j) meaning that the first i digits has been output and the last j digits are to be output

Edge: a code word



You can use BFS, Dijkstra search, or a simple DP filling up the table of size $(n+1)^2$ to solve this problem in $O(n^2)$ or $O(n^2 \log n)$

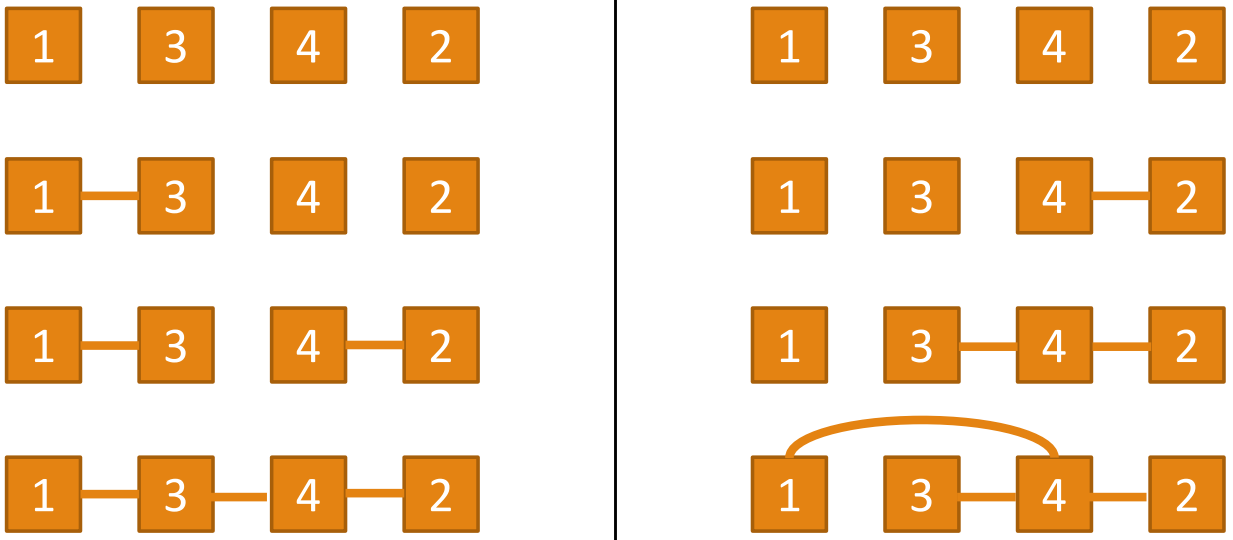
D:Wireless Communication Network

Problem

We are building a tree by the following procedure

1. Start from n trees of singletons
2. Select two adjacent trees and join their highest-numbered nodes

Compute the largest diameter among the possible trees



Solution: DP

Let

diam(*l*,*r*) = the largest diameter for the interval {*n*[*l*], ..., *n*[*r*-1]},
height(l,r) = the height of such tree.

Then, we have

diam(*l*,*r*) = max {diam(*l*,*m*), diam(*m*+1,*r*), height(*l*,*m*)+height(*m+1*,*r*)+1} + 1
where *m* is the index of the node with the highest value.

Because:

- the diameter is spanned by a path not passing *m* or passing *m*, and
- not passing m ⇔ the path is contained in [l..m) or [m+1..r) because of the construction of the tree.

E:Planning Railroad Discontinuation

Problem Description

Compute the cost of a minimum spanning tree (MST) of a given graph.

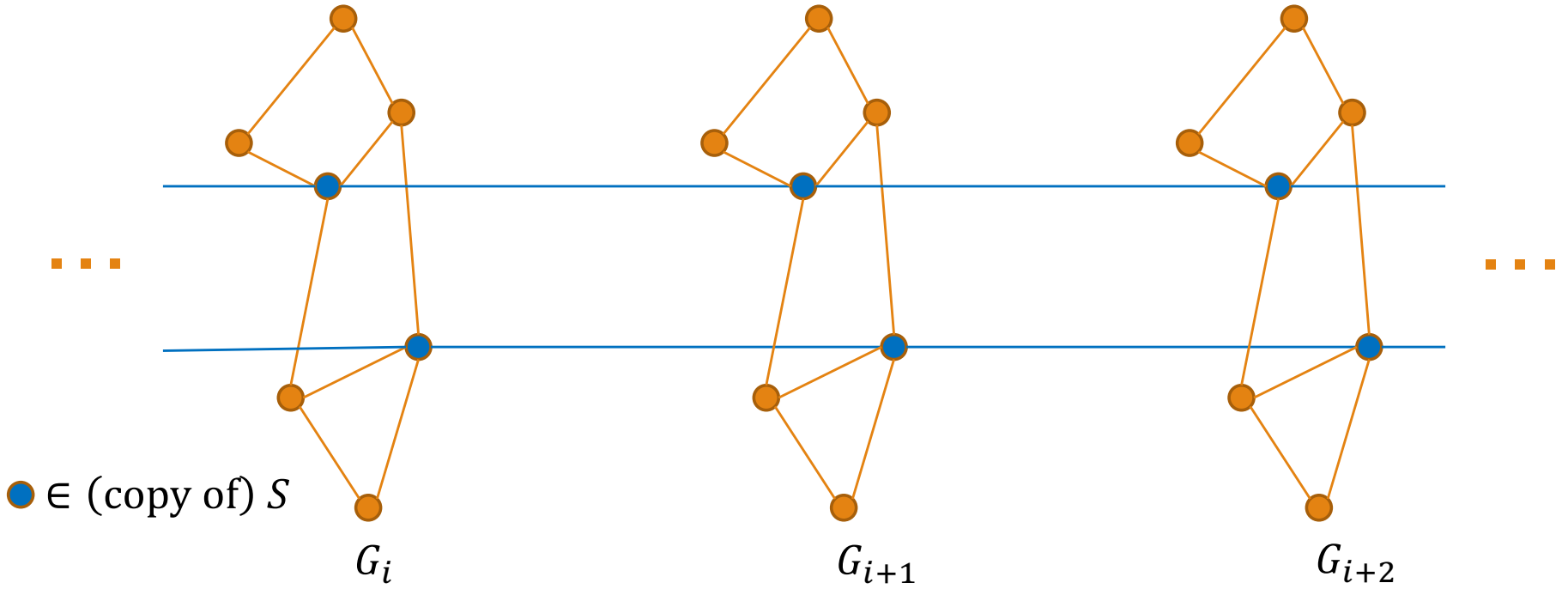
The given graph has a huge number of vertices, but has good symmetry.

Graph

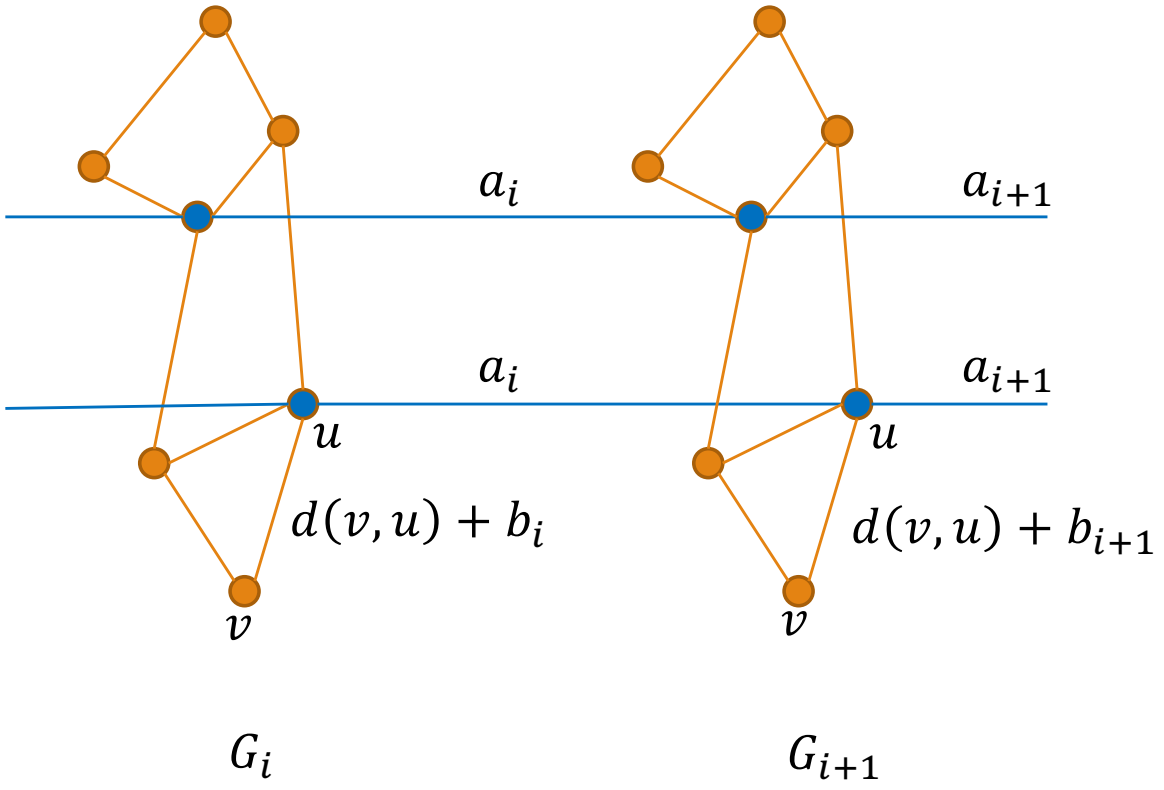
$G = (V, E)$: an (undirected weighted) graph (subway network in a city)

$S = \{s_0, s_1, \dots, s_{r-1}\} \subset V$: subset of V (bullet train stations)

$G_0, G_1, \dots, G_{l-1}$: l copies of G



Cost



Solution

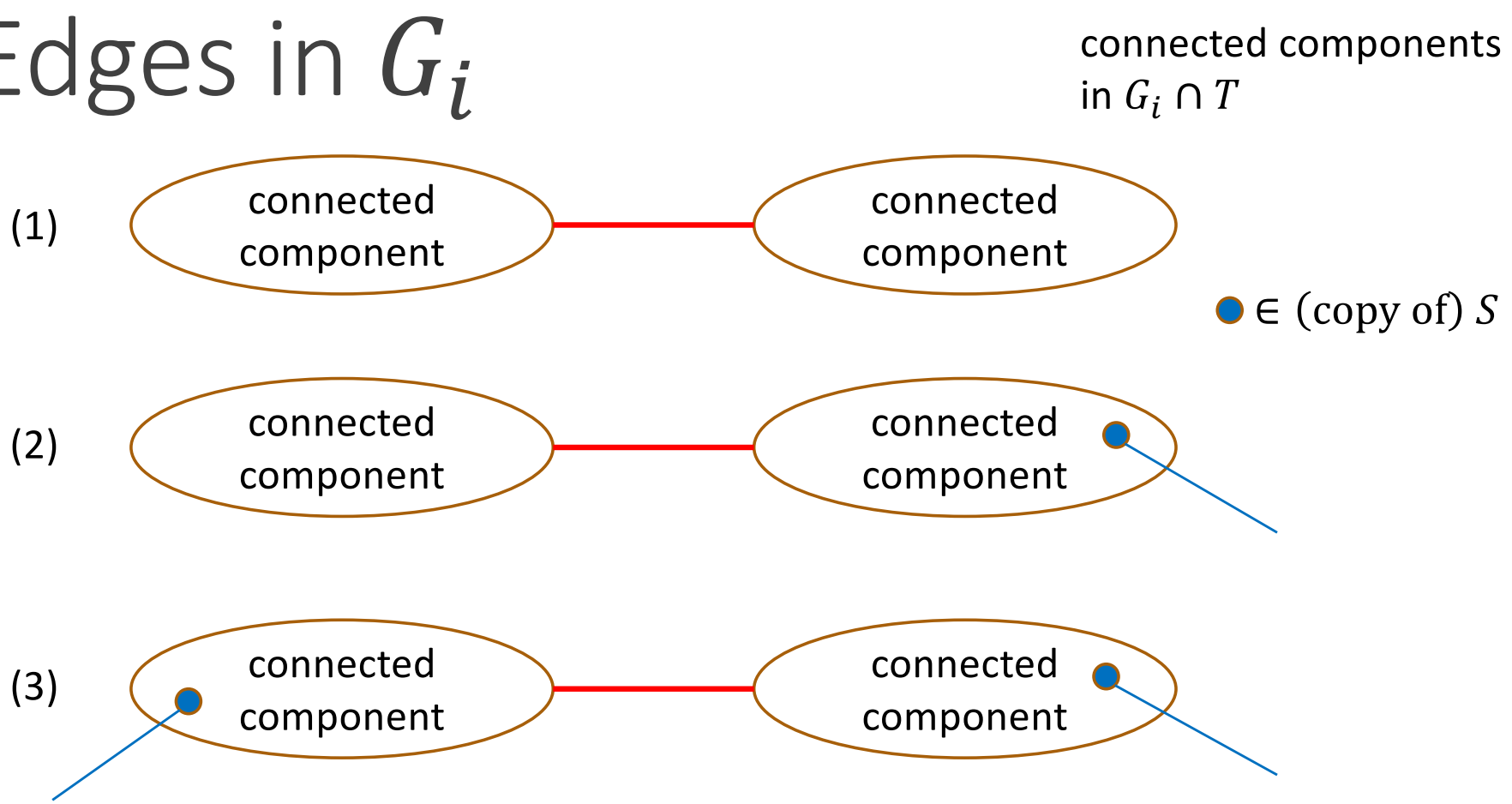
Kruskal's algorithm

- $G^{all} = (V^{all}, E^{all})$: whole graph $\supset G_0, G_1, \ldots$

- Sort edges by their cost.
- $T = (V^{all}, \emptyset)$: forest in G^{all} with no edges
- For each edge, determine whether the edge can be added to the forest T using the union-find data structure.

