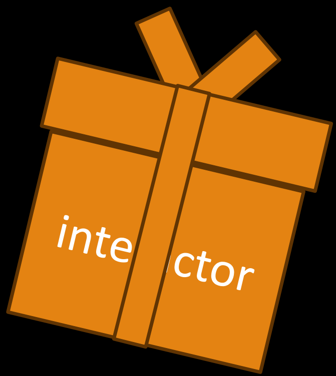
Commentaries on Problems

JUDGE TEAM

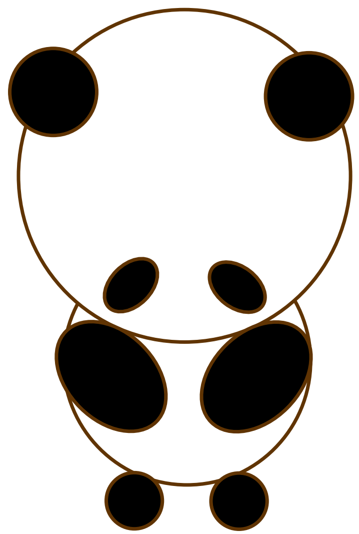
ICPC 2023 2024 ASIA YOKOHAMA REGIONAL

BLACK FRIDAY

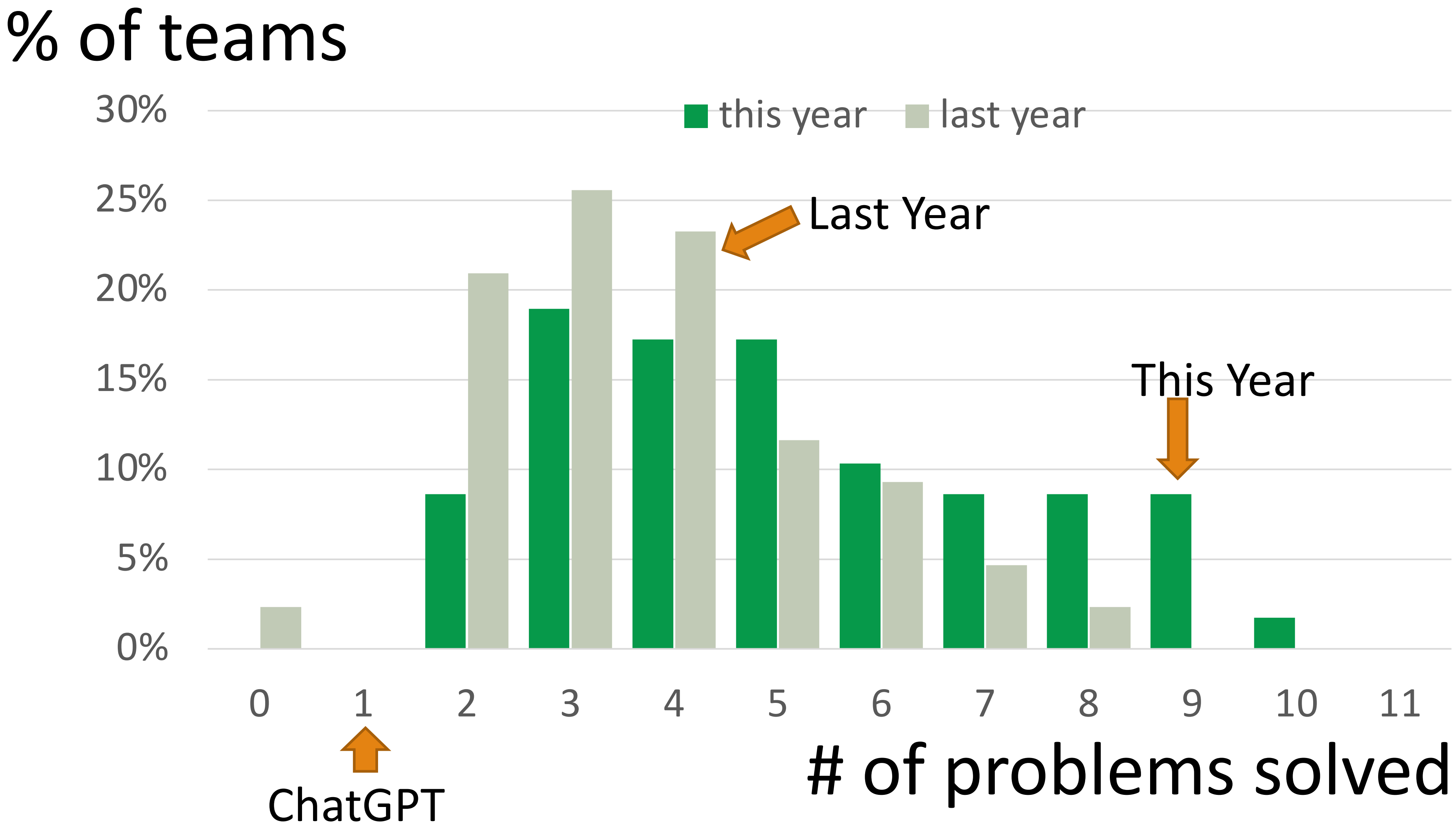


discount difficulties

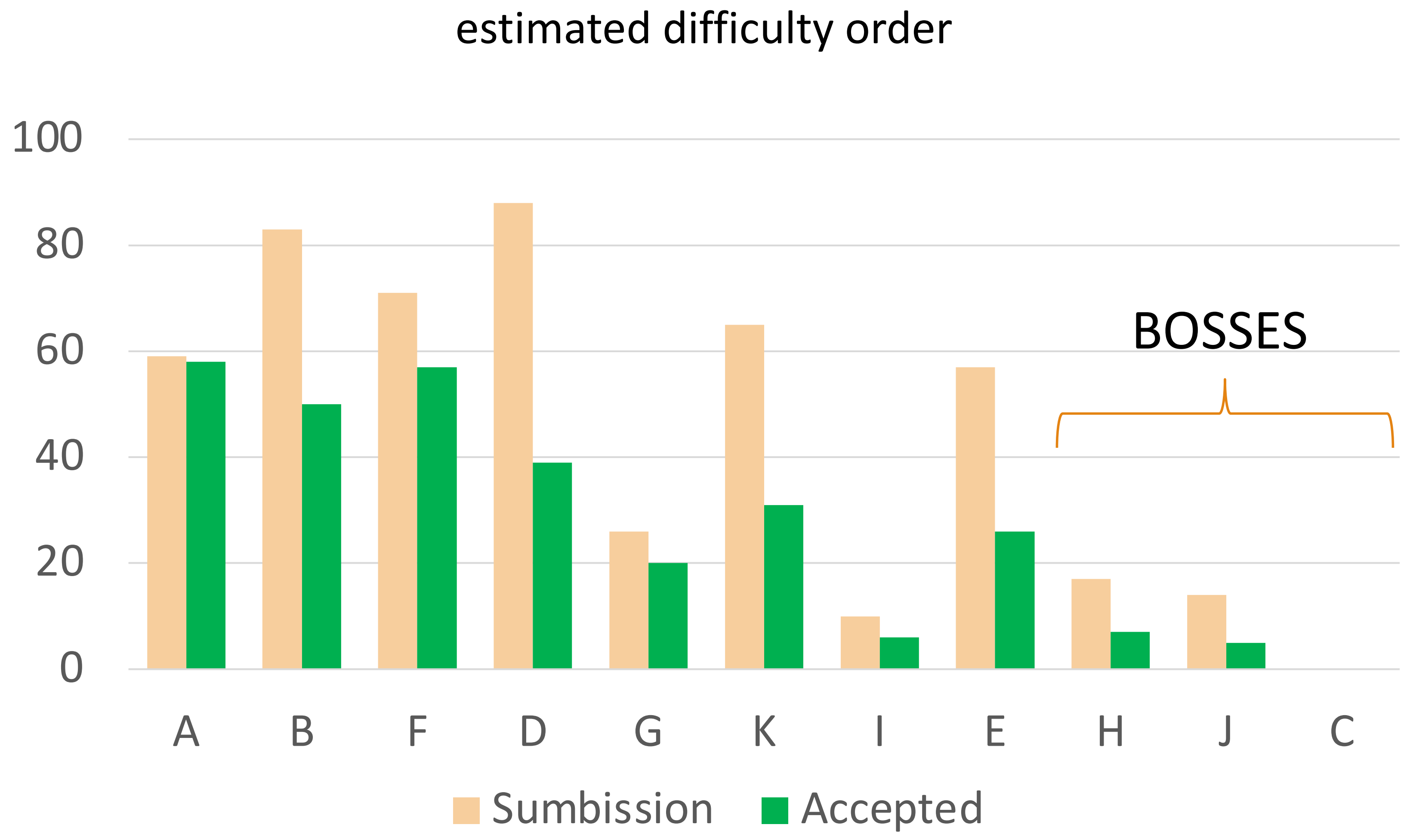
Sorry about the accident …



Solved vs. Teams @Freeze



Problem vs. #Teams @Freeze



A: Yokohama Phenomena

PROPOSER: KAZUHIRO INABA
AUTHOR: TOMOHARU UGAWA

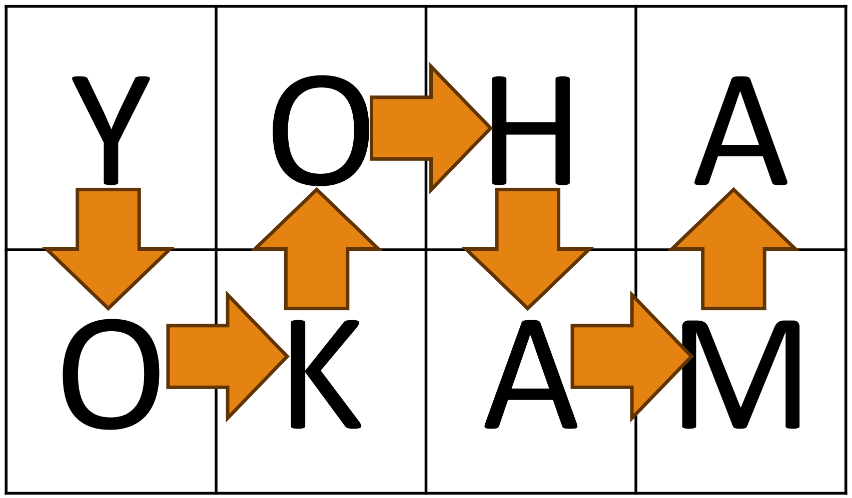
Problem Description

Count “YOKOHAMA” hidden in the board

Y	O	H	A
O	K	A	M

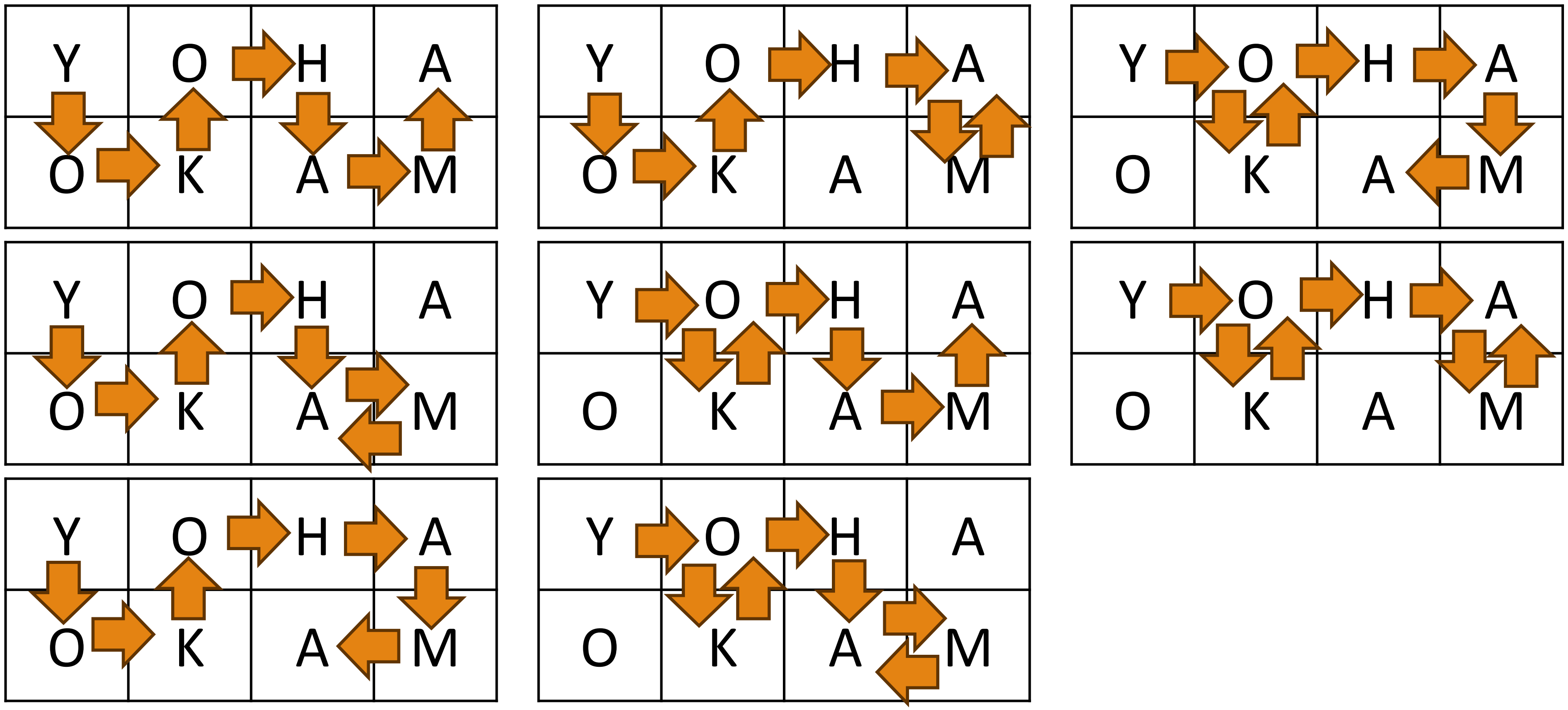
Problem Description

Count “YOKOHAMA” hidden in the board



Problem Description

Count “YOKOHAMA” hidden in the board



Any enumeration will work

- depth-first search
- dynamic programming

Y	O	H	A
O	K	A	M

1			

Y

1	1		
1			

O

	1		
1	2		

K

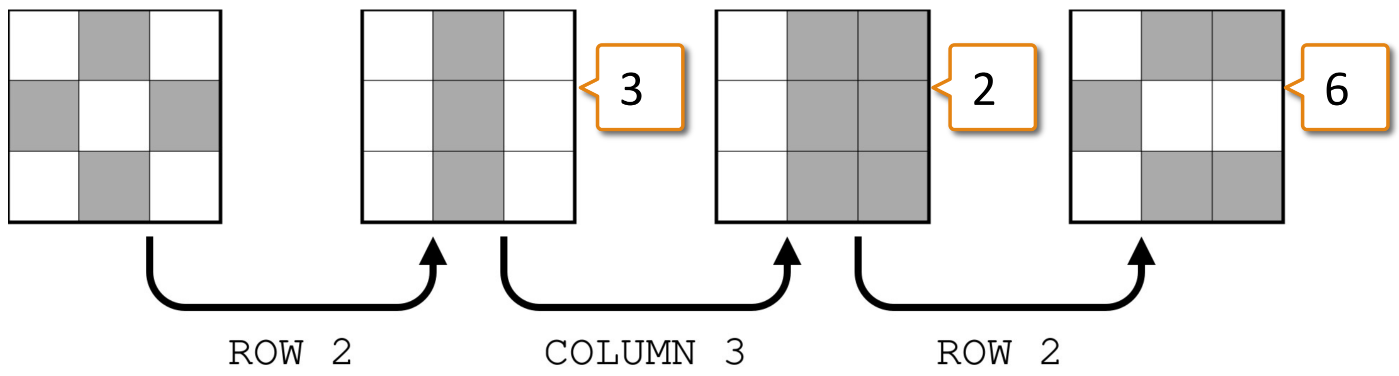
…

F: Color Inversion on a Huge Chessboard

PROPOSER: KOHEI MORITA
AUTHOR: KOHEI MORITA

Problem

- Given $N, Q (1 \leq N, Q \leq 500{,}000)$ (as usual)
- You have to process Q queries for $N \times N$ chessboard.
  - Flip color of a row
  - Flip color of a column
- Print # of areas (= same color components) after each query



Key Point

- You can notice that each area forms rectangle.
  - Let’s try with a random case.

```
bash-3.2$ ./a.out
Random test with n = 20 / q = 100
#.#####.##..#.###..#
#.#####.##..#.###..#
.#.....#..##.#...##.
#.#####.##..#.###..#
.#.....#..##.#...##.
#.#####.##..#.###..#
#.#####.##..#.###..#
#.#####.##..#.###..#
#.#####.##..#.###..#
#.#####.##..#.###..#
.#.....#..##.#...##.
.#.....#..##.#...##.
#.#####.##..#.###..#
#.#####.##..#.###..#
.#.....#..##.#...##.
.#.....#..##.#...##.
#.#####.##..#.###..#
.#.....#..##.#...##.
.#.....#..##.#...##.
.#.....#..##.#...##.
```

  - Why: row-i color is same with row-1 or inversion of row-1

Solution

- Managing row-1 color & column-1 color.
  - And, (# of connected component) of row-1 & column-1.
- Print (# of area of row-1) * (# of area of column-1) after query
- You can process each query in $O(1)$ time, total time complexity is $O(N+Q)$

B: Rank Promotion

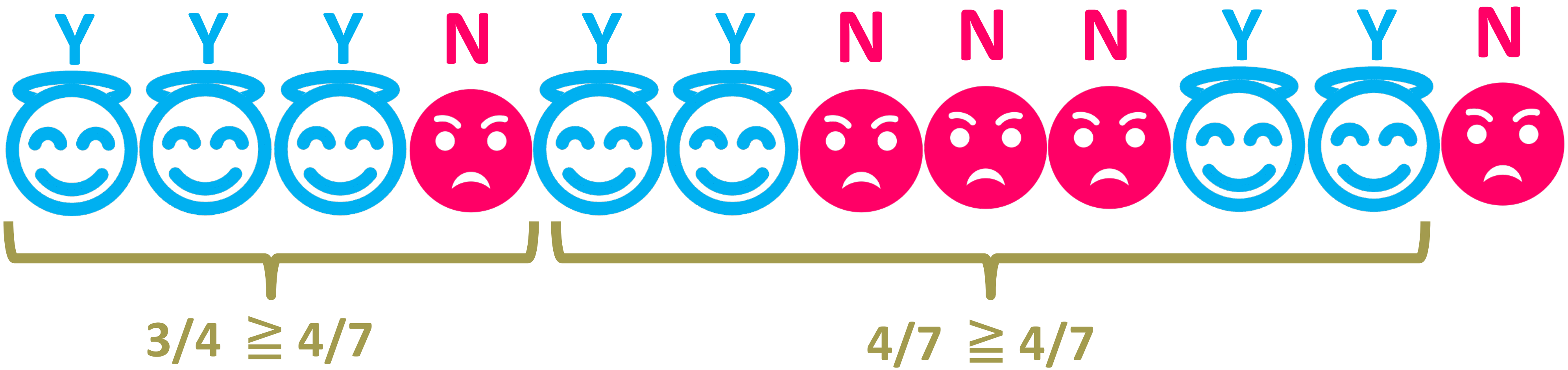
PROPOSER: KAZUHIRO INABA
AUTHOR: KAZUHIRO INABA

$n \leqq 500000$
$c \leqq 200$

Problem

If a sufficiently long (≧c) range contains Y's in a sufficiently high (≧p/q) ratio, rank += 1.
What's the final rank?