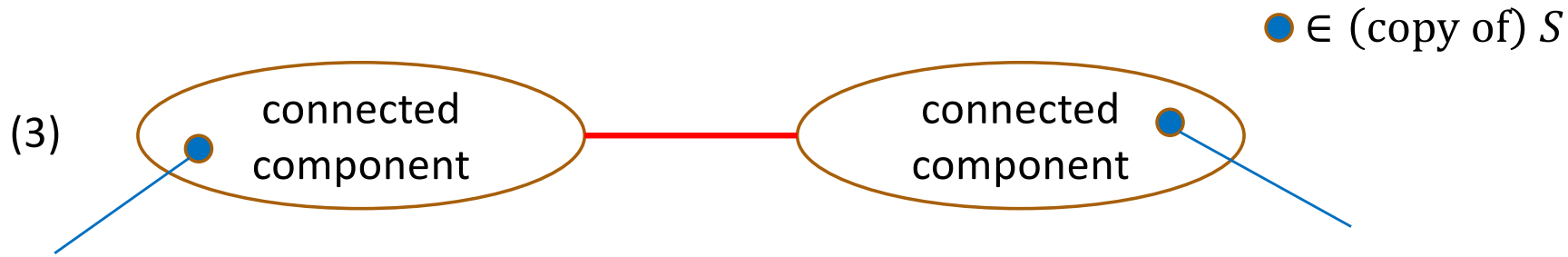
This is too slow and uses too much memory

Edges in G_i



In the cases (1) and (2),
whether the red edge can be added to T is determined only in G_i.
This is the same for all subgraph G_i.

Preparation



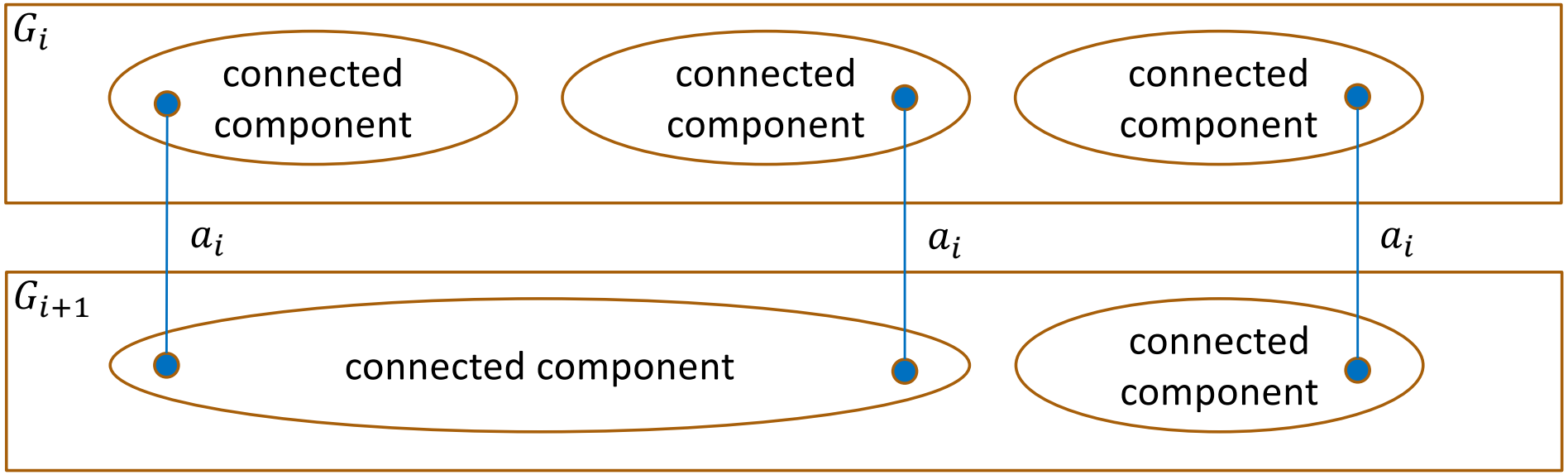
- Apply Kruskal's algorithm to G and make a table

conn[c] := #(connected components of $G(c)$ containing some $v \in S$)
cost[c] := ∑cost of edges of type (3) with cost $\leq c$

$G(c)$: subgraph with edges whose costs $\leq c$

Edges between G_i and G_{i+1}

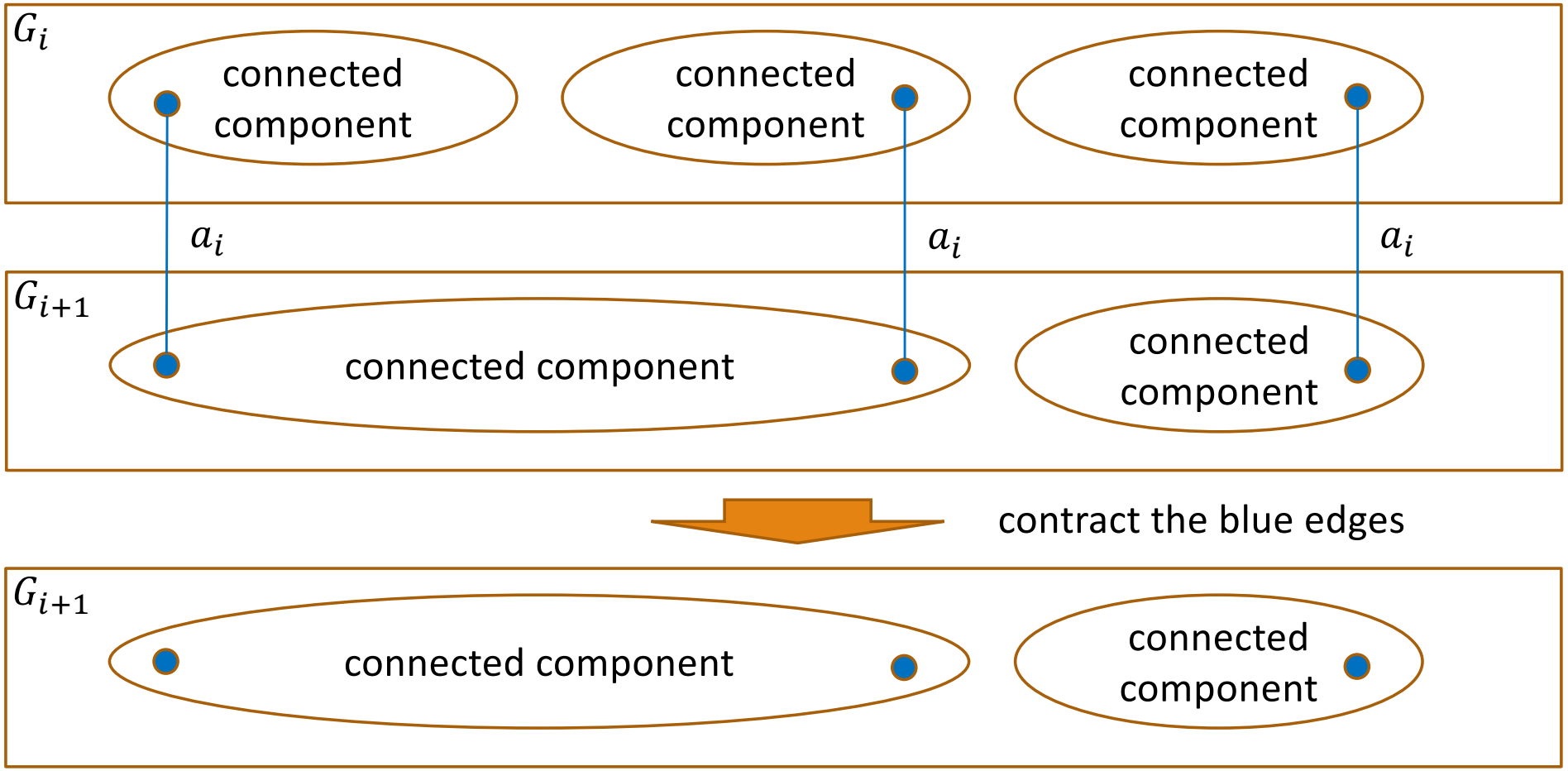
if $b_i > b_{i+1}$



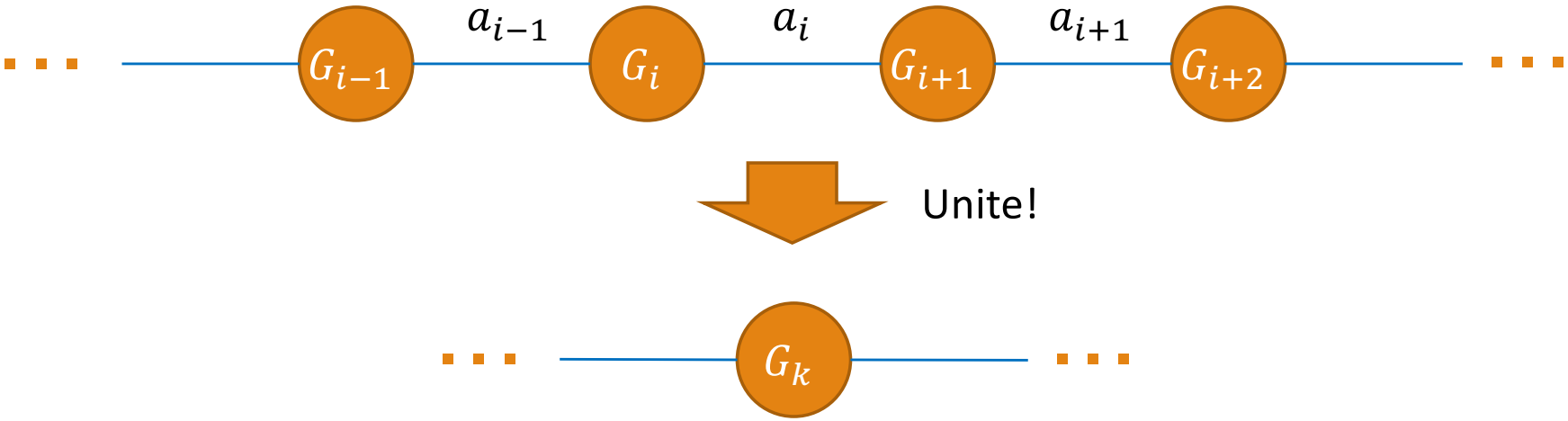
#(edges with cost a_i in an MST) = #(connected components of $G_i \cap T$)
$= \text{conn}[a_i - b_i - 1]$
if $b_i > b_{i+1}$

Edges between G_i and G_{i+1}

if $b_i > b_{i+1}$



If the edges with costs a_{i-1}, a_i, a_{i+1} are contracted, …



$$b_k = \min\{b_{i-1}, b_i, b_{i+1}, b_{i+2}\}$$

F:It’s Surely Complex

Problem Description

Given a prime number p and a positive integer n, compute the product of all the complex numbers $a+bi$ such that

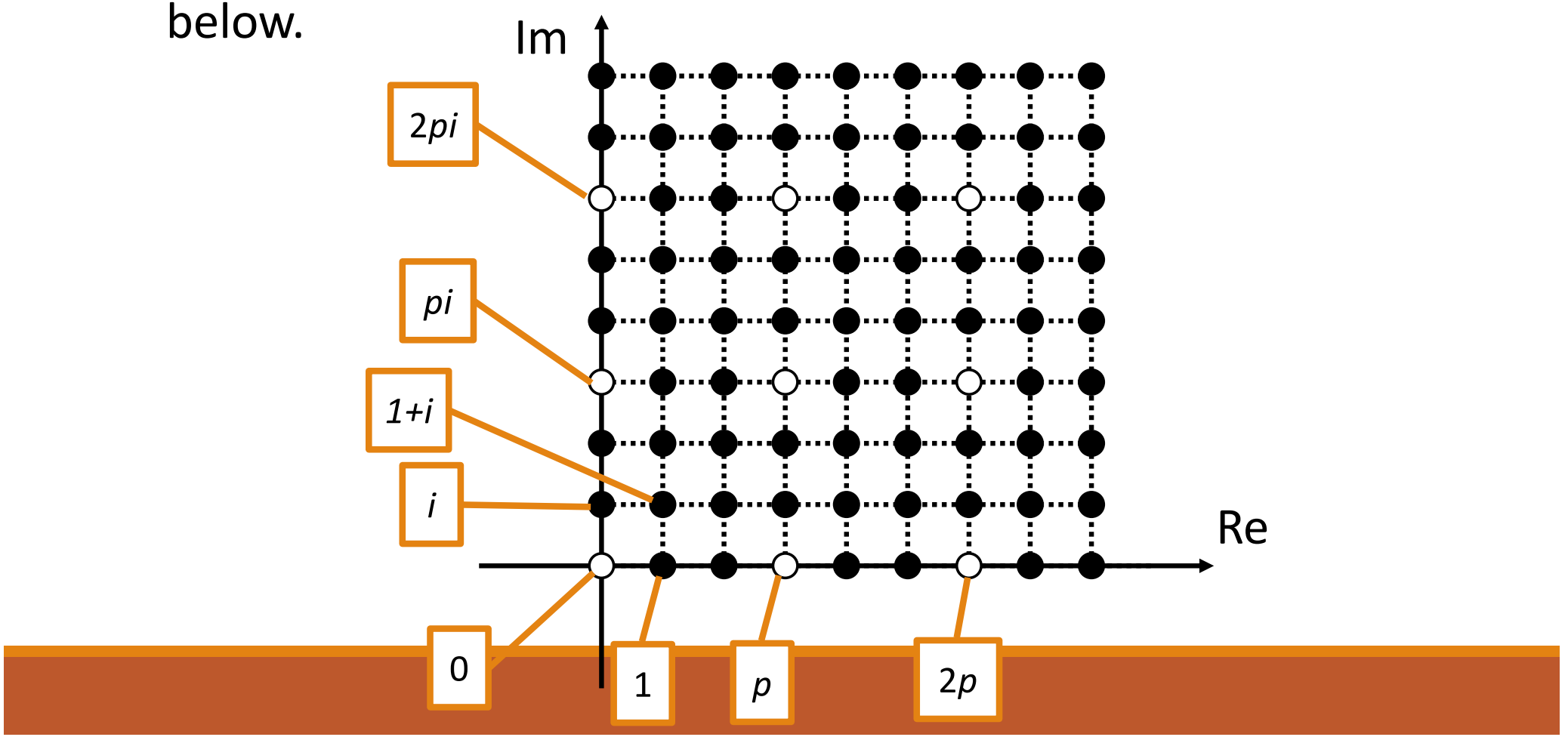
- a and b are integers between 0 and n, inclusive, and
- a and/or b is not a multiple of p.

The number of the complex numbers can be up to almost 10^{36}, and the output should be in modulo p.

In the following, "mod p" may be omitted.

Problem Description

More visually, compute the product of all the numbers represented as black dots on the complex plane in the figure below.

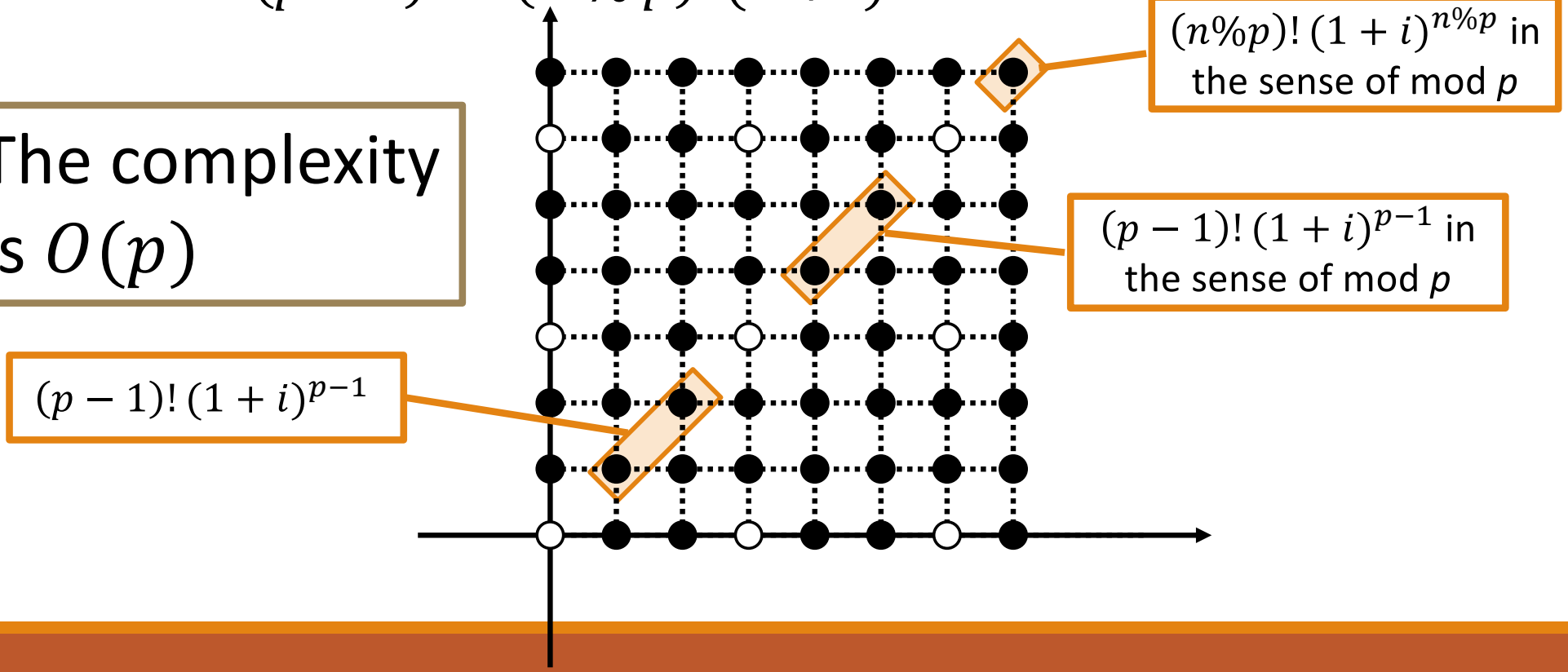


Solution

Since $(1+i)\cdots(k+ki) = k!\,(1+i)^k$, the product of the numbers on the diagonal is

$$(p-1)!^{\left[\frac{n}{p}\right]}\,(n\,\%\,p)!\,(1+i)^{(p-1)\left[\frac{n}{p}\right]+(n\%p)}$$

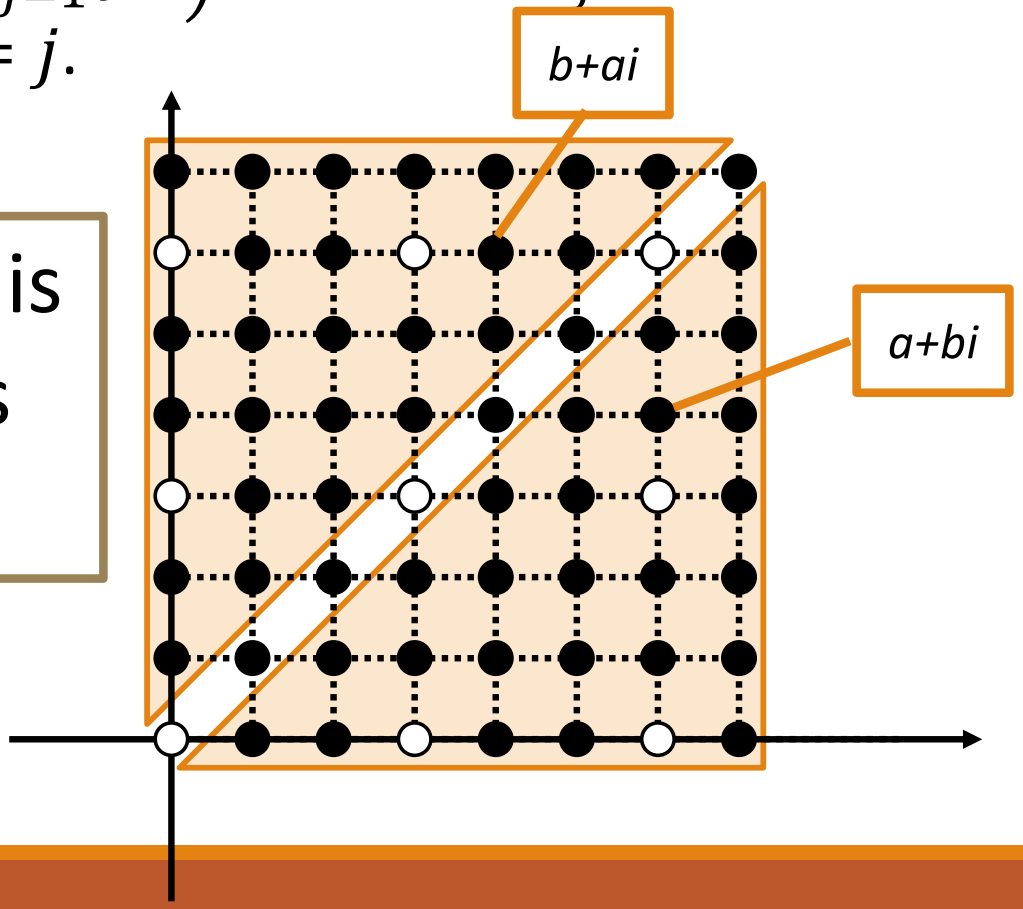
The complexity is $O(p)$



Solution

Because $(a+bi)\times(b+ai) = (a^2+b^2)i$, the product in the triangle areas is $\left(\prod_{j=1}^{p-1} j^{c_j}\right) i^{\sum c_j}$ where c_j is half the number of $\langle a, b\rangle$ s.t. $a^2+b^2 = j$.