Sample Input: c=4, p/q = 4/7



Solution: O(nc)

No need to think about too-long (≧ 2c) ranges. Just check the Y-ratio of all the len ≦ 2c-1 substrings.

If a 2c sequence has a high Y-ratio,

ratio(Y) ≧ p/q

either the first or the latter half also has.

ratio(Y) ≧ p/q

or

ratio(Y) ≧ p/q

Advanced Solution : O(n)

You can solve the problem even if the upperbound of c were large.

$$\frac{\sum_{i=1}^{k} x_i}{k} \geq r$$

Average is larger than r.

$$\Updownarrow$$

$$\sum_{i=1}^{k} (x_i - r) \geq 0$$

Sum of x_i-r is above 0.

Maintain the cumulative sum of **`(S[i]=='Y'?1:0)-p/q`** and the max after the last rank promotion. Then, in O(1) you can check if a “higher than p/q” range exists.

D: Nested Repetition Compression

PROPOSER: KENTO EMOTO
AUTHOR: TAKASHI CHIKAYAMA

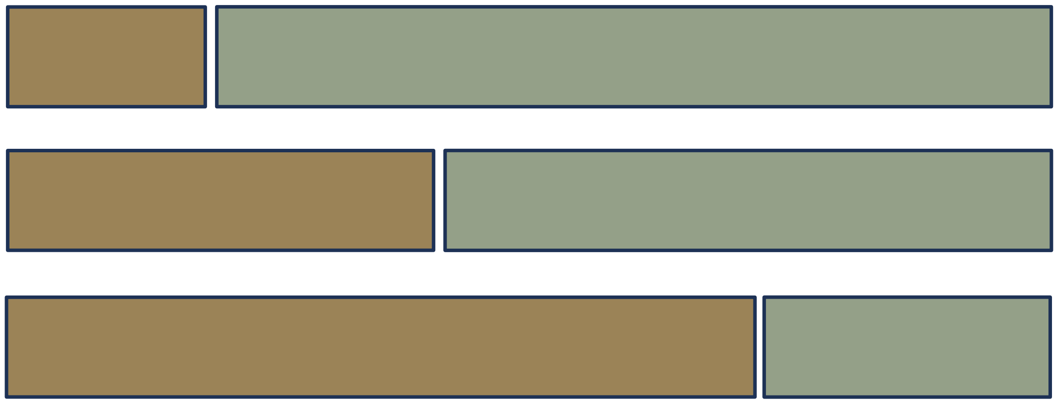
Compression Specifying Repetitions

- Up to nine repetitions of the same string can be specified
  - ababab → 3(ab)
  - abababaaaaa → 3(ab)5(a)
- Repetitions can be arbitrarily nested
  - aaaaaaaaaaaa → 3(4(a))
- As this compression scheme is *context-free,* compression of distinct substrings are *independent*

The Best Compression is Either:

- Repetition of optimally compressed segments,

- Two optimally compressed ones concatenated, or



- As is, i.e., no compression at all.

Preparation: Repetition Table

For all the segments beginning from all the positions in the original string, a table of repeated patterns and their lengths should be prepared.

The table can be made with complexity $O(n^3)$.

Bottom-up Construction

Build a table of the shortest representations for all the string segments, starting from the shortest ones and gradually expanding to longer ones.

- Any segments of length four or less should be *as-is.*
- Knowing the shortest reps for lengths n and less, the shortest for of length $n+1$ segments are either:
  - Concatenation of the shortest reps of the first k characters and the remaining $n+1-k$ characters, for $k = 1, \ldots, n$. This can be checked with complexity of $O(n)$, or
  - Repetition of j identical segments of length $(n+1)/j$ for any factor j of $n+1$. Whether this is possible can be looked up in the repetition table.

The total complexity is $O(n^3)$.

K: Probing the Disk

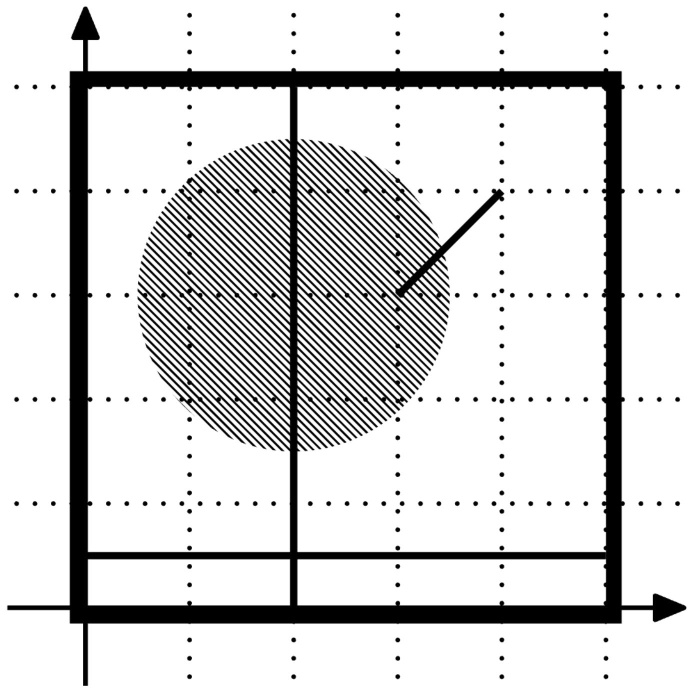
PROPOSER: KIMINORI MATSUZAKI
AUTHOR: KIMINORI MATSUZAKI
MITSURU KUSUMOTO

Problem

Given a disk (radius $\geq$ 100) in a square (side = 10^5), decide the position and the size of the disk, by at most 1024 probes.

Each prove:

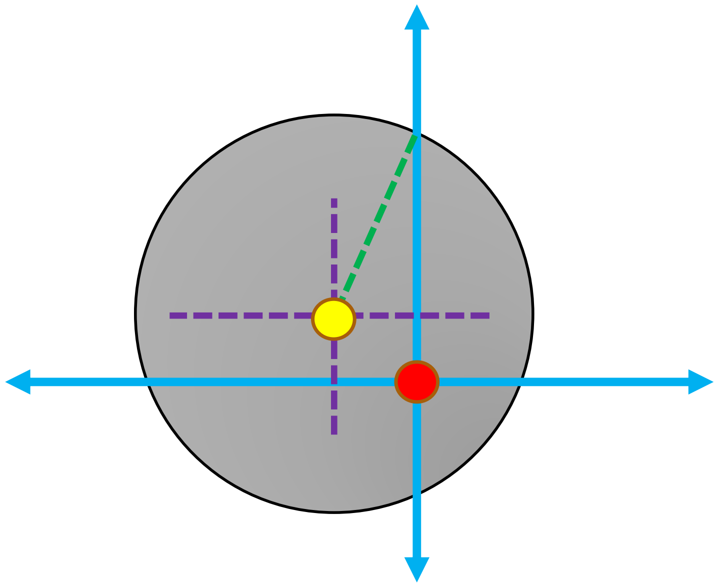
- Query: a line segment
- Answer: length on disk



Key to Solution

"Find a point that is surely in a disk"

If you find a point in a disk, you can solve the problem in 4 more probes.

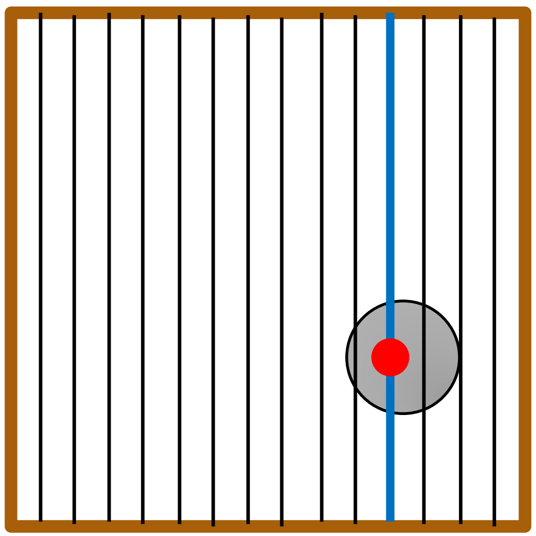


A Simple Solution

1. Probe by vertical lines (1000 probes)
   and find a line with the largest common length

2. Do binary search (11 probes)
   to find a point that is surely in the disk

3. Find the center
   and radius (4 probes)



E: Chayas

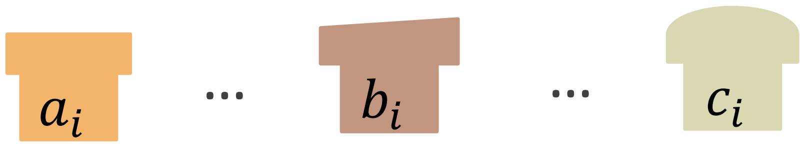
PROPOSER: SOU KUMABE
AUTHOR: SHINYA SHIROSHITA

Overview

There were n chayas (teahouses) in a line.

You have m records showing the following information:

Record i: chaya b_i is between chaya a_i and c_i.

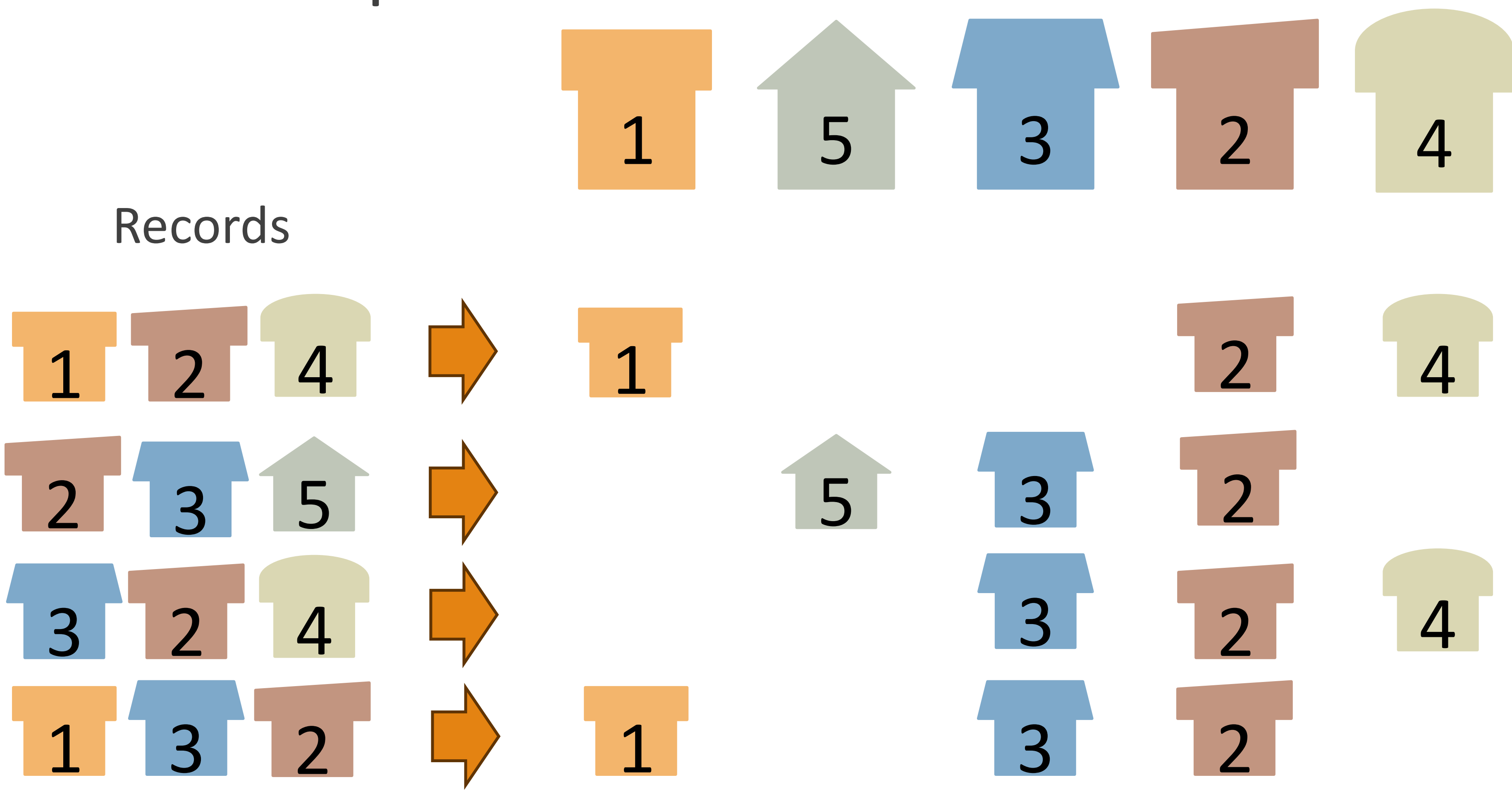


※ Reversing the order is OK

How many orders were there satisfying all the records?

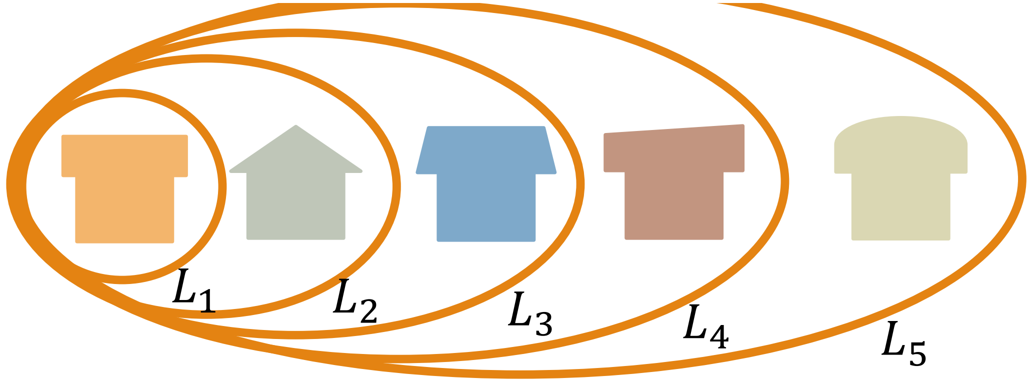
- $3 \leq n \leq 24$
- $1 \leq m \leq n(n-1)(n-2)/2$

Example



※ Reversing the order is OK

Analysis

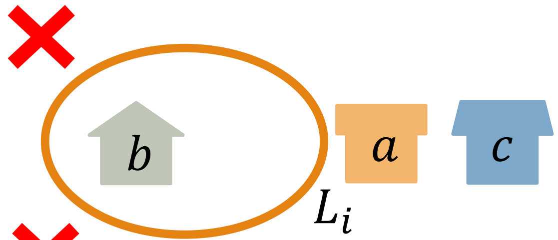


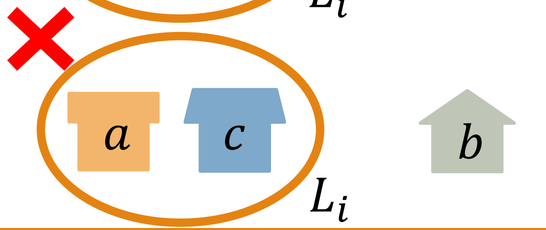
Let's consider when we select chayas from left to right.

Let L_i be the subset of the i chayas from the left.

The condition "b is between a and c" can be formulated as follows:

- For all $1 \leq i \leq n-1$, **NONE** of the below must hold.
  - $b \in L_i$ and none of a, c are not in L_i.
  - $b \notin L_i$ and both of a, c are in L_i.





How to check these conditions quickly?

Analysis

For simplicity, we hereby consider the condition

$b \in S$ and none of a, c are not in S

$= S$ where $\{b\} \subseteq S \subseteq (\text{all chayas}) \setminus \{a, c\}$

for each of the records.

When we create a 2^n boolean tables memorizing the conditions' true or false, the naïve check for each record takes $O(m \cdot 2^n) = \ O(n^3 \cdot 2^n)$, which is too slow.

→ *Let's focus on all the records whose b are the same.*

Precomputation

When we define

$$f(S) = \begin{cases} 1 \text{ if } S = (\text{all chayas}) \setminus \{a_i, c_i\} \text{ for some } (a_i, b_i, c_i), \\ 0 \text{ otherwise,} \end{cases}$$

Then, the subset S containing b contradicts the records if

$$g(S) := \sum_{S \subseteq T} f(T) \geq 1.$$

where $g(S)$ is the sum of fs of the supersets of S.