The complexity is $O(p \log n)$ if c_js are available



Solution

The complexity is $O(p \log p)$ with Fast Fourier Transform (FFT)

The number of $\langle a, b \rangle$ s.t. $a^2 + b^2 = j$ $(0 \leq a, b, j < p)$ is calculated by performing self-convolution of $(d_0, \ldots, d_{p-1}, 0, \ldots, 0)$ where d_k is the number of as s.t. $a^2 = k$ $(0 \leq j < p)$.

$$(d_0, \ldots, d_{p-1}, 0, \ldots, 0)$$

Self-convolution

This element is not used

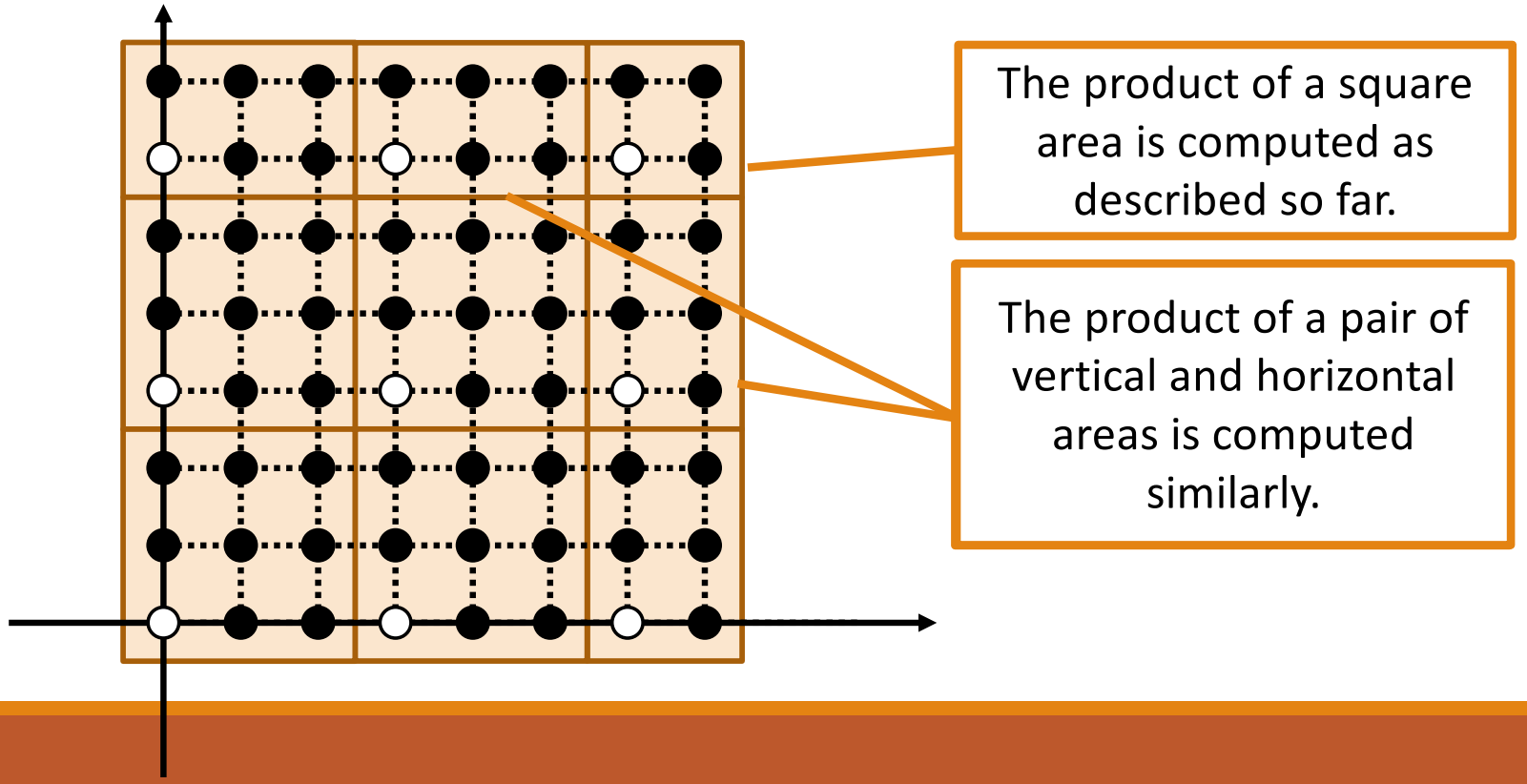
$$(d_0 d_0, \ldots, \textstyle\sum_{k_1+k_2=k} d_{k_1} d_{k_2}, \ldots, d_{p-1} d_{p-1})$$

These numbers include the cases where a = b, and so need some adjustments

Solution

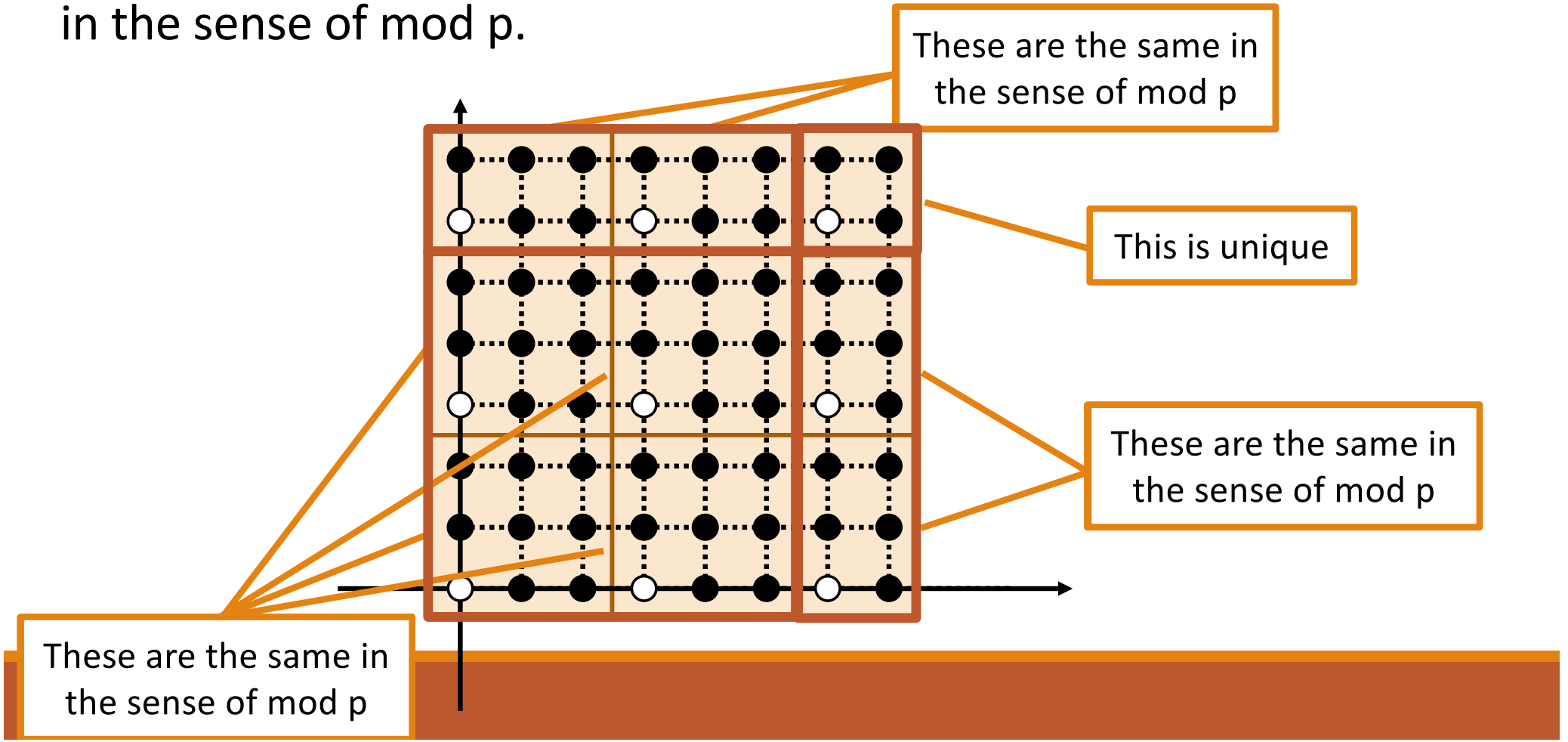
FFT Convolution is efficient, but FFT over the complex field with double precision FP numbers doesn't have enough precision.

The precision problem is solved by dividing the area like the following.



Solution

Although the number of square and rectangle areas can be very large, at most four of them are unique and the others are replicas in the sense of mod p.



The Last Comment

A straightforward FFT program in Java/Kotlin may create and destroy many objects, and so it can be rather slow. To make matters worse, the Java execution environment of today's contest uses Serial GC.

G:Genealogy of Puppets

Problem Description

Generate a binary tree under the following constraints:

- The tree consists of n (≤ 300) vertices.
- Vertices are numbered from 1 to n.
- Vertex i has at least x_i and at most y_i children.
- If vertex i has any child, there must be a child whose index is larger than i.

How many trees (modulo $10^9 + 7$) can be generated?

Examples

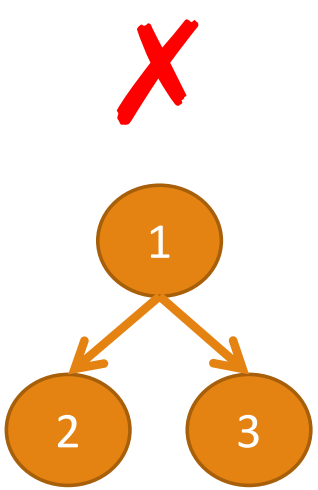
Sample Input 1

$n = 3$

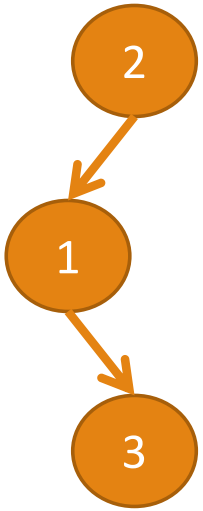
$(x_1, y_1) = (0,1)$

$(x_2, y_2) = (1,2)$

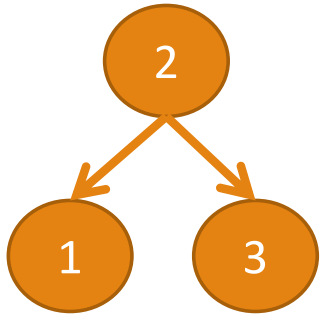
$(x_3, y_3) = (0,2)$



Vertex 1 cannot have 2 children.



At least 1 index of children must be larger index than their parent's index.



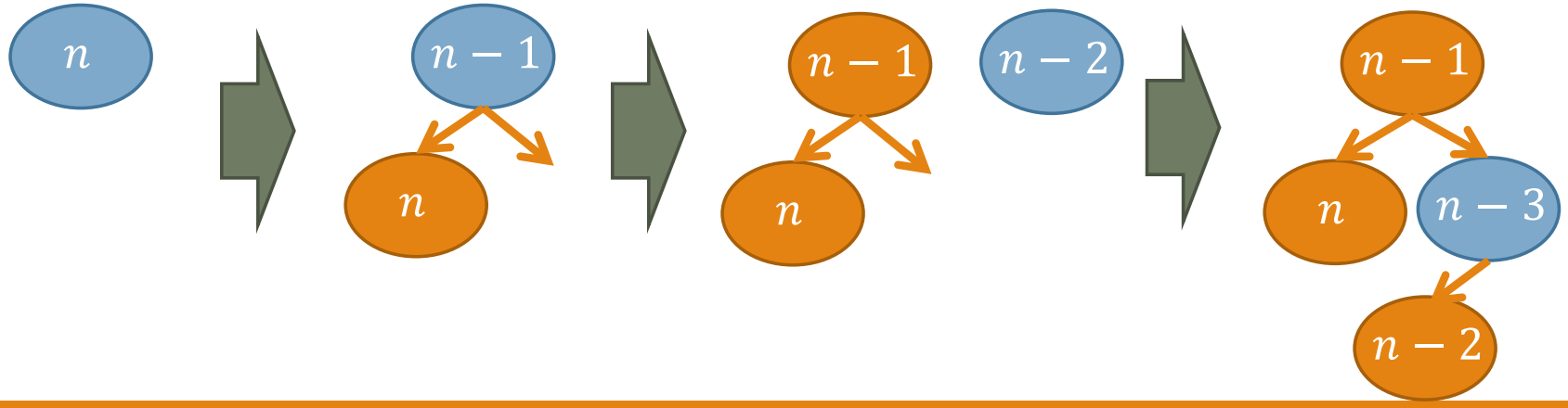
The tree satisfies all constraints.

Solution

Basic strategy: Build subtrees by adding vertices from vertex n to vertex 1.

Dynamic Programming memorizing

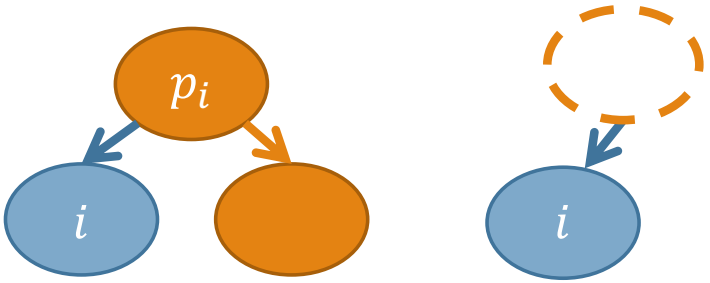
- the number of roots, and
- the number of unassigned edges (connected to smaller-index vertices).



How to add a new vertex?

To add a new vertex i, you need to consider the indices of the parent p_i and the children $c_{i,left}$, $c_{i,right}$.

Parent edge

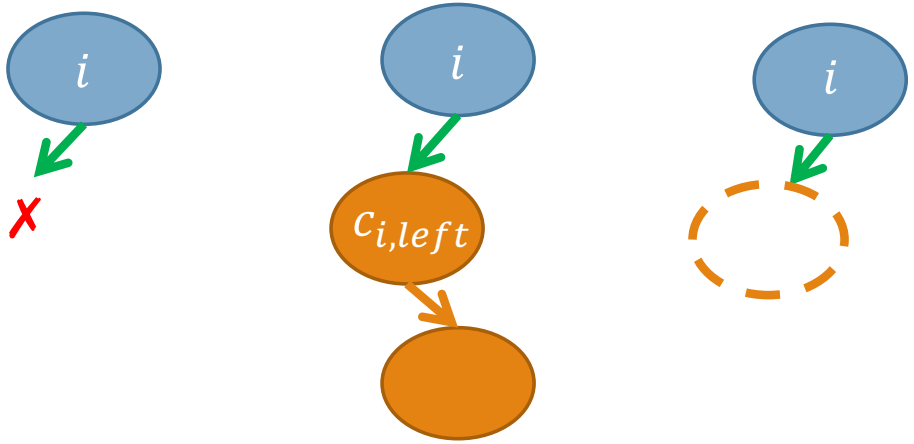


$p_i > i$

$p_i < i$

(decide p_i later)

Left child edge*



No child

$c_{i,left} > i$

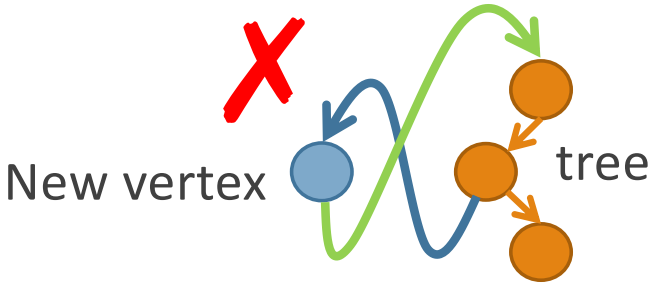
$c_{i,left} < i$

(decide $c_{i,left}$ later)

*) $c_{i,right}$ can be determined in the similar way.

How to add a new vertex?

How to deal with the loop case?



No tree can be used as both parent and child at the same time.

Key insight: whatever the parent is, there is **exactly one** root node cannot be used as the child.

→ #(child tree candidates) is always #(all trees) – 1.

The overall time complexity is O(n^3).

H: Cancer DNA

Problem Description

You are given m DNA patterns $P_1, \ldots, P_m$ of length n.

The task is to approximate

$$\Pr_{S \sim \{A,G,C,T\}^n}[S \text{ matches } P_i \ (\exists i)]$$

up to a 5% multiplicative error.

Example: $P_1 =$ "AC?“ , $P_2 =$ "A?T“

ACA, ACG, ACC, ACT, AAT, AGT, ATT matches P_1 or P_2.

$$\Pr_S[S \text{ matches } P_1 \text{ or } P_2] = \frac{7}{4^3} = 0.109375.$$

A First Attempt

- Random sampling

  Randomly choose k samples $S_1, \ldots, S_k \sim \{A, G, C, T\}^n$, and approximate the probability by

  $$\frac{\#\{S_j \mid S_j \text{ matches } P_1, \ldots, P_m\}}{k}.$$

- Issue

  However, if $\Pr_S[S \text{ matches } P_1, \ldots, P_m] \approx 0$, the numerator is 0 with high probability, in which case the output is not a good approximation in terms of *multiplicative* error.

  (This is good approximation in terms of an *additive* error, though.)

Solution

- Idea: Random sampling from a smaller space!
- Let $U \coloneqq \{(i, S) \mid S \text{ matches } P_i\}$. (i.e., the disjoint union of DNA sequences that match P_i)
- Let $G \coloneqq \left\{(i, S) \in U \,\middle|\, i = \min\{j | S \text{ matches } P_j\}\right\}$.

  (If S matches P_i but not $P_1, \dots, P_{i-1}$, then $(i, S) \in G$.)
- Our goal is to compute $4^{-n} \cdot |G| = 4^{-n} \cdot |U| \cdot \frac{|G|}{|U|}$.
- $|U|$ can be easily computed by counting the number of ?'s.
- $|G|/|U|$ can be approximated by random sampling.

  Uniformly sample $(i, S) \sim U$ and check if $(i, S) \in G$. (in time $O(nm)$)

  Since $\frac{|G|}{|U|} \geq \frac{1}{m}$, the random sampling can estimate $\frac{|G|}{|U|}$ up to a $1 \pm \epsilon$ error in time $O\left(\frac{m}{\epsilon^2}\right)$.
- Overall, the probability can be $(1 \pm \epsilon)$-approximated in time $O(nm \cdot m/\epsilon^2)$ with high probability.

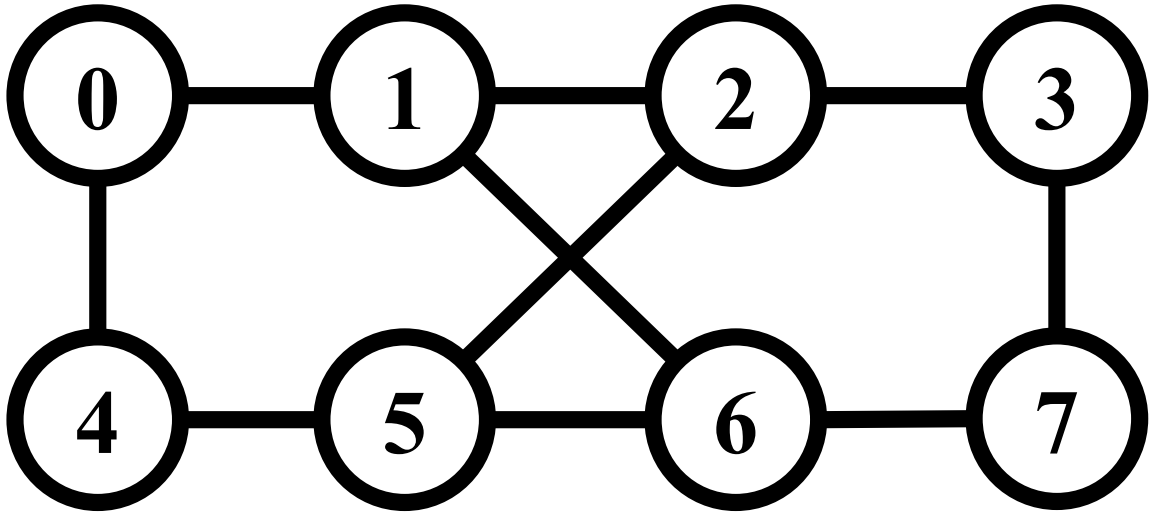
Background

- A randomized algorithm that computes a $(1 \pm \epsilon)$-multiplicative approximation of a solution in time $\text{poly}\left(n, \frac{1}{\epsilon}\right)$ is called *FPRAS* (fully polynomial-time randomized approximation scheme).
- The algorithm is a special case of FPRAS for DNF counting.

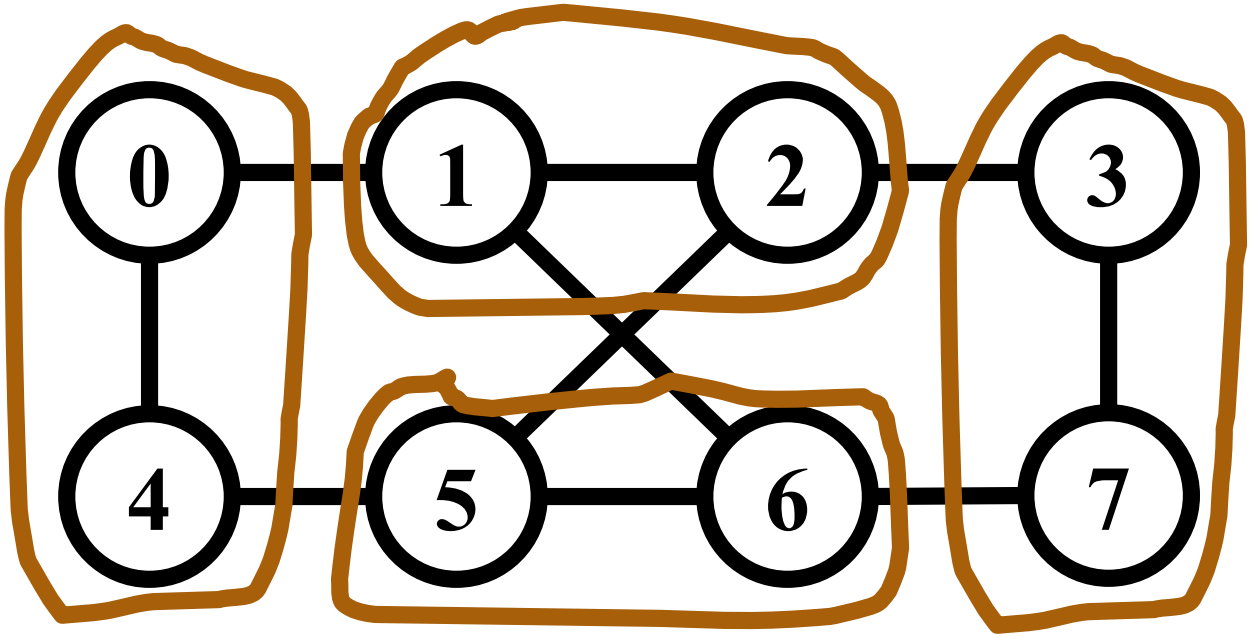
I: “Even” Division

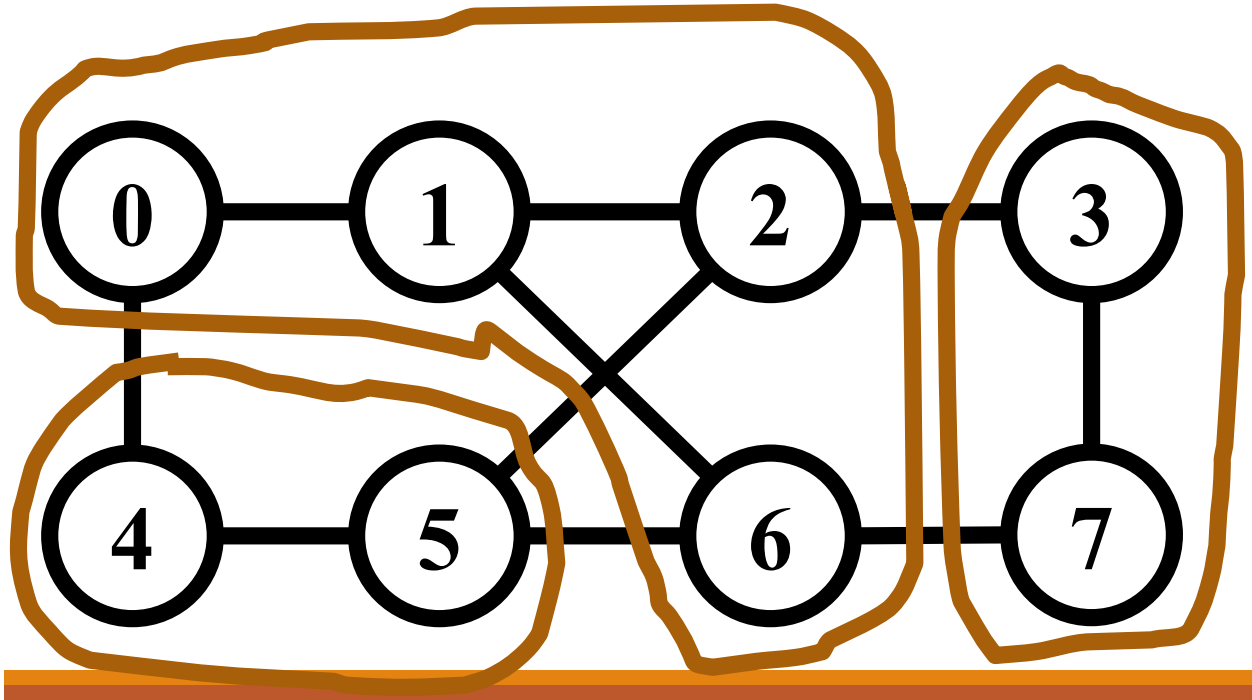
Problem Description

Divide a graph into connected subgraphs with **even** number of nodes
(#node, #edge ≦ 100,000)



Any such division is accepted, as long as it does not allow further division.