The calculation of $g(S)$ can be speed up based on **Fast Zeta Transformation**.

Precomputation

The following dp calculates $g(S) = \mathrm{dp}[n][S]$.

$\mathrm{dp}[0][S] = f(S)$ **for each subset** S.
for each chaya $i = 1, \dots, n$:
  for each subset S **of** $i \notin S$:
    $\mathrm{dp}[i][S] = \mathrm{dp}[i-1][S] + \mathrm{dp}[i-1][S \cup \{i\}]$
  for each subset S **of** $i \in S$:
    $\mathrm{dp}[i][S] = \mathrm{dp}[i-1][S]$

S	.	3	2	23	1	13	12	123
dp[0]	0	0	0	1	0	0	1	0
dp[1]	0	0	1	1	0	0	1	0
dp[2]	1	1	1	1	1	0	1	0
dp[3]	1	1	1	1	1	0	1	0

An example where chayas are $\{1, 2, 3, b\}$ and queries are $(1, b, 2)$ and $(2, b, 3)$.

Precomputation

The transformation of the same b can be done in $O(n \cdot 2^n)$.

For other bs, we can calculate in the same transformation when we use different digits, totaling $O(n \cdot 2^n)$ time complexity.

We can solve the other condition in the similar way.

Solution

The solution is equal to the number of sequences where the left i chayas satisfies the records for all $0 \leq i \leq n$.

This can be solved by dynamic programming.

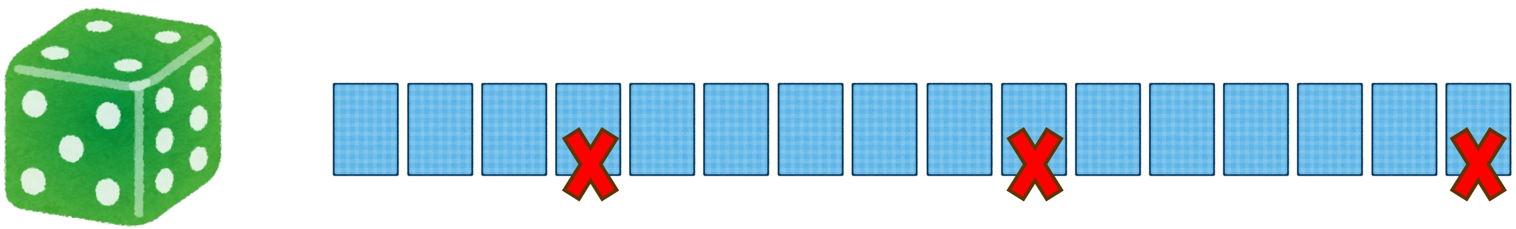
As the condition check of each subset takes $O(1)$ by the precomputation, the time complexity is $O(n \cdot 2^n)$.

G: Fortune Telling

PROPOSER: MITSURU KUSUMOTO
AUTHOR: MITSURU KUSUMOTO

Problem Overview

- n cards are lined up ($2 \leq n \leq 300000$)
- Each time, we roll a die and when it shows x, we remove cards x-th, $(x+6)$-th, $(x+12)$-th, … from left.
- We end this when only one card remains.
- Compute the probability each initial card survives.



Naive DP

dp[n'][k] := "Probability that, when there are
n' cards, card k-th from left survives"

Naive DP

dp[n'][k] := "Probability that, when there are n' cards, card k-th from left survives"

$\Theta(n^2)$ entries!! Too many!! 😭

Dependency

dp[n][:]

Dependency

dp[n][:] **only** depends on

dp[(5/6)n][:]

dp[(5/6)n+1][:]

Dependency

dp[n][:] **only** depends on

dp[$(5/6)n$][:]

dp[$(5/6)n+1$][:] **only** depend on

dp[$(5/6)^2n$][:]

dp[$(5/6)^2n+1$][:]

dp[$(5/6)^2n+2$][:]

Dependency

dp[n][:] **only** depends on

dp[$(5/6)n$][:]
dp[$(5/6)n+1$][:] **only** depend on

dp[$(5/6)^2n$][:]

Required entries for DP calculation may be **much smaller than n^2** ?

Bound

The number of required entries for DP computation is roughly bounded by

$$n \sum_{k=1}^{\infty} k \left(\frac{5}{6}\right)^{k-1}$$

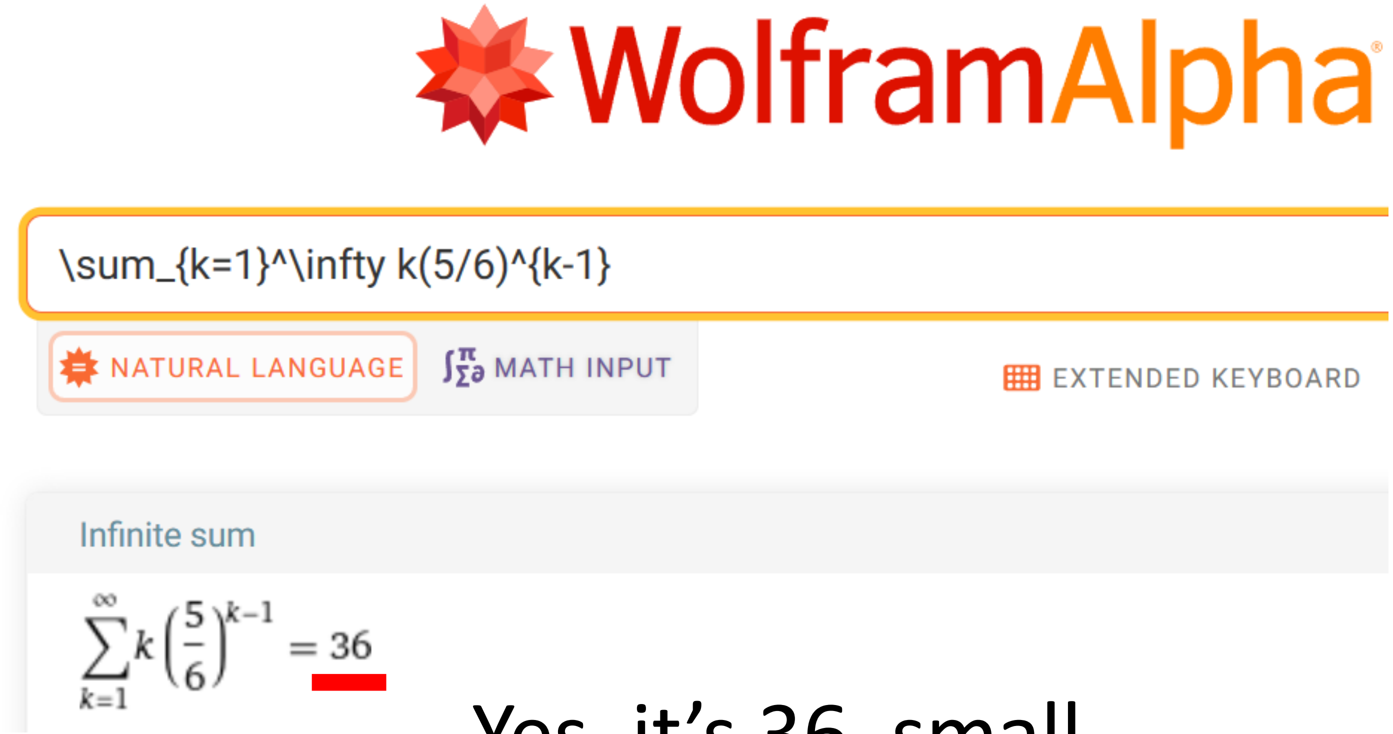
Bound

The number of required entries for DP computation is roughly bounded by

$$n \sum_{k=1}^{\infty} k \left(\frac{5}{6}\right)^{k-1}$$

?

If you can access to Wolfram Alpha…



Yes, it's 36, small.

Another method

You can estimate it without Wolfram Alpha:

- Approximate it by a tiny code
- Differentiate $1+x+x^2+\ldots+x^n = (1-x^{n+1})/(1-x)$ and set $x=5/6$, then take $n\to\infty$.

$$n \sum_{k=1}^{\infty} k \left(\frac{5}{6}\right)^{k-1} = 36n$$

Solution

Compute a DP table with memorization.

In general, if the die has A faces, time complexity is $O(A^3n)$.

H: Task Assignment to Two Employees

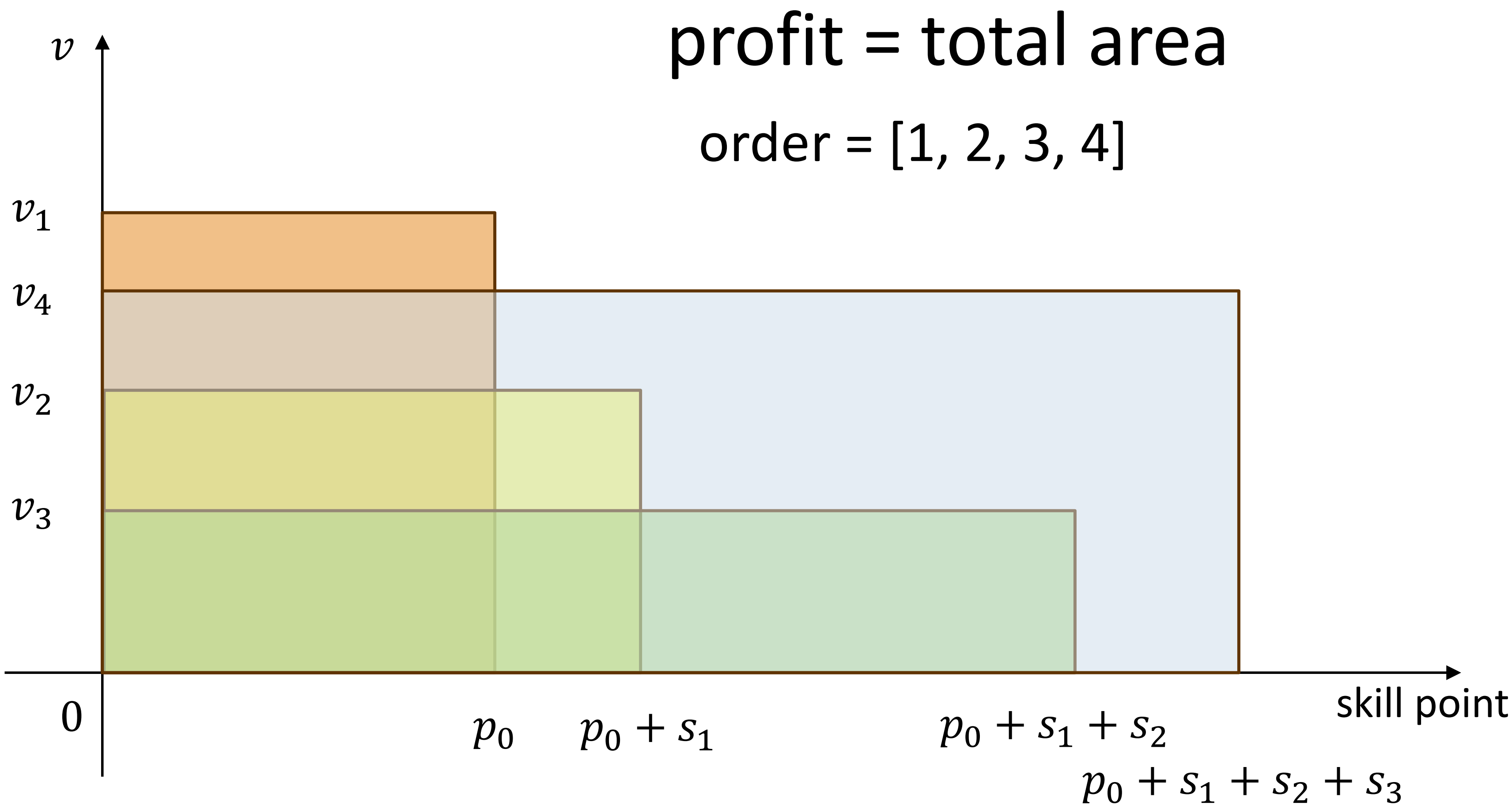
PROPOSER: YOICHI IWATA
AUTHOR: YOICHI IWATA

Problem

Assign tasks to two employees in an appropriate order to maximize the total profit.

- initial skill point: p_0
- task compatibility: $v_{i,j}$
- skill growth: $s_{i,j}$
- profit = current skill point $\times$ $v_{i,j}$
- new skill point = current skill point $+$ $s_{i,j}$

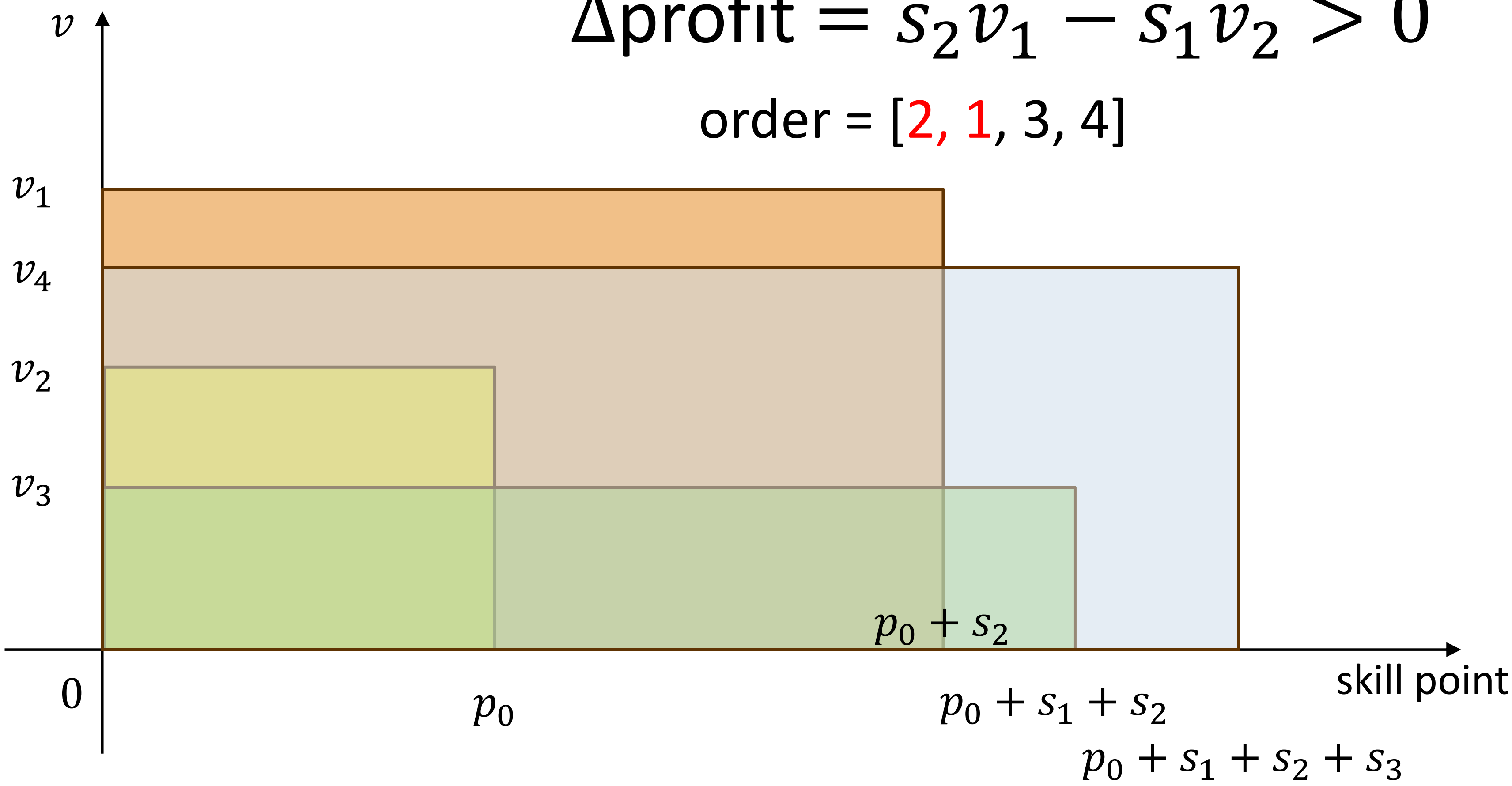
Optimize Ordering for Single Employee



Optimize Ordering for Single Employee

$$\Delta\text{profit} = s_2 v_1 - s_1 v_2 > 0$$

order = [2, 1, 3, 4]



Optimize Ordering for Single Employee

Optimal ordering $= [i_1, i_2, \ldots, i_n]$
s.t. $s_{i_{j+1}} v_{i_j} \leq s_{i_j} v_{i_{j+1}}$

$\Rightarrow$ Sort & Greedy

Key Observation

Optimal profit

$$= \sum_i p_0 v_i + \sum_{i,j} \max(s_i v_j, s_j v_i)$$

Optimize Assignment

x_i: task i is assigned to employee 1

Profit $=$

$$\sum_i p_0 v_{1,i} x_i + \sum_{i,j} \max(s_{1,i} v_{1,j}, s_{1,j} v_{1,i})\, x_i x_j$$
$$+ \sum_i p_0 v_{2,i} \bar{x}_i + \sum_{i,j} \max(s_{2,i} v_{2,j}, s_{2,j} v_{2,i})\, \overline{x_i} \overline{x_j}$$