Approach

1. **Divide into two** connected even-nodes subgraphs
2. **Repeat** it until no further division is possible

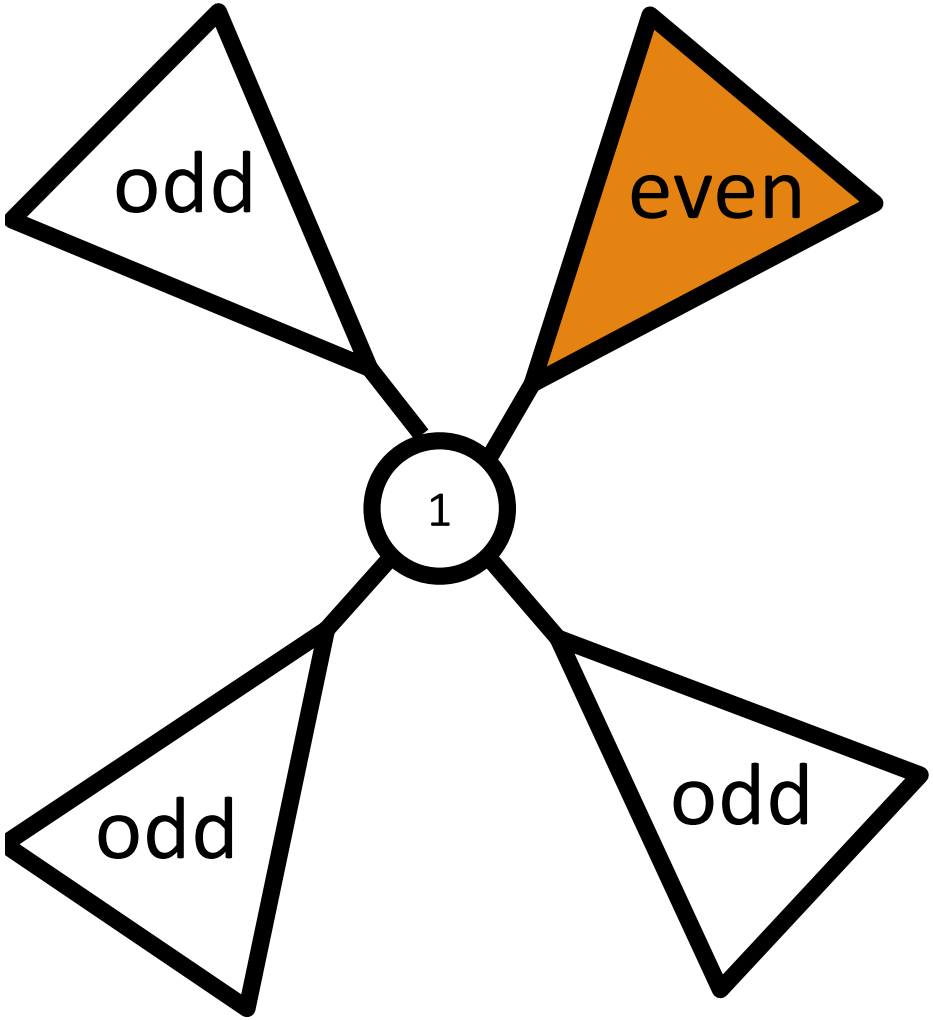
Divide to two: Trees

[Key insight]

Dividing to two parts is easy for **trees**.

Let’s think about the case of trees first.

Divide to two: Trees



A node with even degree

→

At least one subtree
has even nodes
(if not, 1 + odd*even = odd)

All nodes have odd degrees

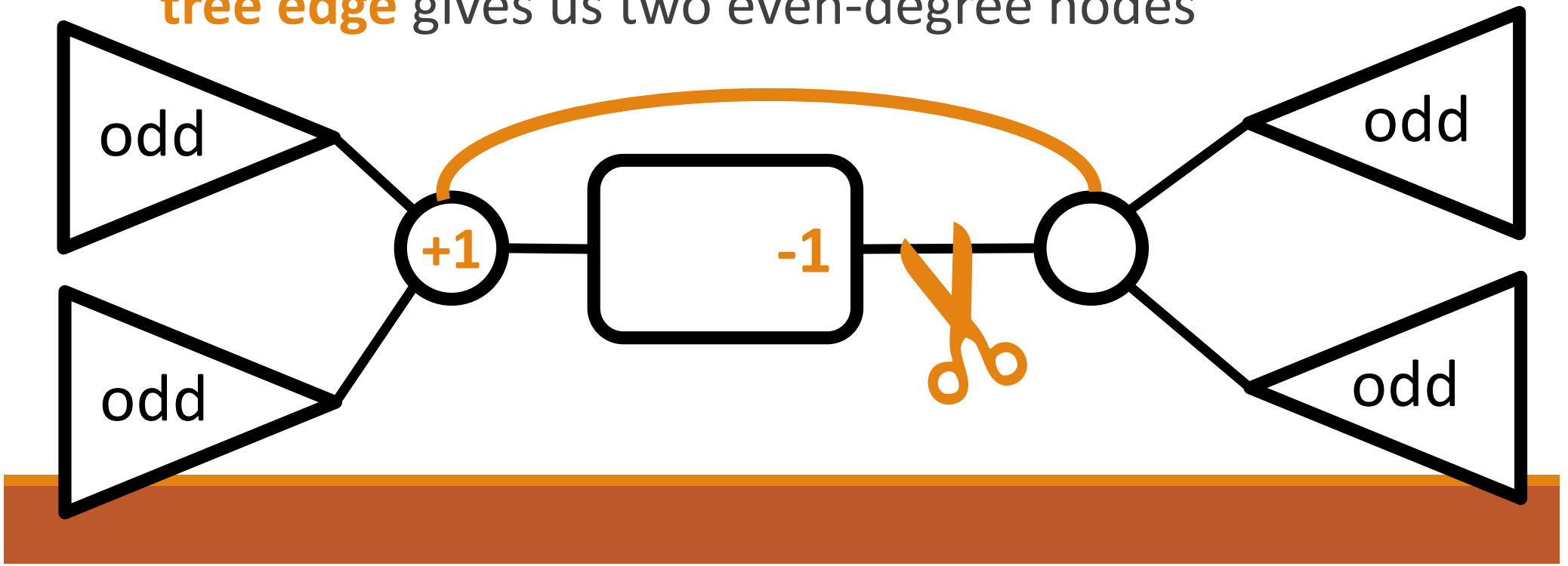
→

no division is possible
(proof by induction)

Divide to two: General case

Take a spanning tree.

- Even degree node → Divide at the node
- All degrees odd → Replacing an edge with **a non-tree edge** gives us two even-degree nodes



Repeat

- Naively repeating the process takes $\Omega(NM)$ time, but
- **You don't need to recompute the spanning tree every time**.
  - Take a spanning tree (DFS tree has a good structure) of the whole graph **only once**
  - Run a **one-pass bottom-up traversal** to find even-degree nodes and non-tree edges

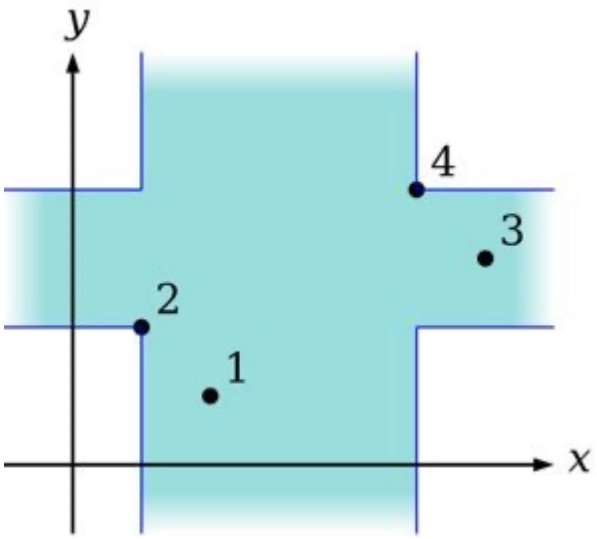
Background

- The division is a kind of generalization of the *perfect matching* (= all subgraphs have 2 nodes, instead of even nodes)
  - A. D. Scott. *On induced subgraphs with all degrees odd*. Graphs and Combinatorics, 17(3):539–553, 2001.

J:The Cross Covers Everything

Problem

1. Given n points
2. Select a pair of them
3. Create a cross shaped area
4. Count how many pairs that covers all given points



Amendment

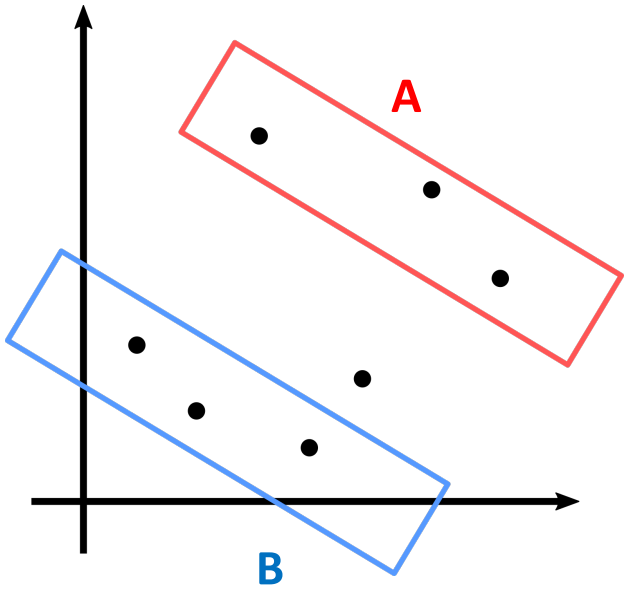
The problem text is missing the condition

A pair <p, q> must satisfy $x_p \leq x_q$ and $y_p \leq y_q$

In this slide we consider the problem J with the condition

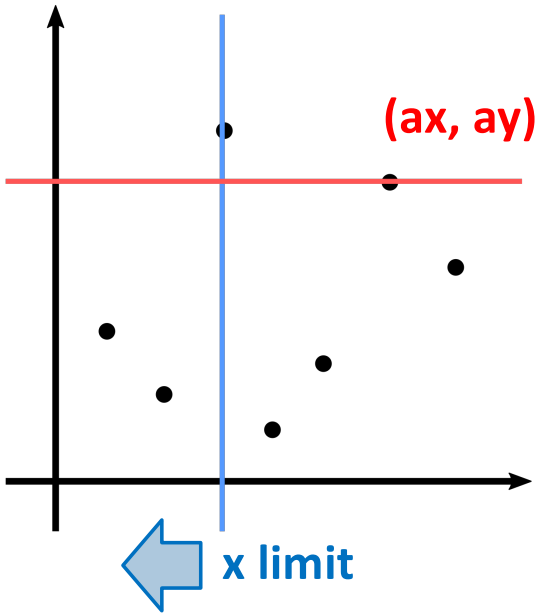
Solution

1. The upper-right candidates A: no more points in upper right area
2. The lower-left candidates B: no more points in lower left area



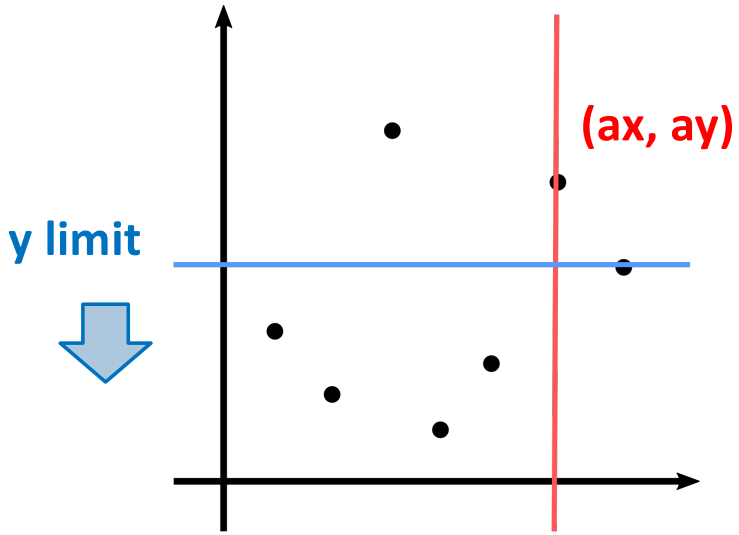
Solution

1. Select an upper-right point (ax, ay) from A
2. x-limit = min{ x | (x, y) in the given set, ay ≤ y}



Solution

1. Select an upper-right point (ax, ay) from A
2. x-limit = min{ x | (x, y) in the given set, ay ≤ y}
3. y-limit = min{ y | (x, y) in the given set, ax ≤ x}