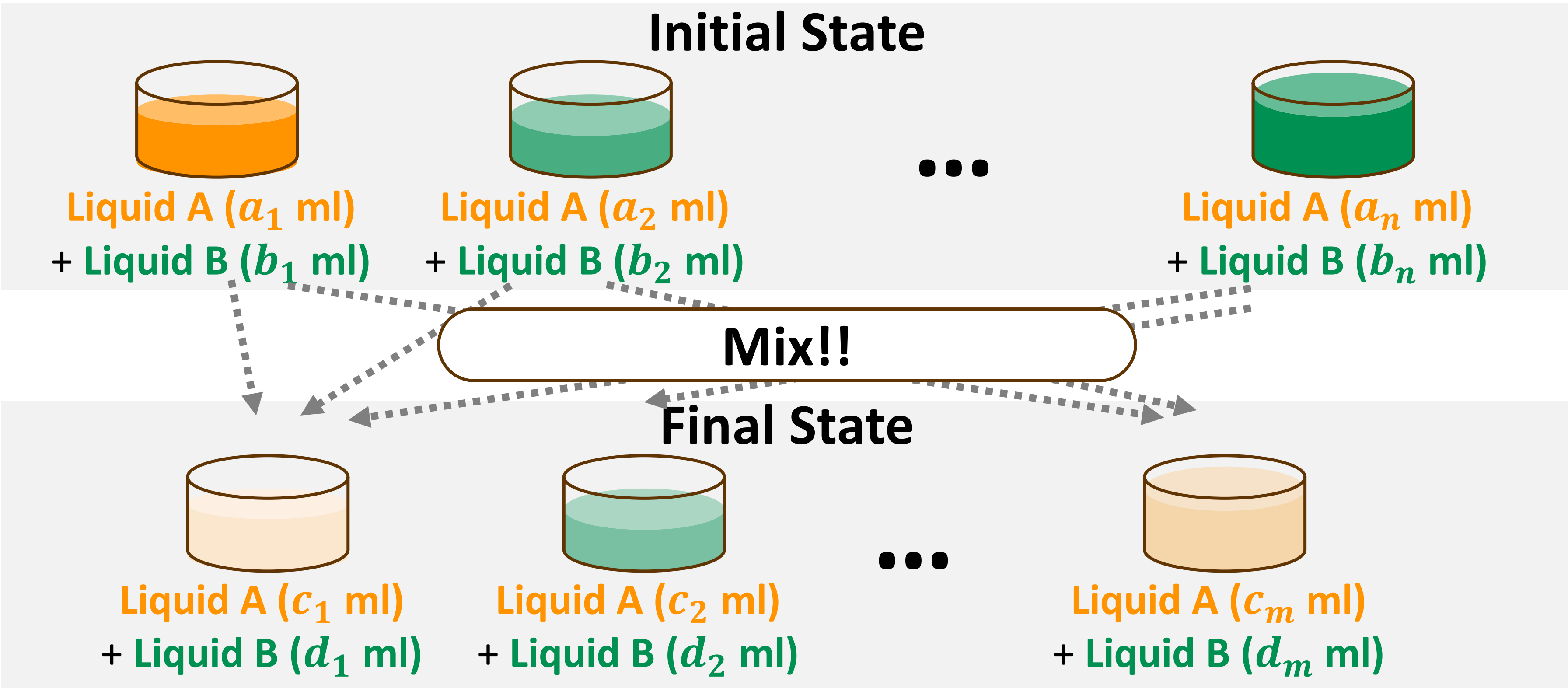
maximization of Quadratic pseudo-Boolean supermodular function → mincut !!!

I: Liquid Distribution

PROPOSER: RYOTARO SATO
AUTHOR: RYOTARO SATO

Problem Overview

Judge whether mixture process below is feasible.

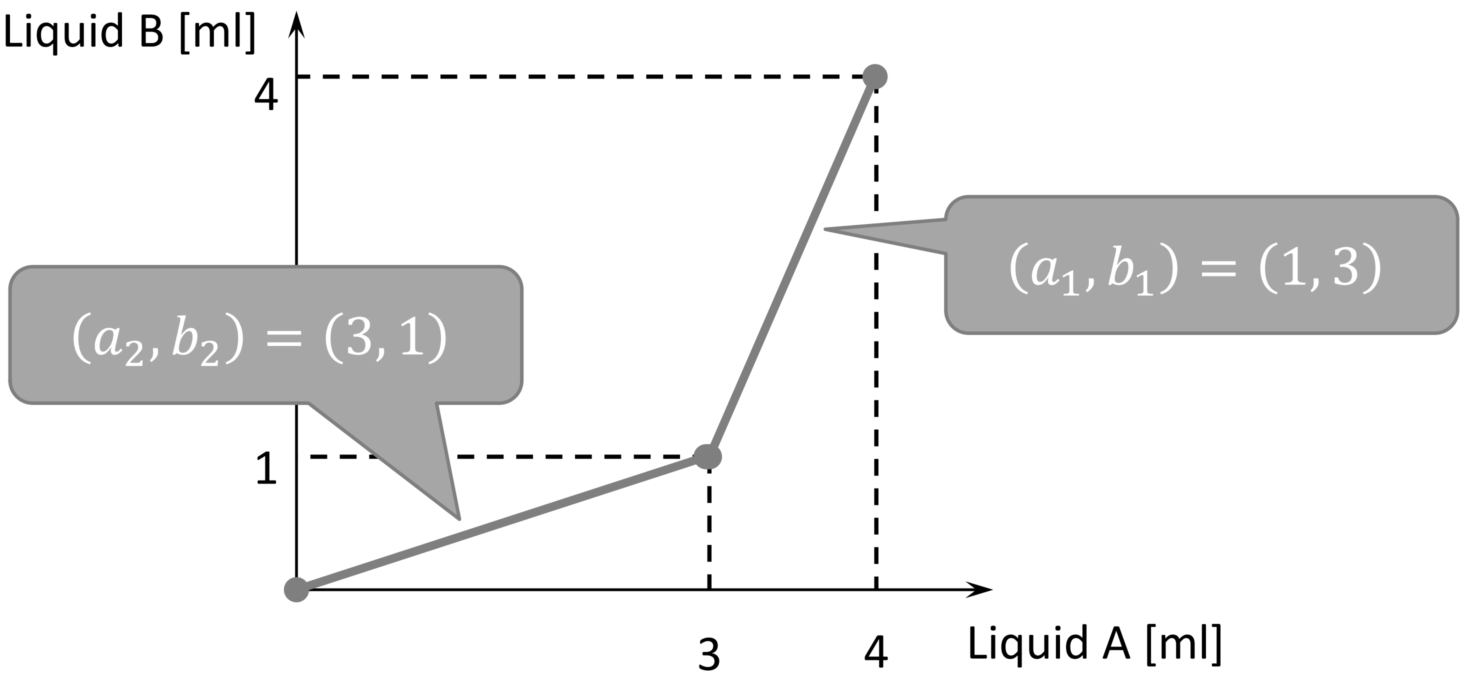


Constraints: $1 \leq n, m \leq 500$, $\sum a_i = \sum c_j$, $\sum b_i = \sum d_j$.

Observation: Curves

Sort all liquids by b_i/a_i (or d_j/c_j) and plot cumulative sum.

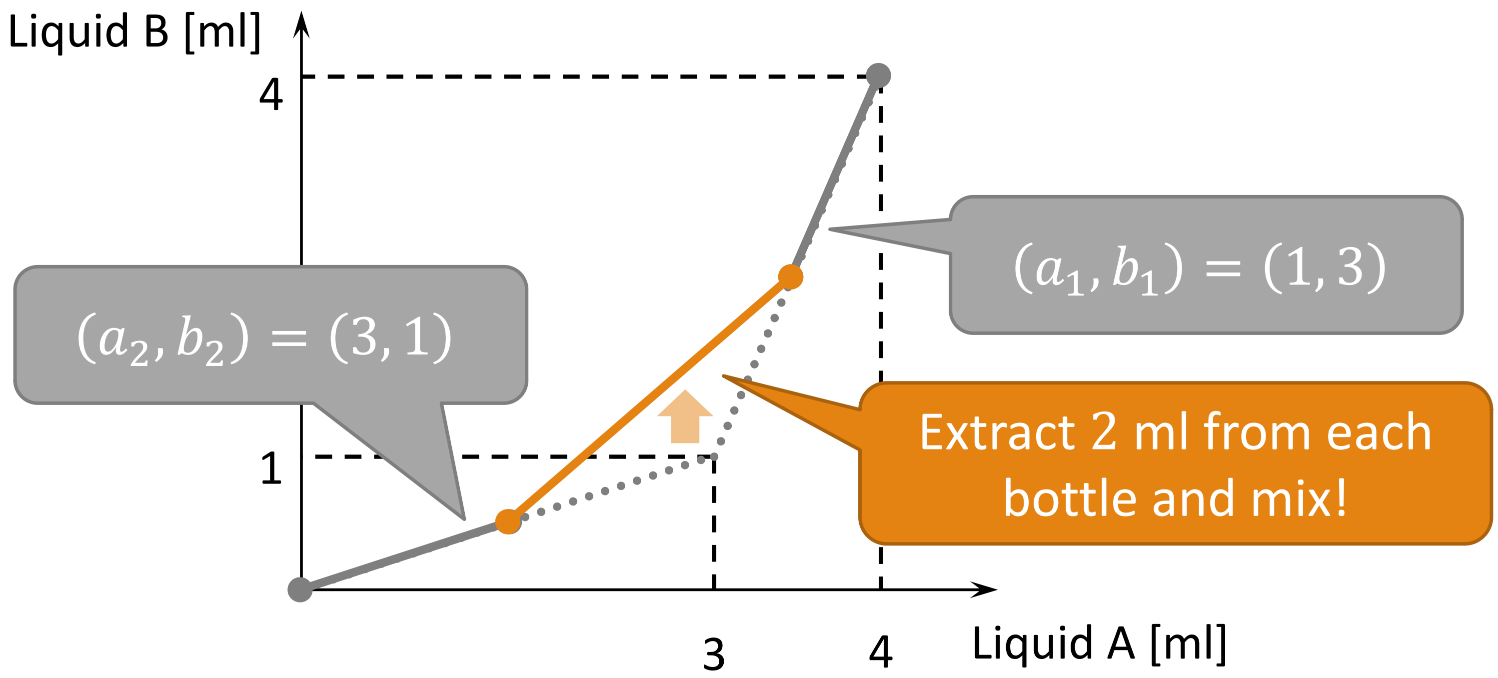
Generated curves are always convex.



Observation: Mixture

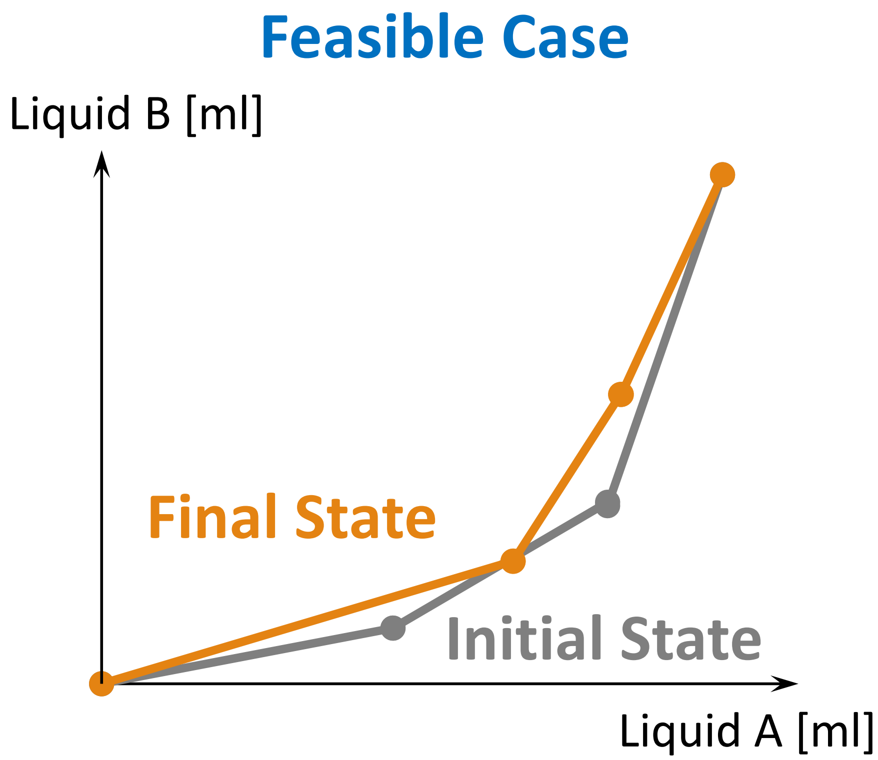
What happens to curves when liquids are mixed?

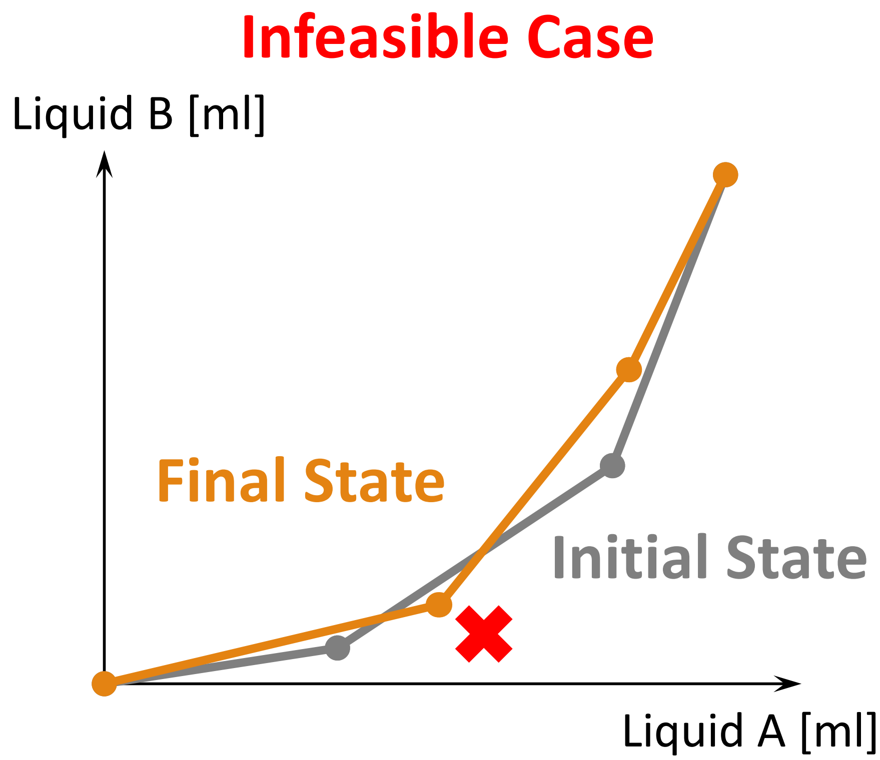
→ Curves **always move upper!**



Solution

Mixture process is feasible if and only if final state curve **NOT** passes under initial state curve.

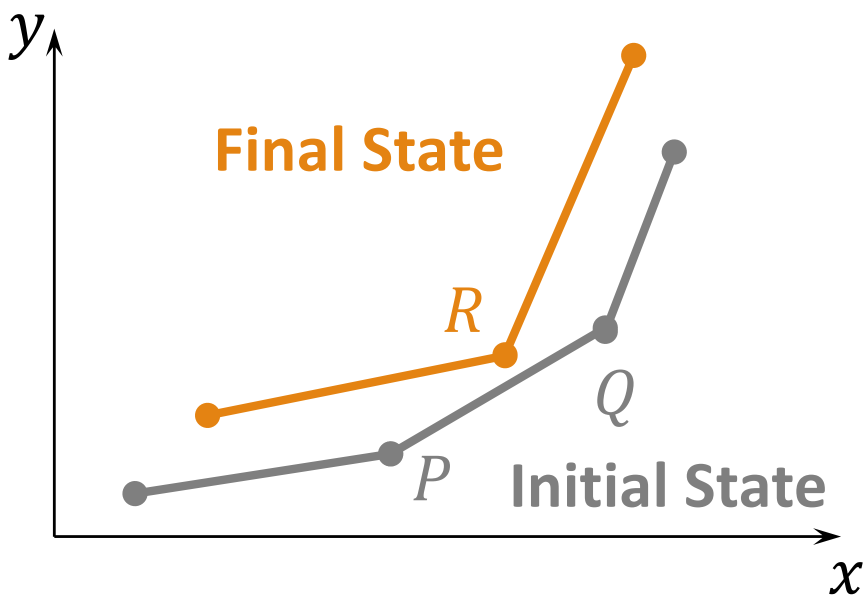




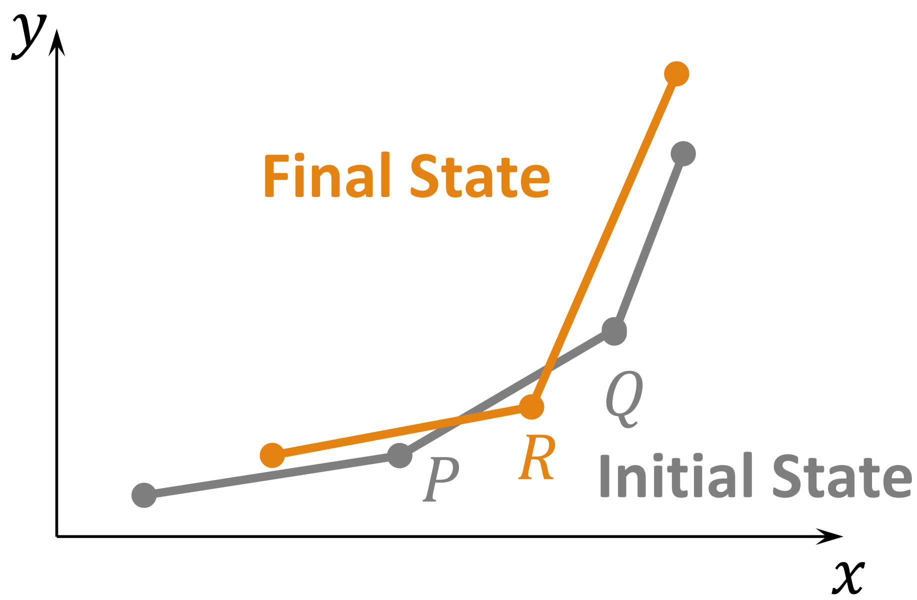
$O(nm)$ Implementation

For each segment PQ of initial curve and each breakpoint R of final curve, check sign of $\overrightarrow{PQ} \times \overrightarrow{PR}$.

$\left(\overrightarrow{PQ} \times \overrightarrow{PR}\right)_z \geq 0$ **is required**



$\left(\overrightarrow{PQ} \times \overrightarrow{PR}\right)_z < 0$ **→ NG**

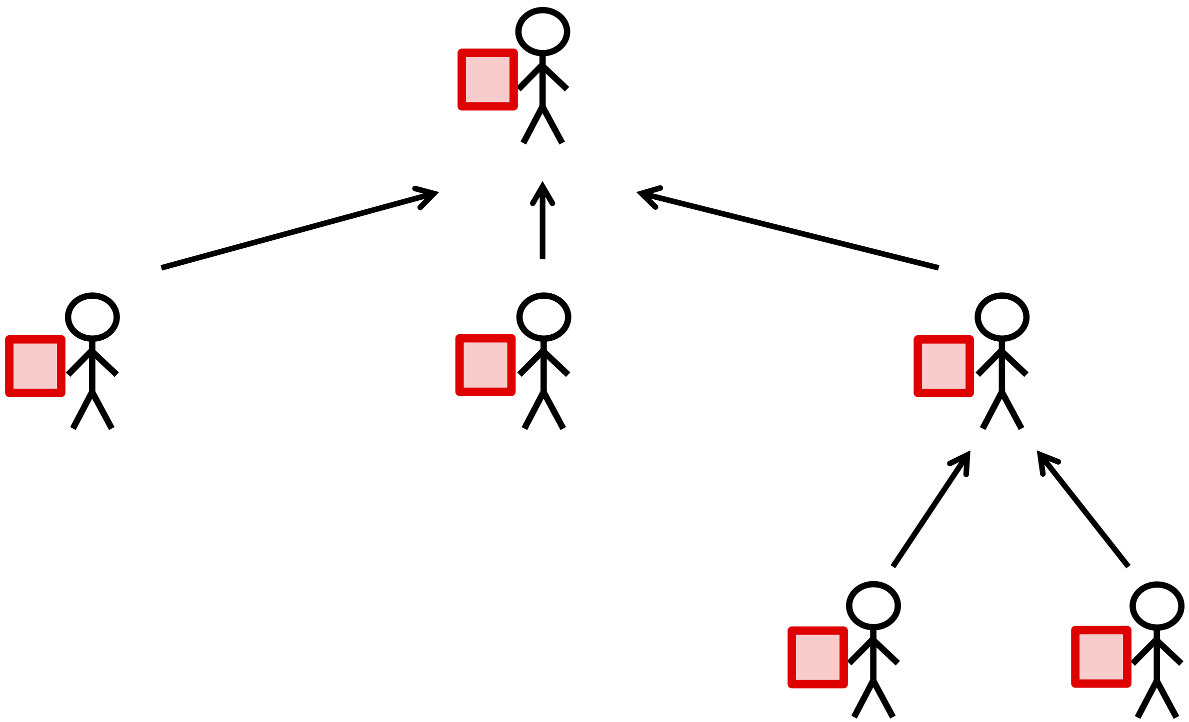


J: Do It Yourself?

PROPOSER: YUTARO YAMAGUCHI
AUTHOR: YUTARO YAMAGUCHI

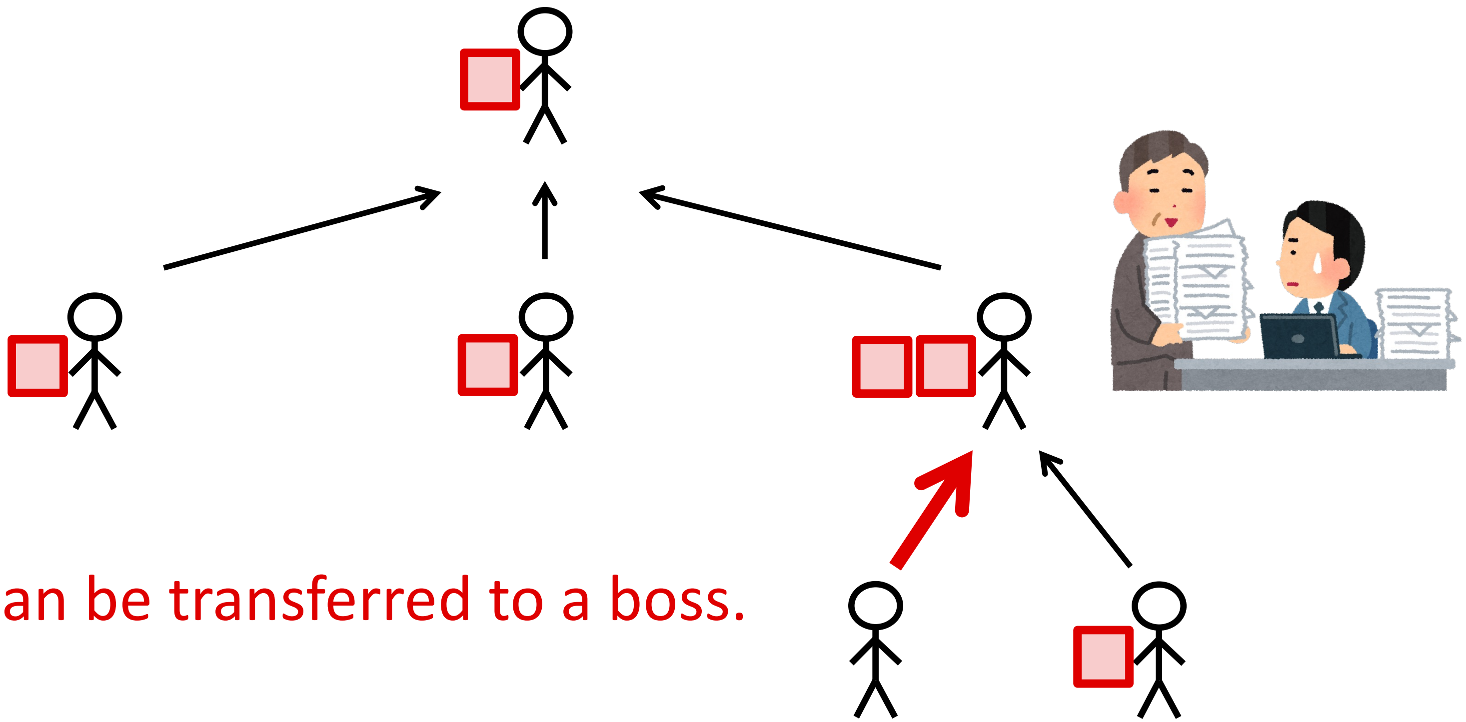
Story

Complete **the tasks** with **the smallest total fatigue** of employees.



Story

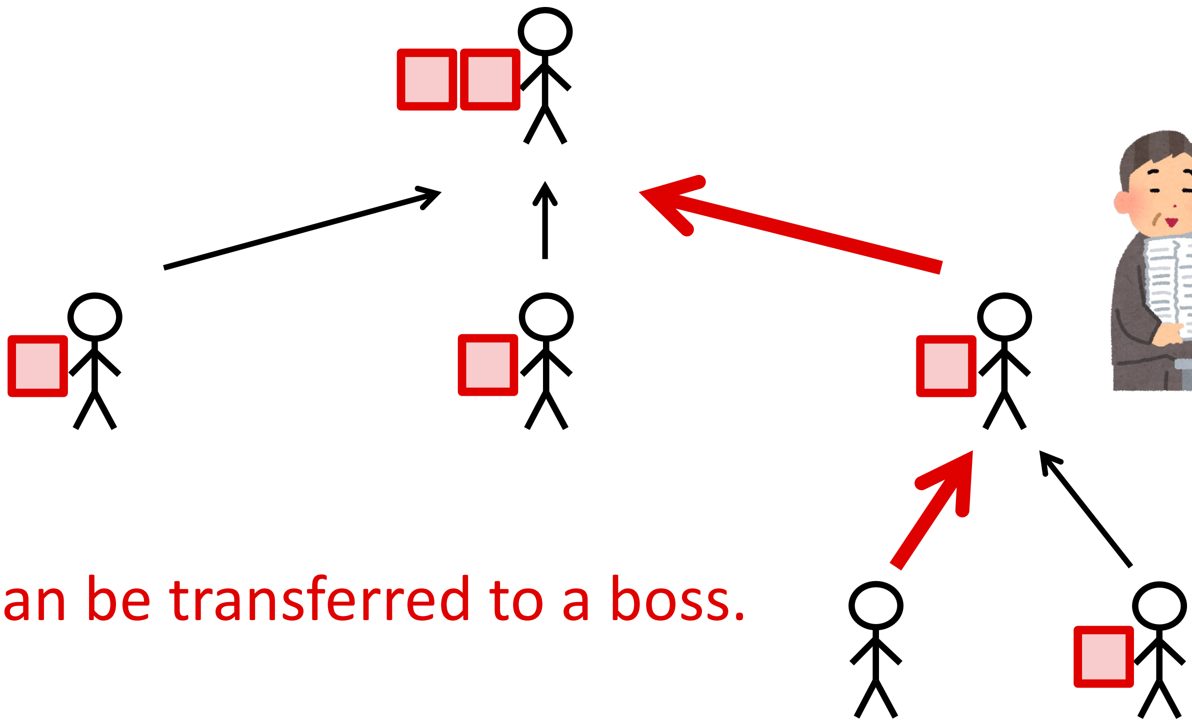
Complete **the tasks** with **the smallest total fatigue** of employees.



Every task can be transferred to a boss.

Story

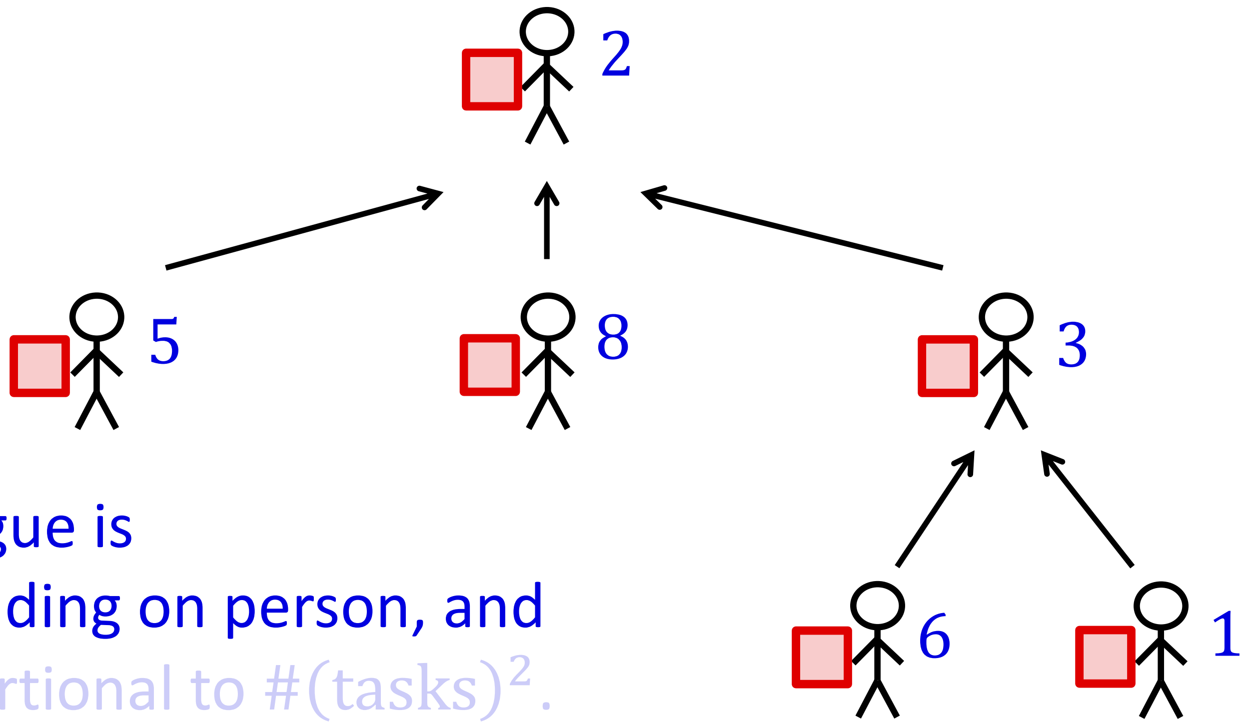
Complete **the tasks** with **the smallest total fatigue** of employees.



Every task can be transferred to a boss.

Story

Complete **the tasks** with **the smallest total fatigue** of employees.



The fatigue is

- depending on person, and
- proportional to $\#(\text{tasks})^2$.

Story

Complete **the tasks** with **the smallest total fatigue** of employees.

$2 \times 5^2 = 50$

$5 \times 0^2 = 0$ $8 \times 0^2 = 0$ $3 \times 1^2 = 3$

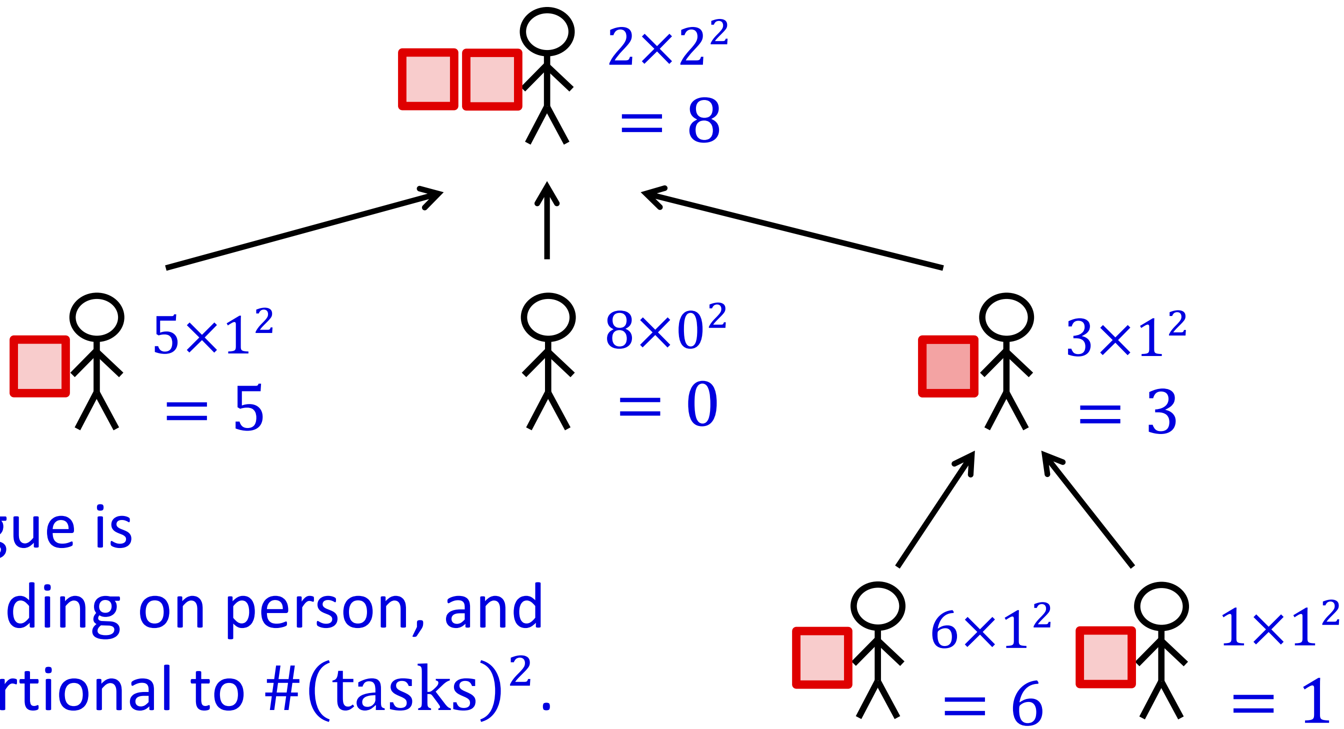
$6 \times 0^2 = 0$ $1 \times 0^2 = 0$

The fatigue is
- depending on person, and
- proportional to $\#(\text{tasks})^2$.

Story

Complete **the tasks** with **the smallest total fatigue** of employees.

$$8+5+0+3+6+1=23$$



The fatigue is

- depending on person, and
- proportional to $\#(\text{tasks})^2$.

Problem

Given a rooted tree of n vertices. $(2 \leq n \leq 5\times10^5)$

Given a fatigability constant f_i of each employee. $(1 \leq f_i \leq 10^{12})$

$$\boxed{\text{minimize } \sum_{i=1}^{n} f_i x_i^2 \text{, where } x_i = \#(\text{tasks done by } \#i)}$$

Solutions

$2 \le n \le 5 \times 10^5$

$1 \le f_i \le 10^{12}$

TL: 10 sec

[AC1] Greedy Algorithm with Heavy-Light Decomposition

- Min-weight base of a laminar matroid (Minimization of M-convex function)
- $O(n \cdot (\log n)^2)$ time

[AC2] Greedy + DP with Weighted-Union Heuristic

- $\mathrm{dp}(v) =$ opt. solution of the subtree of v (maintained by priority queue)
- $O(n \cdot \log n \cdot \log F)$ time $\left(F = \max_i f_i\right)$

[TLE] Naive DP on Tree

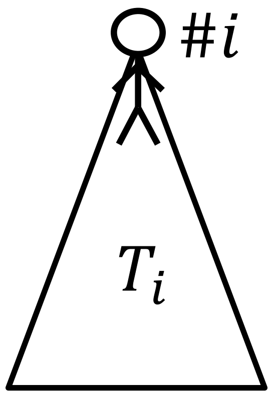
- $\mathrm{dp}(v, k) =$ opt. value of the subtree of v with k tasks completed
- $\Theta(n^2)$ time

Key Observations

$$\text{minimize } \sum_{i=1}^{n} f_i x_i^2 \text{, where } x_i = \#(\text{tasks done by } \#i)$$

- $(x_1, x_2, \dots, x_n)$ is feasible $\iff \sum_{j \in T_i} x_j \le |T_i| \ (\forall i)$,

  where T_i is the subtree of i.