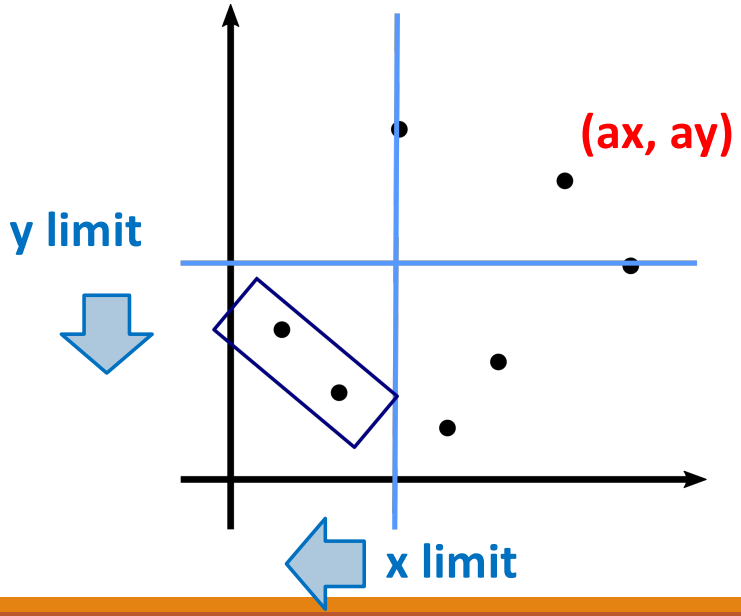
Solution

1. The valid lower-left points (bx, by) must satisfy

   bx ≤ x-limit and by ≤ y-limit

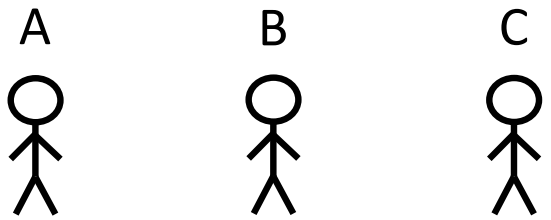
2. If B is ordered by x-increasing and y-decreasing, valid lower-left points are consecutive in B

3. Count the valid points in B with binary search algorithm



K: Distributing the Treasure

Problem Setting



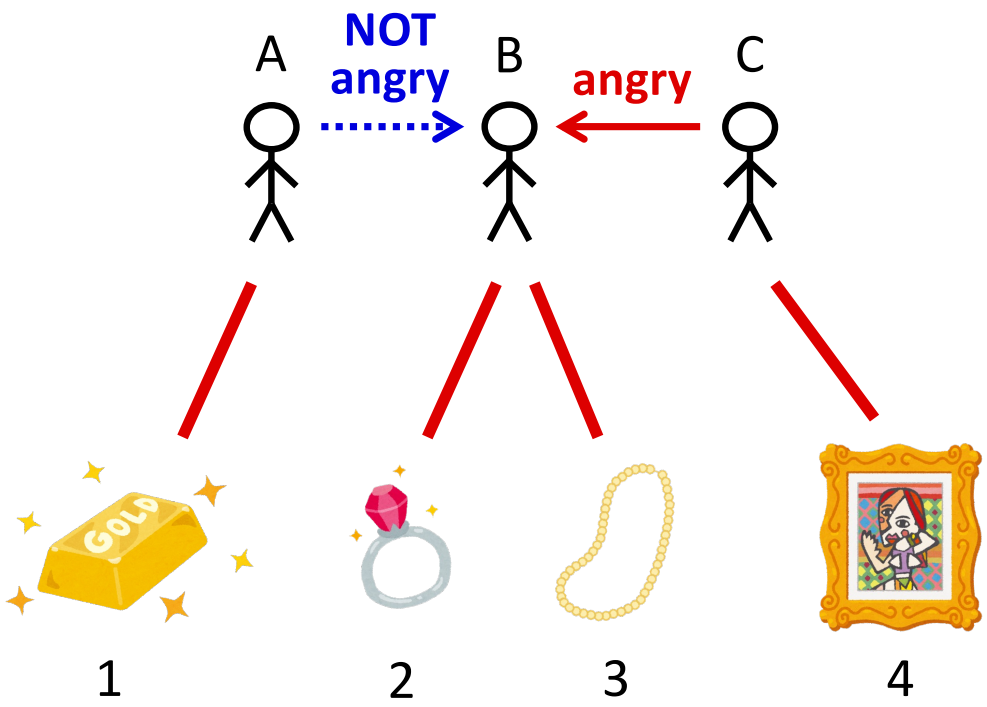
Value Estimation

	1	2	3	4
A	10	9	9	5
B	4	4	4	4
C	100	2	1	1

1 2 3 4

Distribute Items to Members
without Making Anyone Angry

Problem Setting

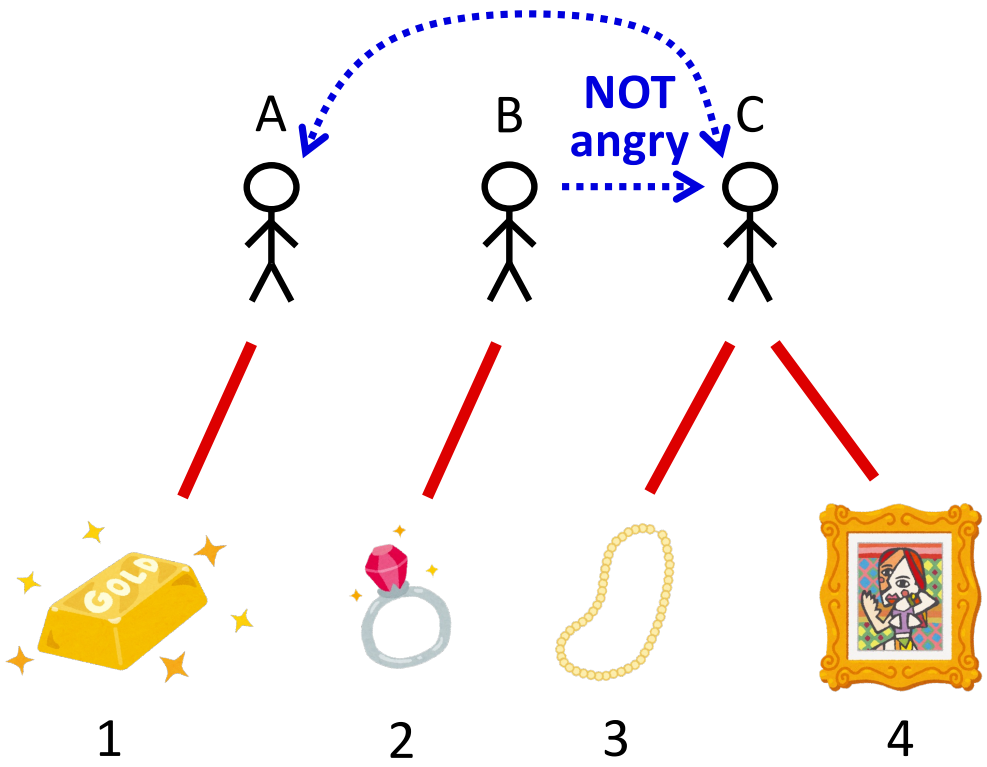


Distribute Items to Members
without Making Anyone Angry

Value Estimation

	1	2	3	4	
A	**10**	**9**	~~**9**~~	5	$10 \geq 9$
B	4	**4**	**4**	4	
C	100	**2**	~~**1**~~	**1**	$1 < 2$

Problem Setting

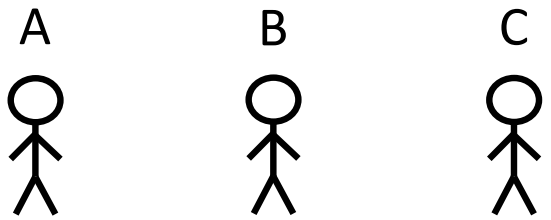


Distribute Items to Members
without Making Anyone Angry

Value Estimation

	1	2	3	4	
A	10	9	9	5	$10 \geq 9$
B	4	4	4	4	$4 \geq 4$
C	100	2	1	1	$2 \geq 0$

Problem Setting



Value Estimation

	1	2	3	4
A	10	9	9	5
B	4	4	4	4
C	100	2	1	1

1 2 3 4

Distribute Items to Members
without Making Anyone Angry
(Envy-Free up to any item; EFX)

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5$
$1 \leq v_{i,j} \leq 2\times10^5$

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5$
$1 \leq v_{i,j} \leq 2\times10^5$

If $n \geq m$, then you can distribute the items arbitrarily as long as every member receives **at most one** item.

$\rightarrow$ Assume $n < m$, and then $n \leq 447$ as $n^2 < nm \leq 2\times10^5$

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

If $n \geq m$, then you can distribute the items arbitrarily as long as every member receives **at most one** item.

$\rightarrow$ Assume $n < m$, and then $n \leq 447$ as $n^2 < nm \leq 2\times10^5$

You can always find a good distribution in $O\left(\underbrace{n^2(m-n)}_{\leq 4\times10^7}\right)$ time!

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

	1	2	3	4
A	10	9	9	5
B	100	2	1	1

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

	1	2	3	4
A	**10**	9	9	5
B	100	2	1	1

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

	1	2	3	4
A	**10**	9	9	5
B	100	**2**	1	1

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

	1	2	3	4
A	**10**	9	9	5
B	100	**2**	**1**	1

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

		1	2	3	4	
	A	**10**	9	9	**5**	
Angry	B	**100**	**2**	**1**	~~**1**~~	$3 < 100$

		1	2	3	4	
Angry	A	**10**	**9**	**9**	~~**5**~~	$10 < 18$
	B	100	**2**	**1**	**1**	

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.

Exchange

↓

Both are fine!

	1	2	3	4
A	10	**9**	**9**	5
B	**100**	2	1	**1**

	1	2	3	4
A	10	**9**	**9**	**5**
B	**100**	2	1	1

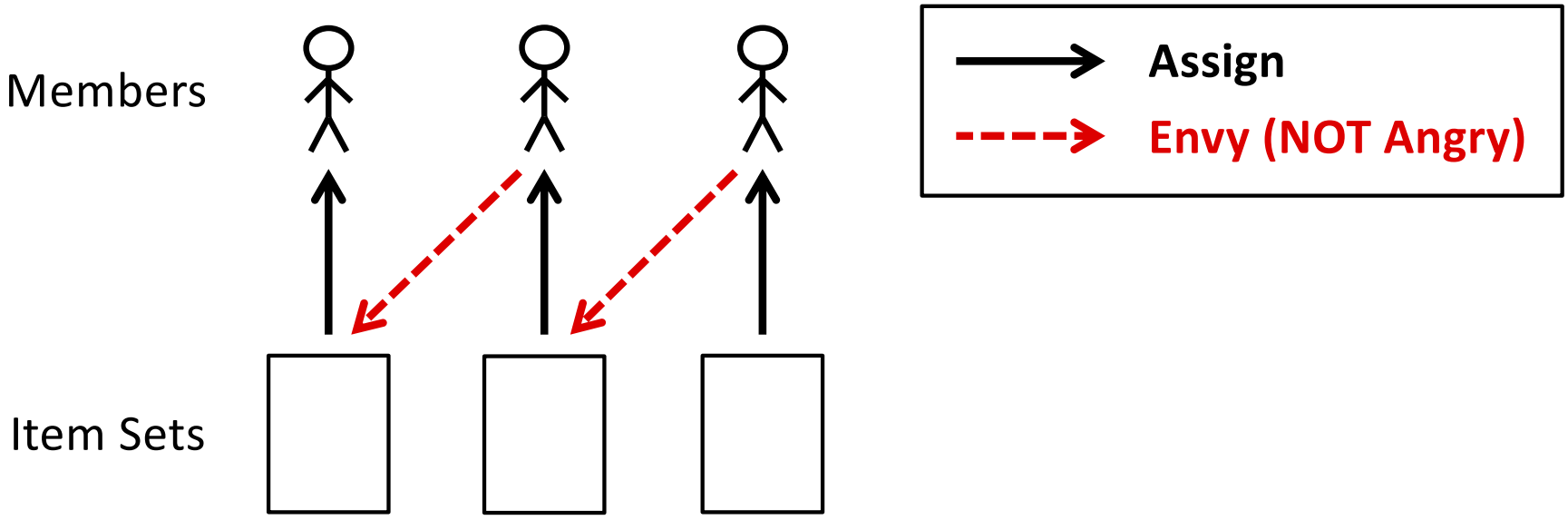
Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.



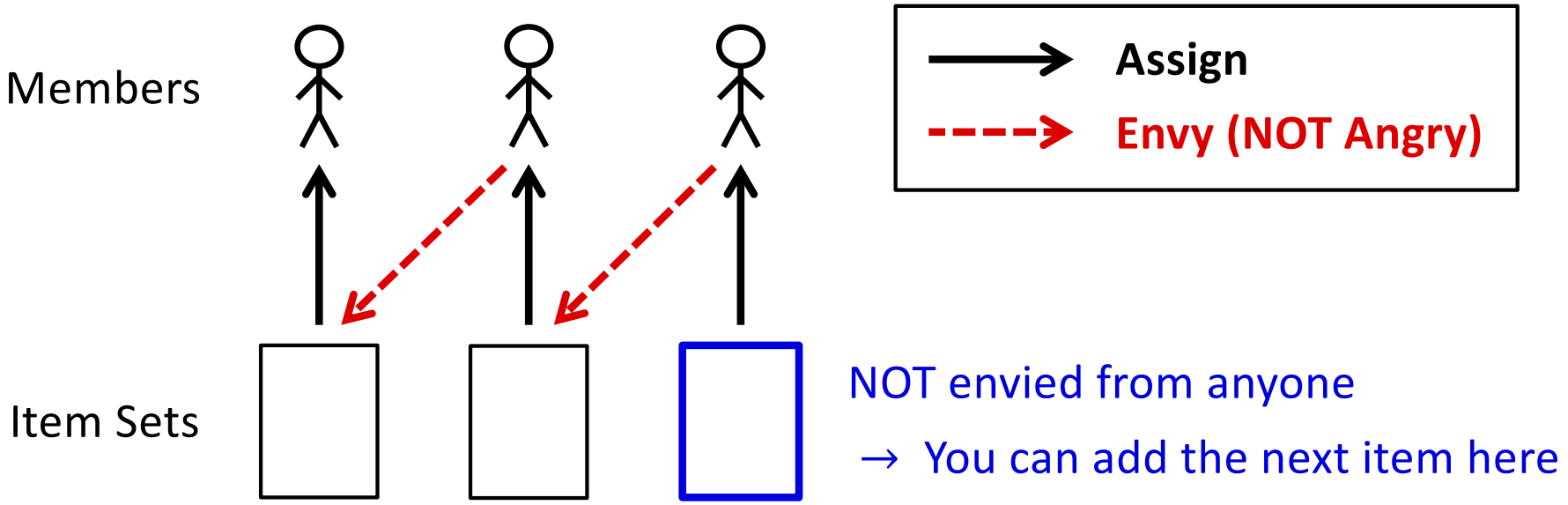
Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.



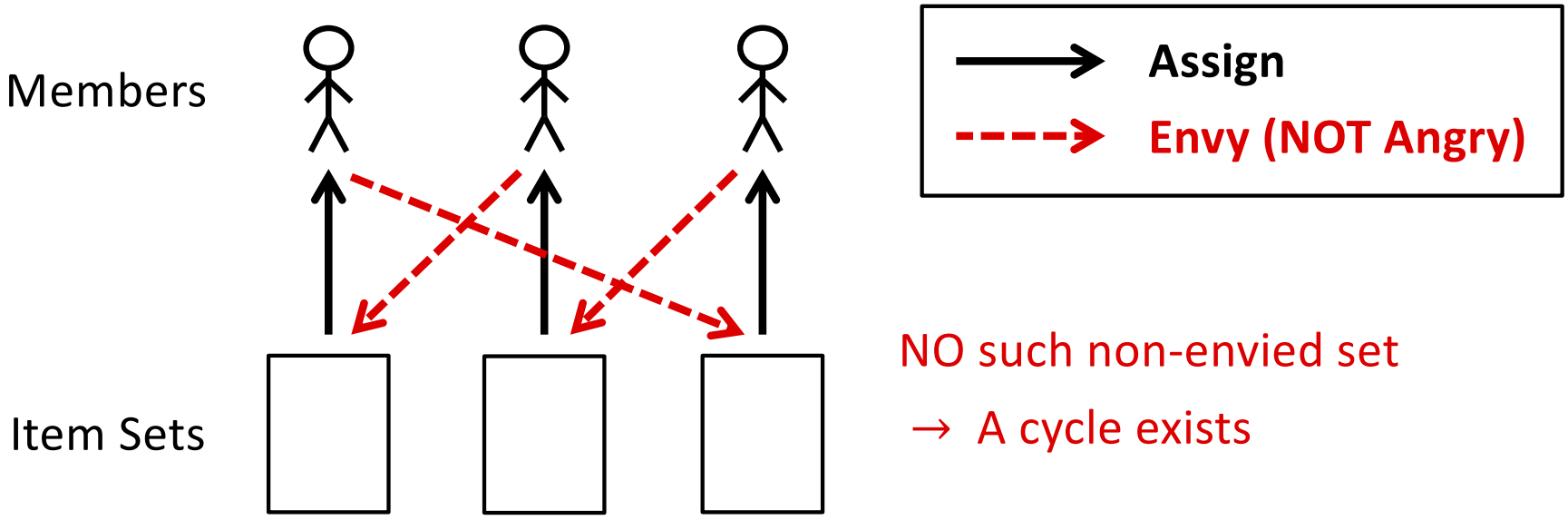
Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.



NO such non-envied set

→ A cycle exists

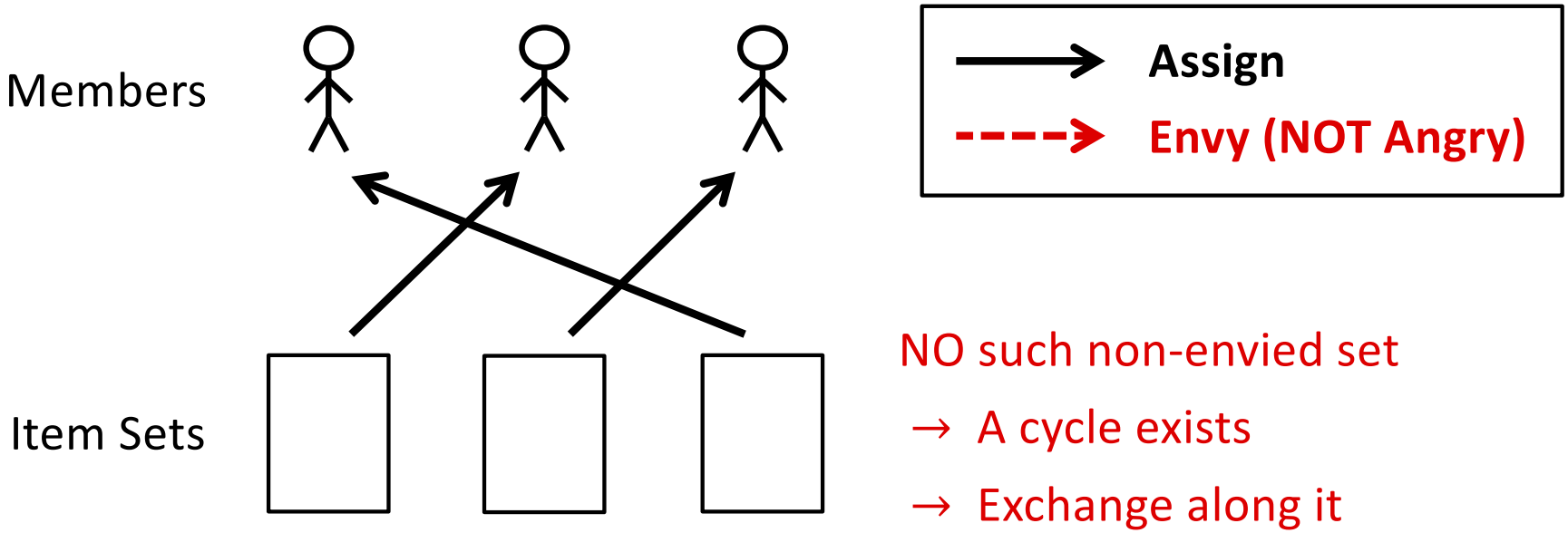
Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.



NO such non-envied set
→ A cycle exists
→ Exchange along it

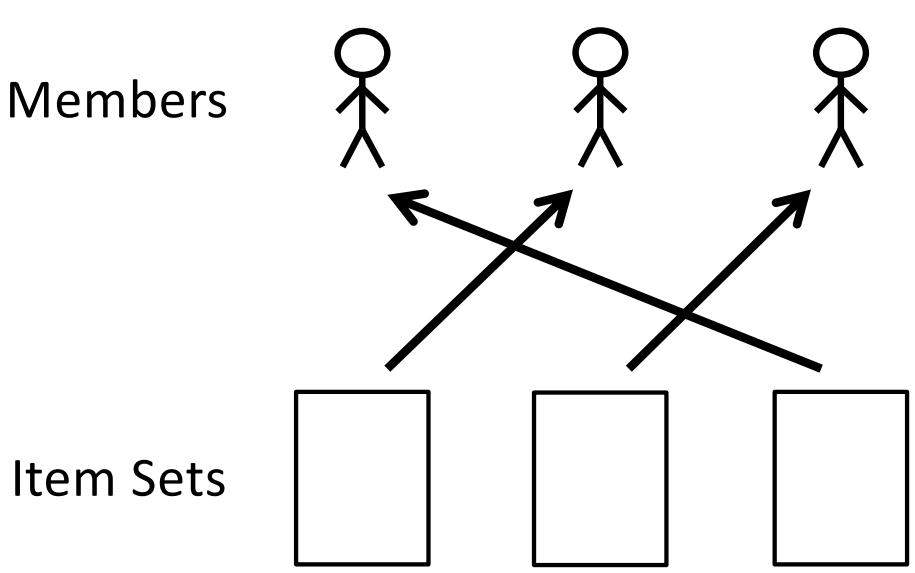
Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Idea]

Distribute the items in descending order of values appropriately, where sometimes exchange the sets of received items.



Assign

Envy (NOT Angry)

After exchange, for everyone,
the estimated values increase

→ Max-weight matching is enough!

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Procedure]

1. While $j = 1, 2, \ldots, m$
2. If no non-envied item set exists
3. Find a max-weight matching
   in the assign-envy graph and exchange
4. Add item j to some non-envied item set X_i
   (Nobody gets angry by this addition)

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

[Procedure]

1. While $j = 1, 2, \ldots, m$
2. If no non-envied item set exists
3. **Find a max-weight matching**
   in the assign-envy graph and exchange
4. Add item j to some non-envied item set X_i
   (Nobody gets angry by this addition)

$O(n^3)$ time by Hungarian method?

→ We can update the current assignment in $O(n^2)$ time

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

$\mathrm{O}(n^3)$ time by Hungarian method?

$\rightarrow$ We can update the current assignment in $\mathrm{O}(n^2)$ time

Why?

Because no improving cycle exists
for the assignment just before adding item $j-1$,
all the new cycles must intersect the item set containing $j-1$.

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

$\mathrm{O}(n^3)$ time by Hungarian method?

→ We can update the current assignment in $\mathrm{O}(n^2)$ time

Why?

Because no improving cycle exists
for the assignment just before adding item $j-1$,
all the new cycles must intersect the item set containing $j-1$.

→ It suffices to find a shortest cycle intersecting a vertex,
which has **exactly one leaving edge** (the assign edge)

→ Just find a shortest path in DAG (by removing that edge)!

Solution

n: # of Members
m: # of Items
$v_{i,j}$: Value of Item j for Member i

$nm \leq 2\times10^5,\ n \leq 447$
$1 \leq v_{i,j} \leq 2\times10^5$

$\mathrm{O}(n^3)$ time by Hungarian method?

$\rightarrow$ We can update the current assignment in $\mathrm{O}(n^2)$ time

This is required only after every item set has at least one item

$\rightarrow$ $\mathrm{O}\big(n^2(m-n)\big)$ time in total $(n^2(m-n) \leq 4\times10^7)$

Other solutions

- Using primal-dual update of weighted matching problem
- Just resolve a cycle in the assign-envy graph greedily
  (very fast but no better bound is known than $\mathrm{O}(n^3m)$ time)