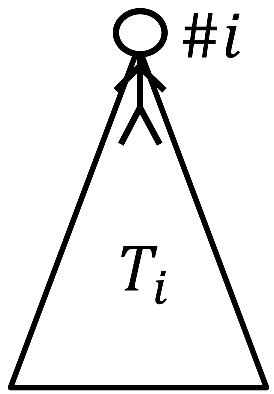
- $f_i x_i^2 = \sum_{k=1}^{x_i} (2k-1) f_i$

  $\rightarrow$ the k-th task takes cost $(2k-1)f_i$

Key Observations

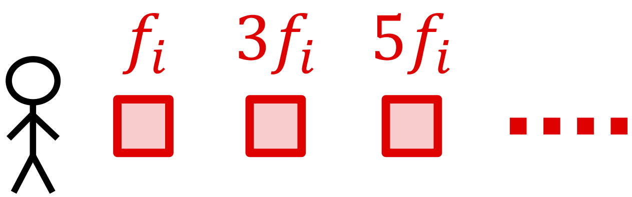
$$\text{minimize } \sum_{i=1}^{n} f_i x_i^2 \text{ , where } x_i = \#(\text{tasks done by } \#i)$$

- $(x_1, x_2, \ldots, x_n)$ is feasible $\Leftrightarrow$ $\sum_{j \in T_i} x_j \leq |T_i|$ $(\forall i)$, where T_i is the subtree of i.



- $f_i x_i^2 = \sum_{k=1}^{x_i} (2k-1)\, f_i$

  $\rightarrow$ the k-th task takes cost $(2k-1) f_i$



Reformulation

$$\text{minimize } \sum_{i=1}^{n} f_i x_i^2 \text{, where } x_i = \#(\text{tasks done by } \#i)$$

- Each employee $\#i$ has n items with cost $f_i, 3f_i, \ldots, (2n-1)f_i$.
- Minimize the total cost by selecting exactly n items in total subject to at most $|T_i|$ items are selected in each subtree T_i.

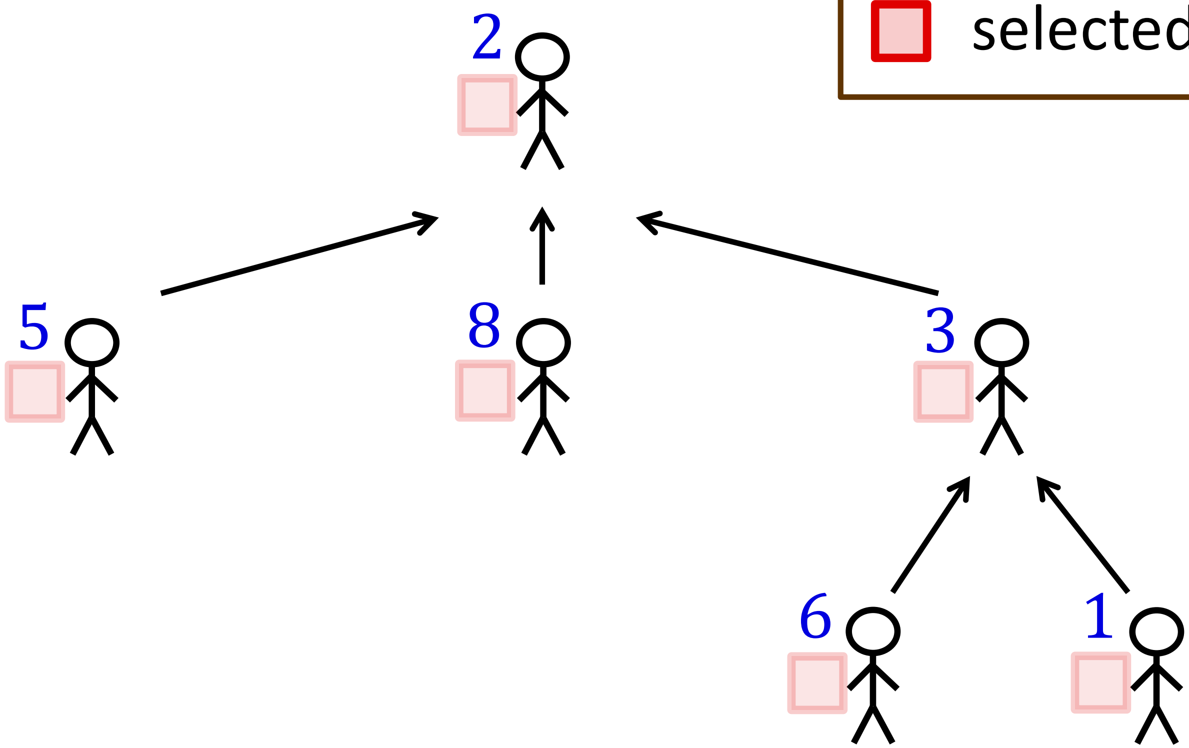
Minimum Weight Base of a Laminar Matroid

$\rightarrow$ Greedy is Optimal

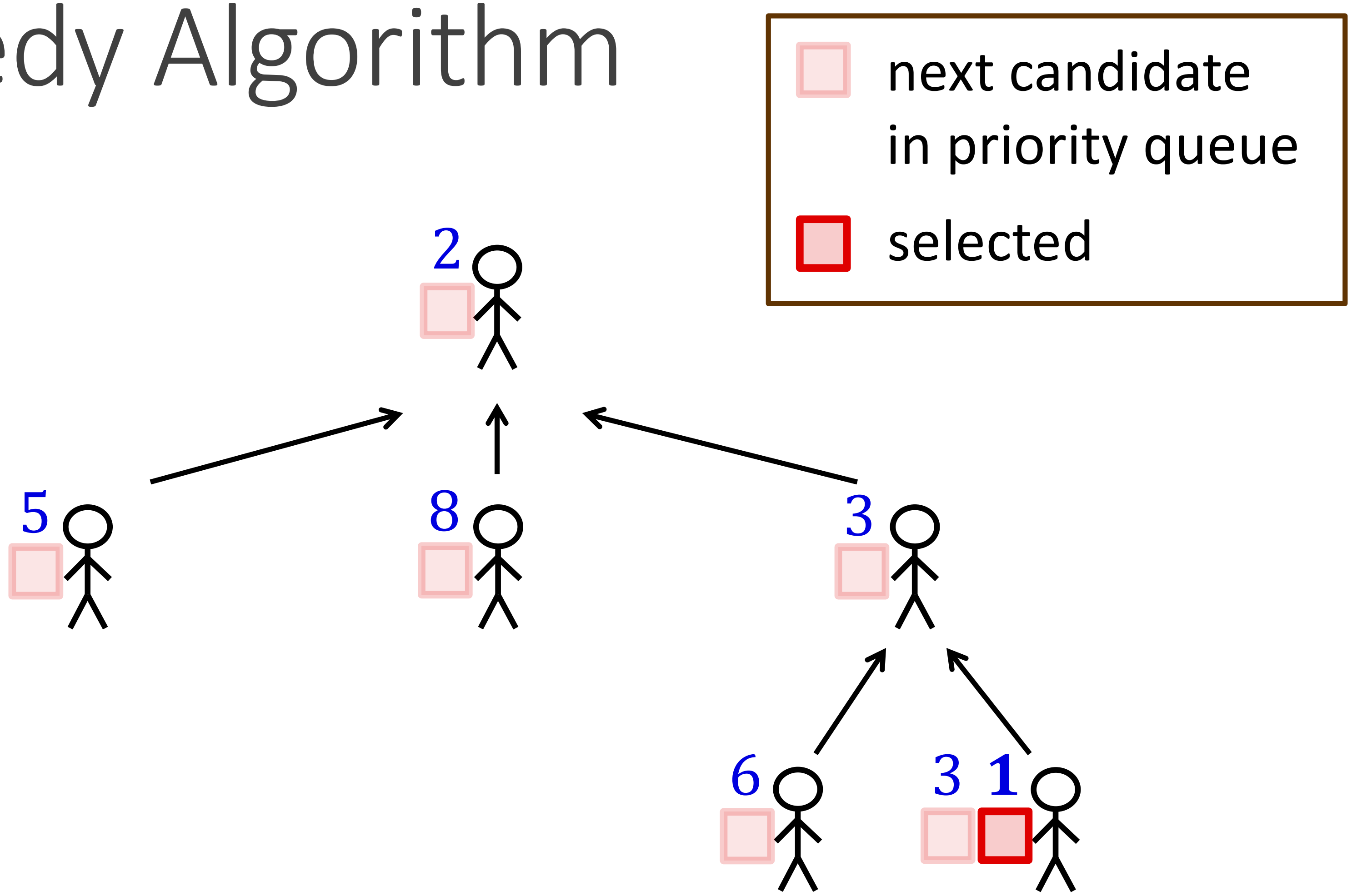
Greedy Algorithm

next candidate in priority queue

selected



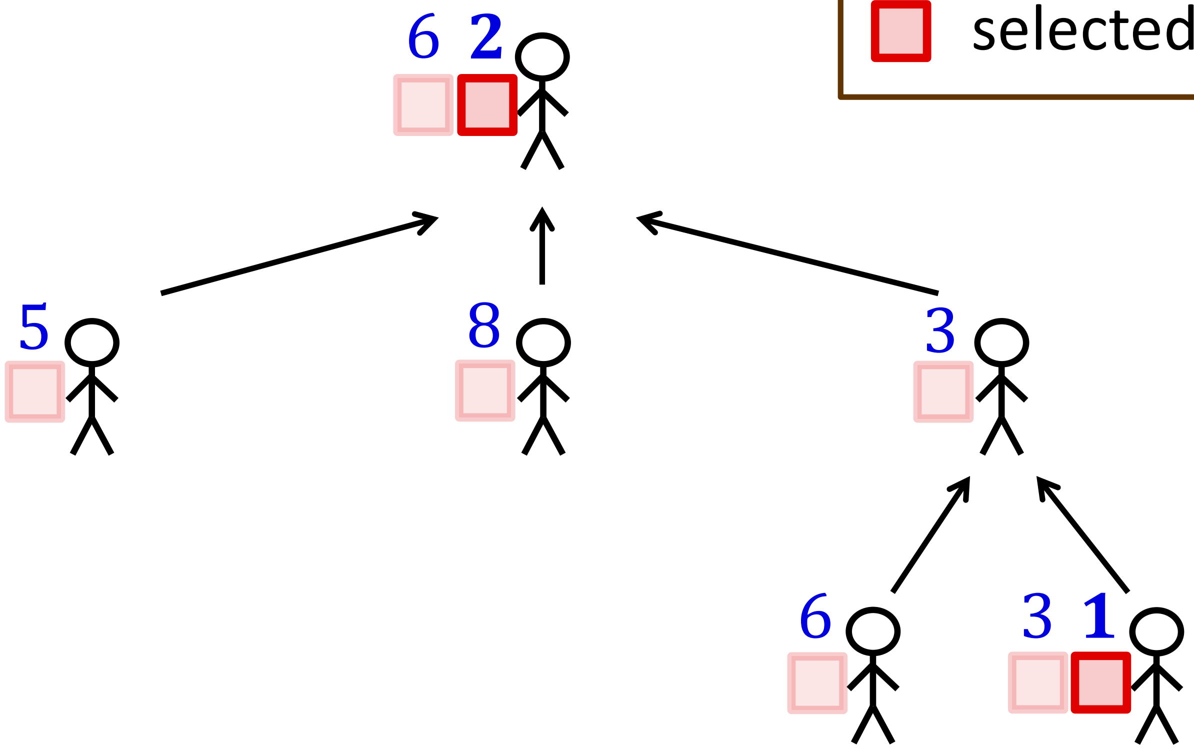
Greedy Algorithm



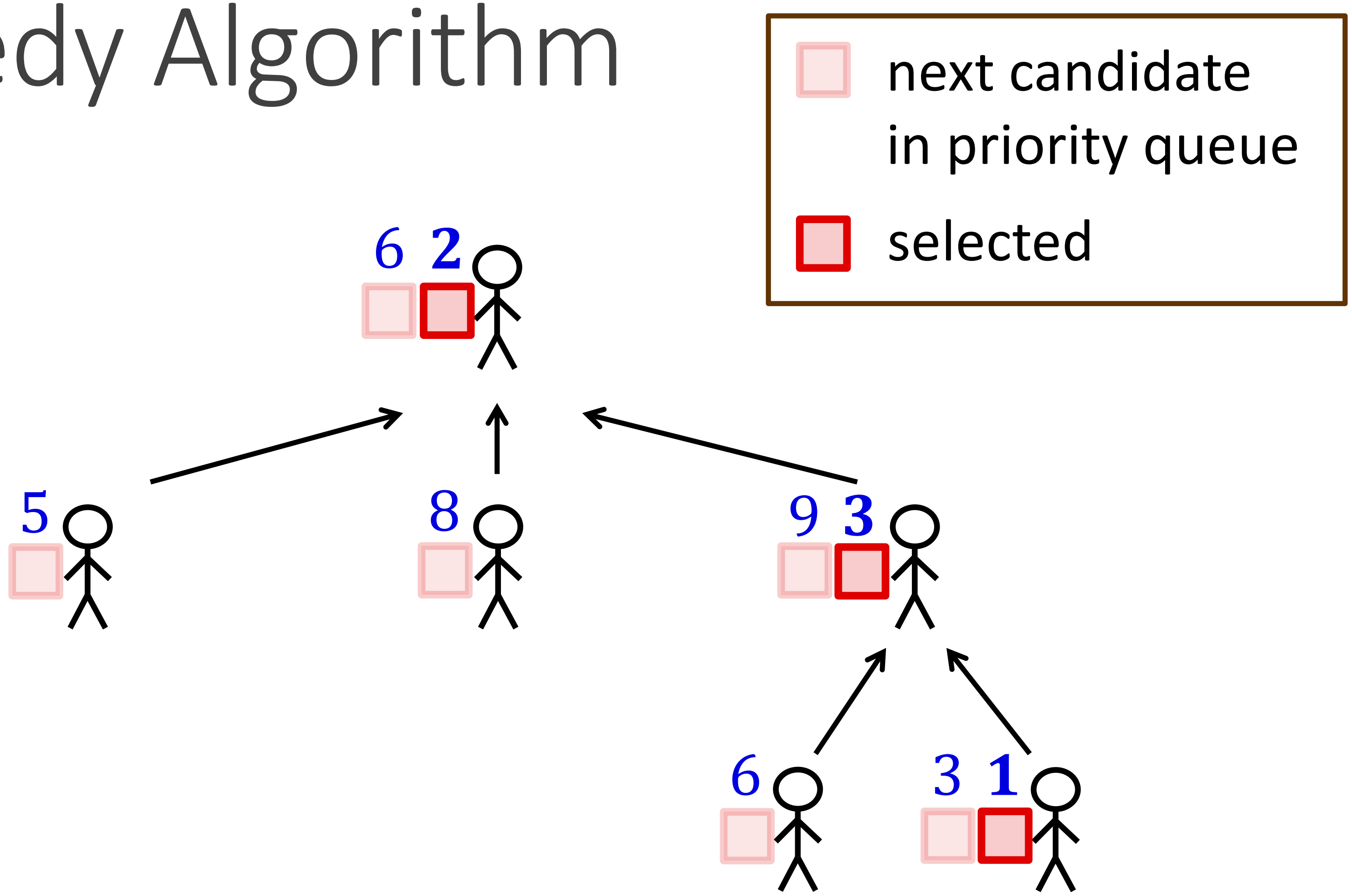
Greedy Algorithm

next candidate in priority queue

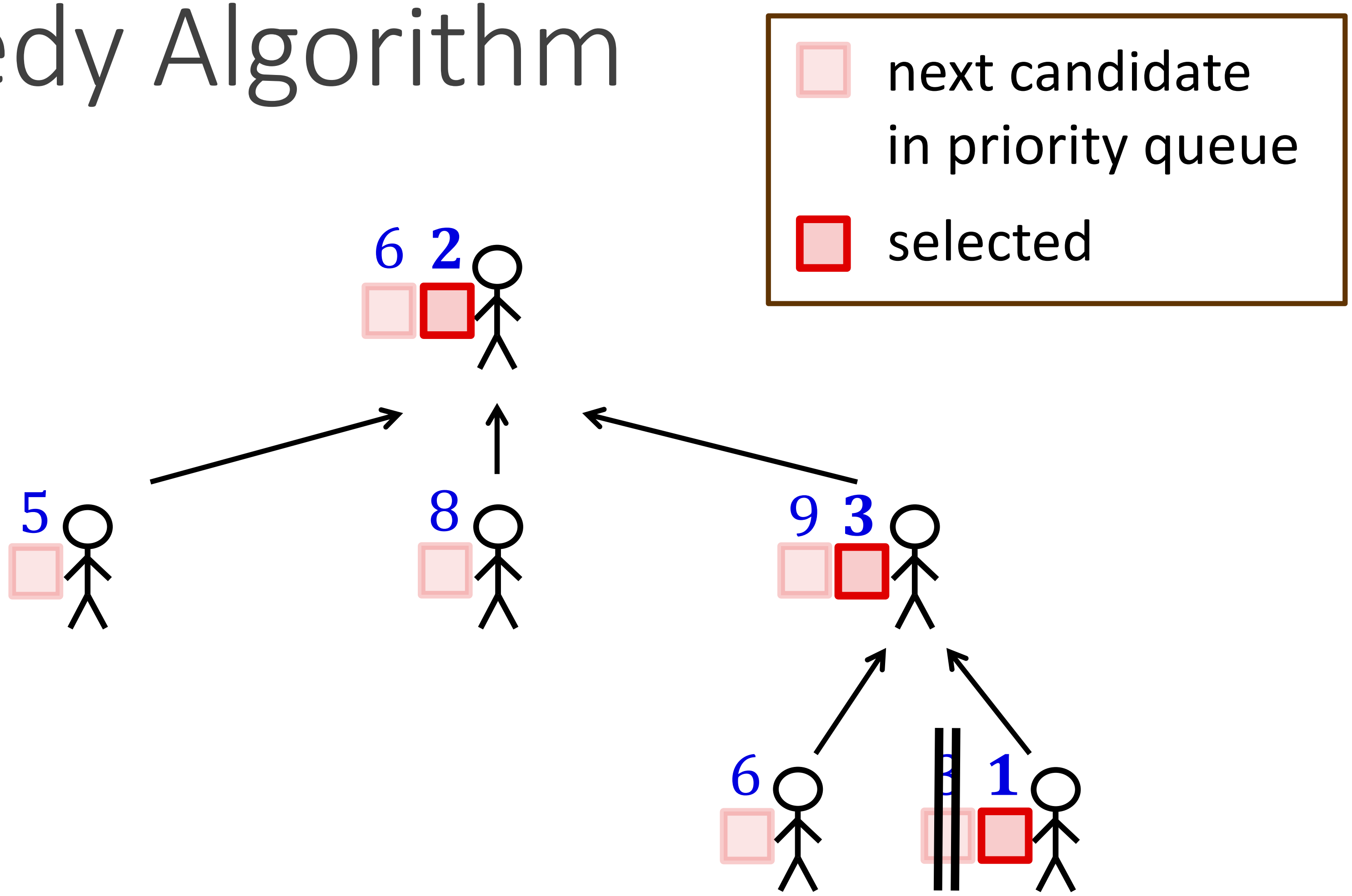
selected



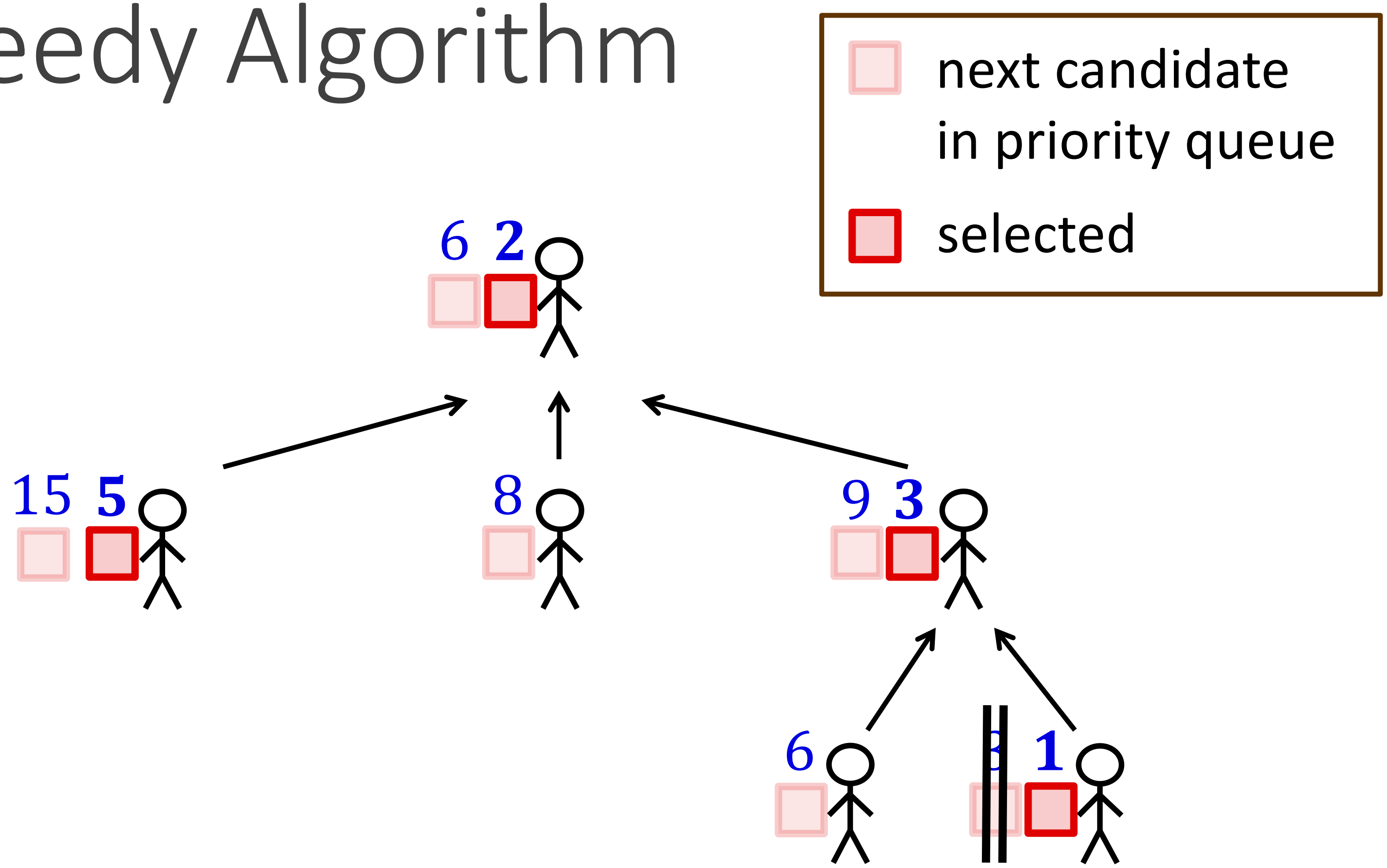
Greedy Algorithm



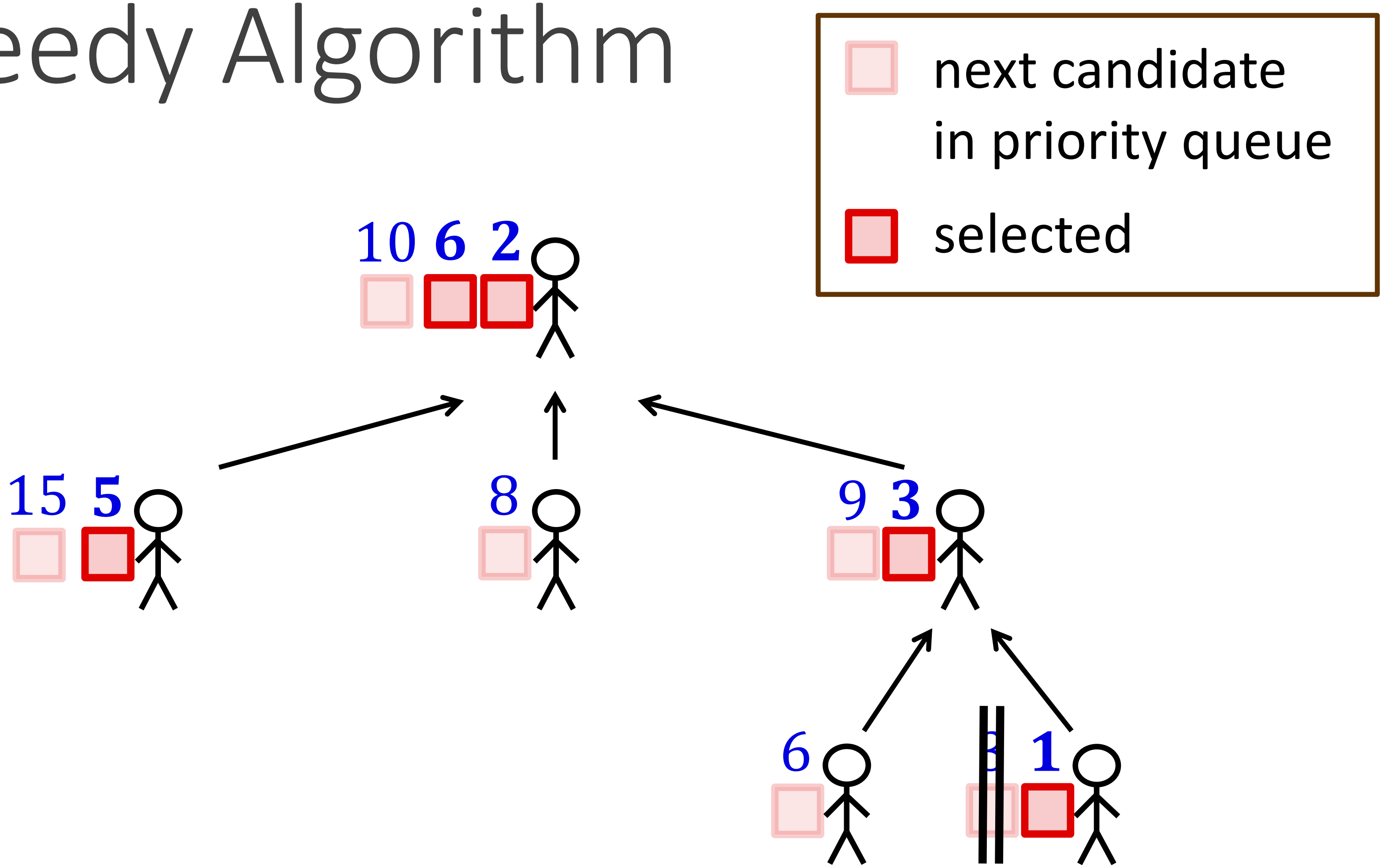
Greedy Algorithm



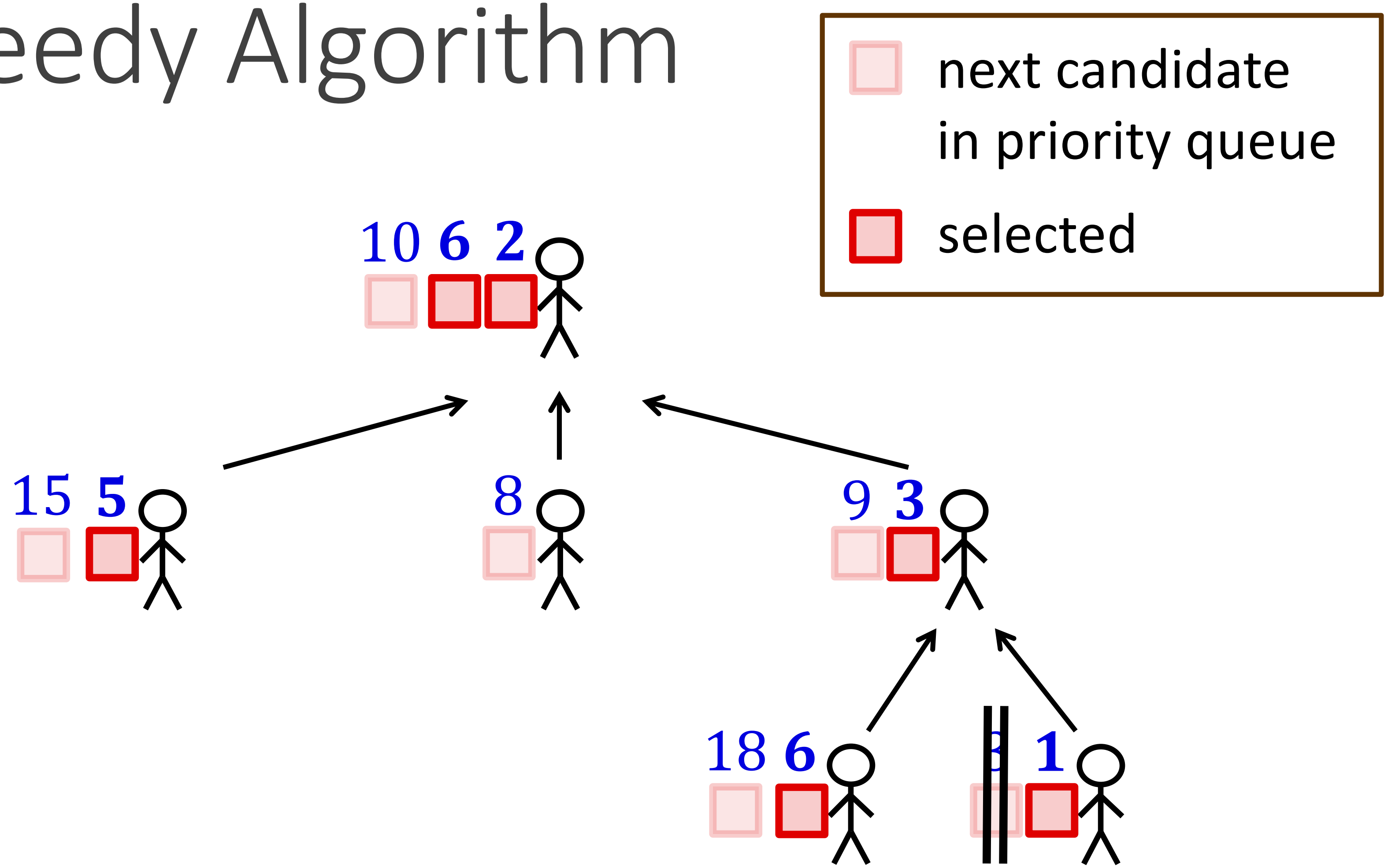
Greedy Algorithm



Greedy Algorithm



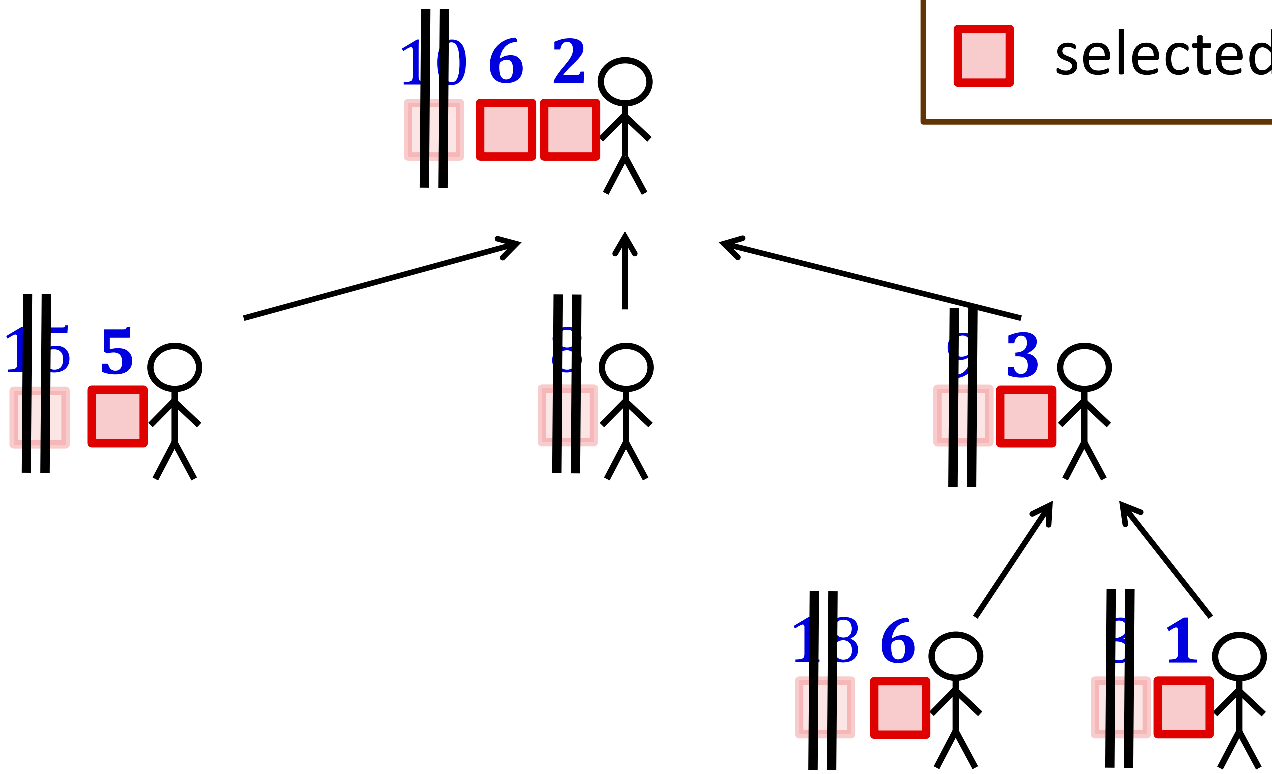
Greedy Algorithm



Greedy Algorithm

next candidate in priority queue

selected



Greedy with HL Decomposition

- An item can be selected

  $\Longleftrightarrow$ The subtree of every boss $\#i$ has positive capacity, i.e.,

  $$\mathrm{cap}(i) \coloneqq |T_i| - \#(\text{items selected in } T_i) > 0$$

- An item is selected $\rightarrow$ Decrease $\mathrm{cap}(i)$ by 1 for every boss $\#i$

- An item is not selected $\rightarrow$ The same person will never work

Range Minimum + **Range Add** $2n$ times

$\mathrm{O}(n \cdot (\log n)^2)$ time with **Heavy-Light Decomposition**

$2 \le n \le 5 \times 10^5$
$1 \le f_i \le 10^{12}$
TL: 10 sec

Solutions

[AC1] Greedy Algorithm with Heavy-Light Decomposition

[AC2] Greedy + DP with Weighted-Union Heuristic

- $\mathrm{dp}(v) =$ opt. solution of the subtree of v (maintained by priority queue)
- $\mathrm{O}(n \cdot \log n \cdot \log F)$ time $\left(F = \max_i f_i\right)$
  - Merge is completed in $\mathrm{O}(n \cdot \log n)$ time (meldable heap) in total;
    $\mathrm{O}(n \cdot (\log n)^2)$ time (usual heap) is also enough.
  - $\#(\text{insertion}) = \mathrm{O}(n \cdot \log F)$ is proved by considering a potential function
    $$\Phi(v) := \sum_{x \in \mathrm{dp}(v)} \log x\,.$$

[TLE] Naive DP on Tree

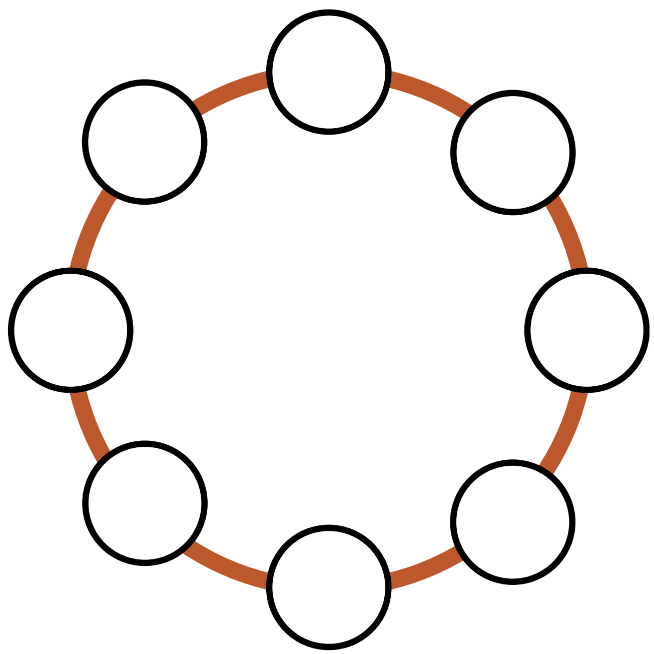
C: Ferris Wheel

PROPOSER: SOH KUMABE

AUTHOR: SOH KUMABE

Problem Description

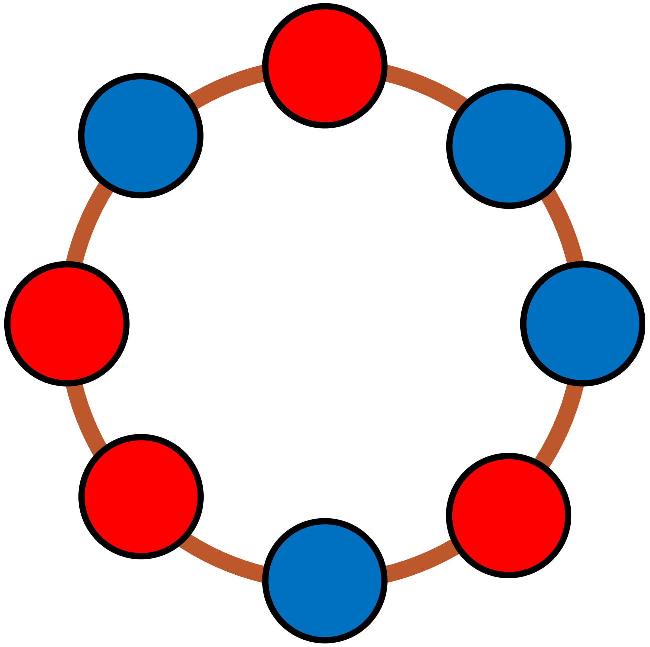
Given $2n$ **points on circle,**



Problem Description

Given $2n$ points on circle,

Count the number of ways to color them by k colors so that



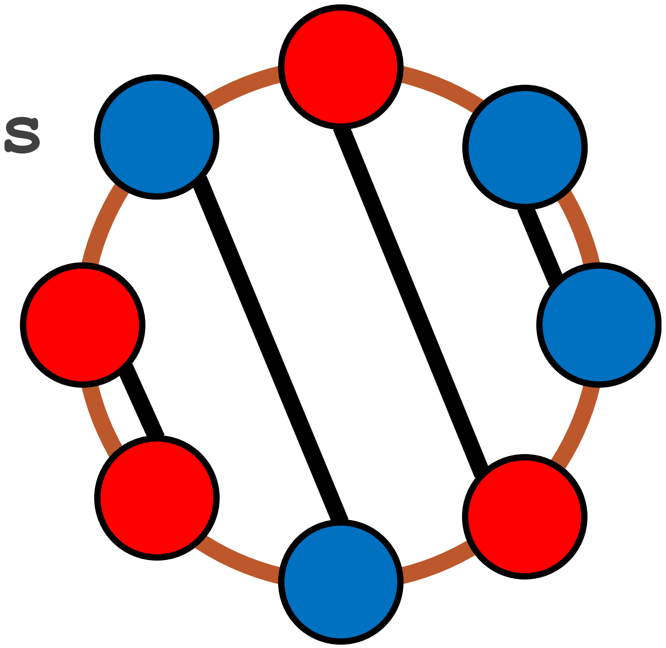
Problem Description

Given $2n$ points on circle,

Count the number of ways to color them by k colors so that

There is a non-crossing perfect matching of points

Such that matched points have the same color



Problem Description

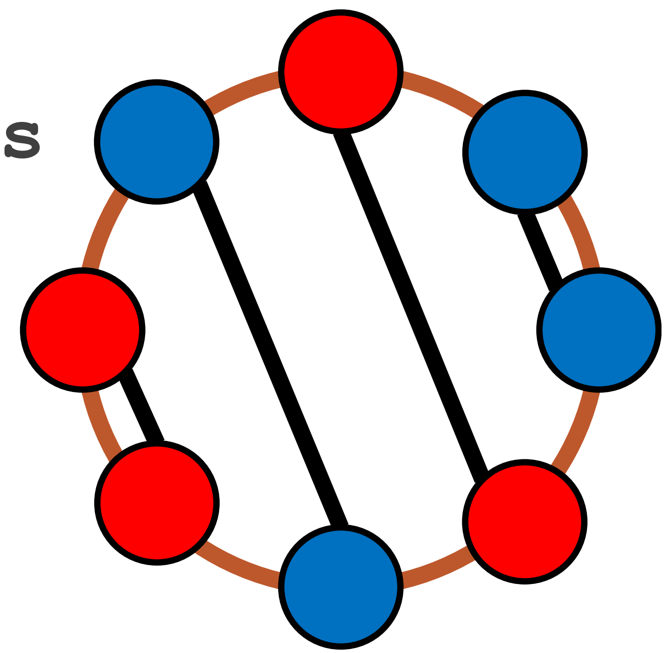
Given $2n$ points on circle,

Count the number of ways to color them by k colors so that

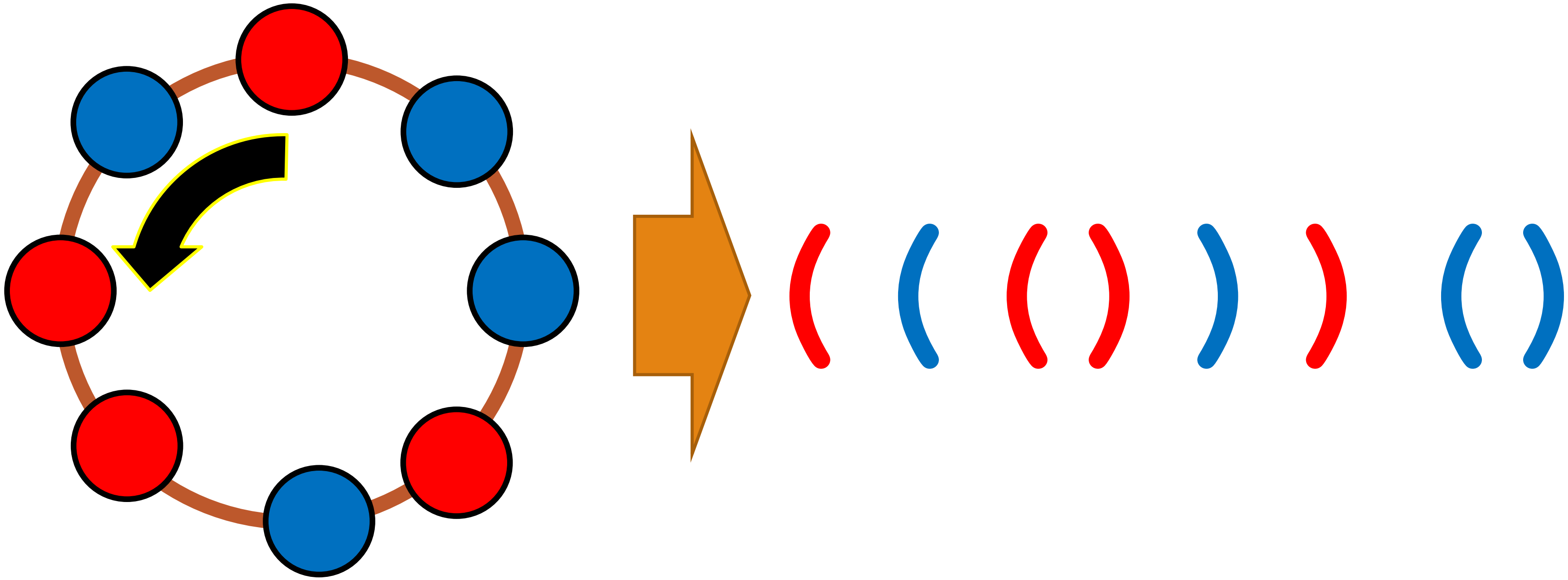
There is a non-crossing perfect matching of points

Such that matched points have the same color

Up to rotation



Matching to Parenthesis

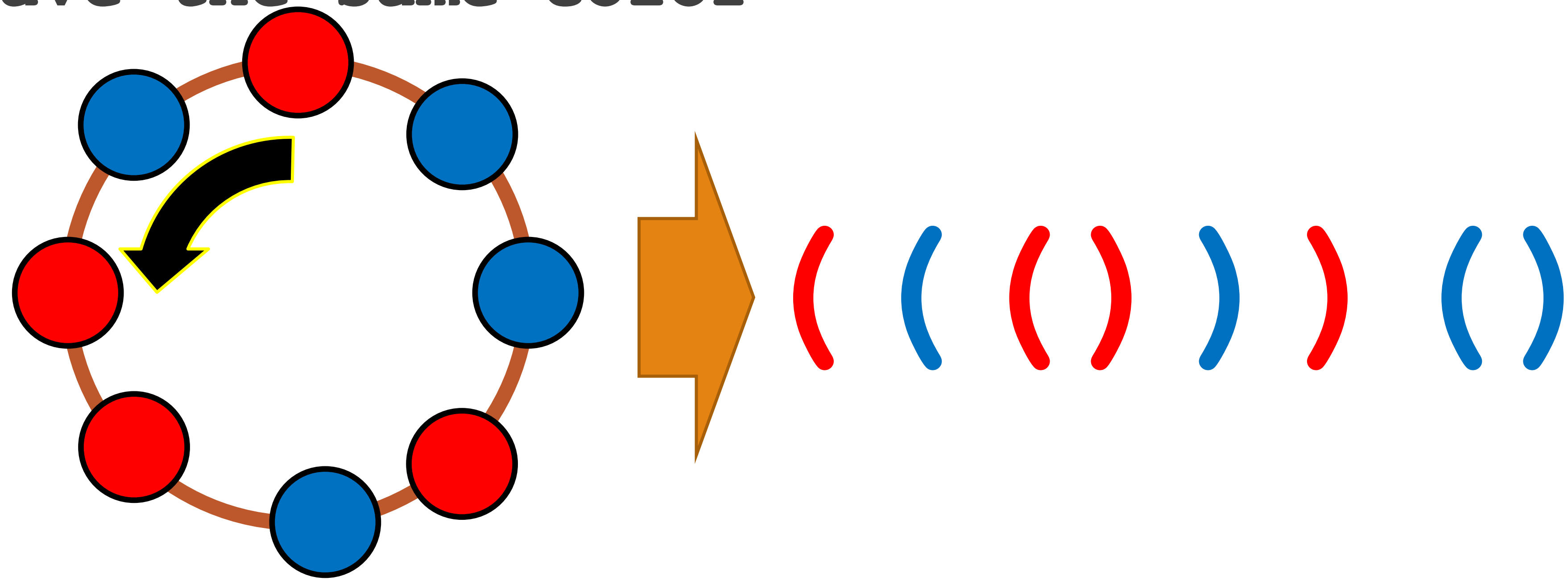


Matching to Parenthesis

There is a non-crossing perfect matching of points

Such that matched points have the same color

Parenthesis is balanced



If not "up to rotation"

Let x_i be the number of balanced parenthesis that have i places with height 0

0 1 2 3 2 1 0 1 0

((( ))) ( )

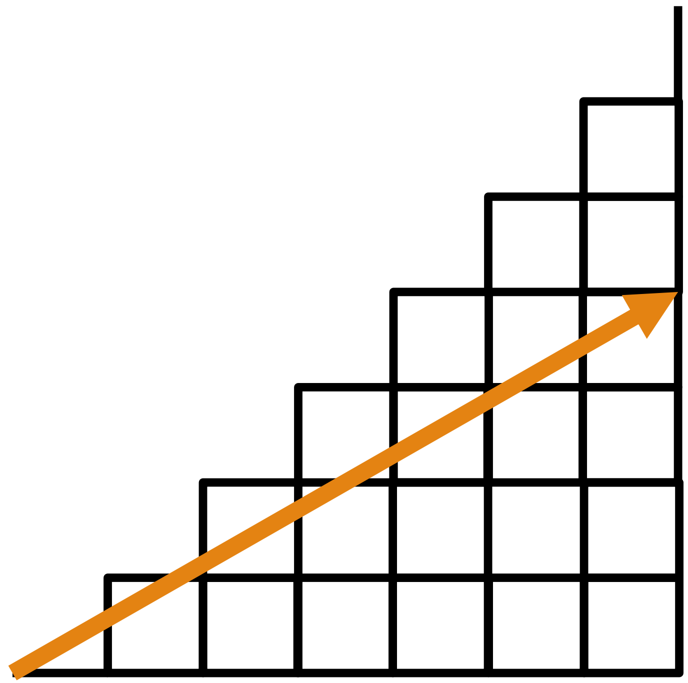
Answer is $\sum_{i=1}^{n} x_i k^i (k-1)^{n-i}$

If not "up to rotation"

Let x_i be the number of balanced parenthesis that have i places with height 0

Can be computed like Catalan numbers

Time Complexity: $O(n)$



“Up to rotation”

Use **Pólya’s enumeration theorem**

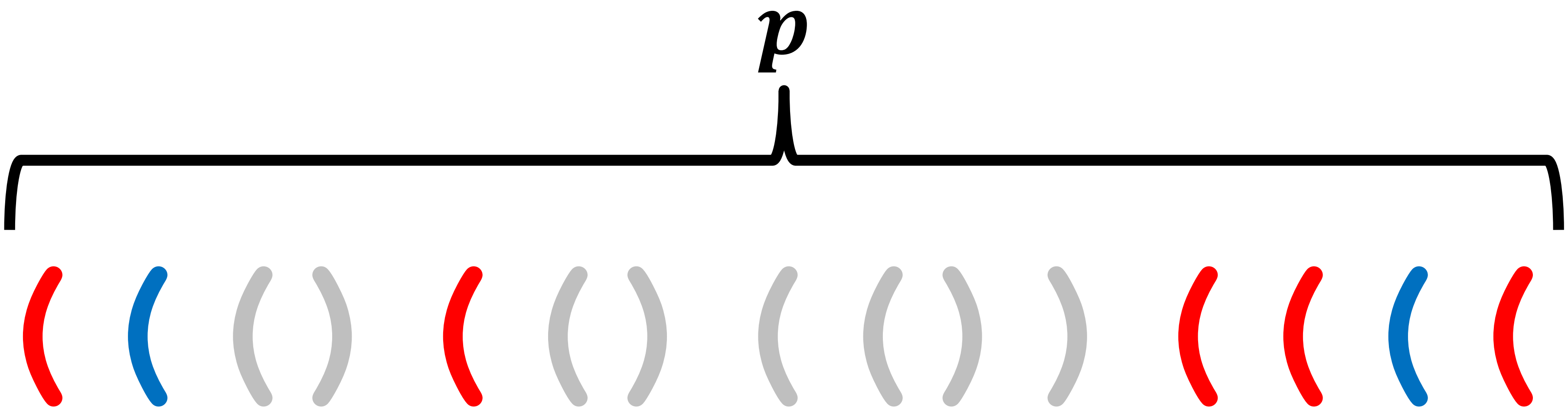
Count **the colorings of period** p

“Up to rotation”

There is a non-crossing perfect matching of points

Such that matched points have the same color

Remaining ‘(‘s are palindrome



“Up to rotation”

If p **is even, no remaining `(`**

same as before

If p **is odd,**
remaining `(`s are palindrome

“Up to rotation”

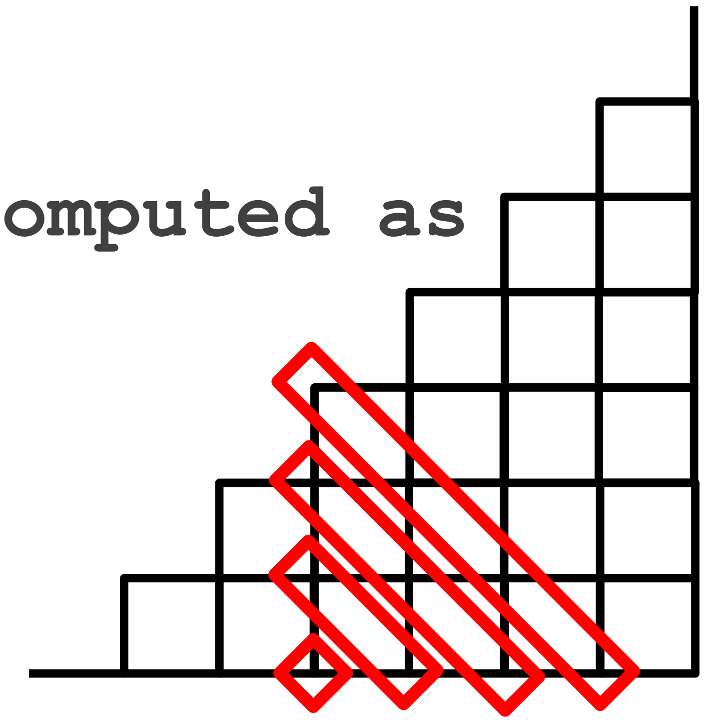
If p is odd,
remaining `(`s are palindrome

Let x_i be the number of parenthesis that have i places with height 0 and some number of `(`s remain

Answer is $\sum_{i=1}^{\frac{p+1}{2}} x_i k^i (k-1)^{\frac{p+1}{2}-i}$

"Up to rotation"

Let x_i be the number of parenthesis that have i places with height 0 and some number of \`(\`s remain



Can be sequentially computed as "diagonal sum" of Catalan number

Time Complexity:
$O(\text{sum of divisors of } 2n)$
$= O(n \log n)$