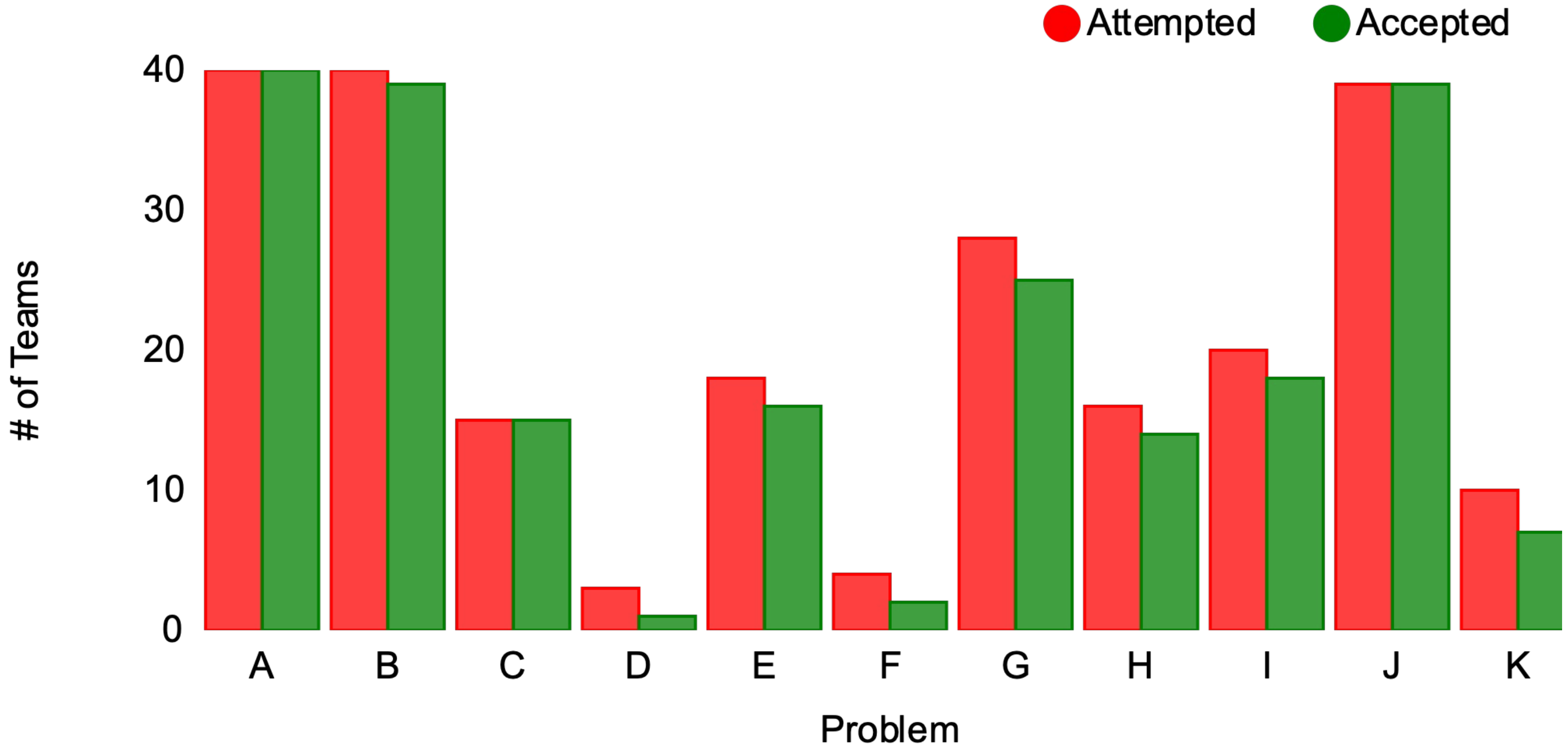
Commentaries on Problems

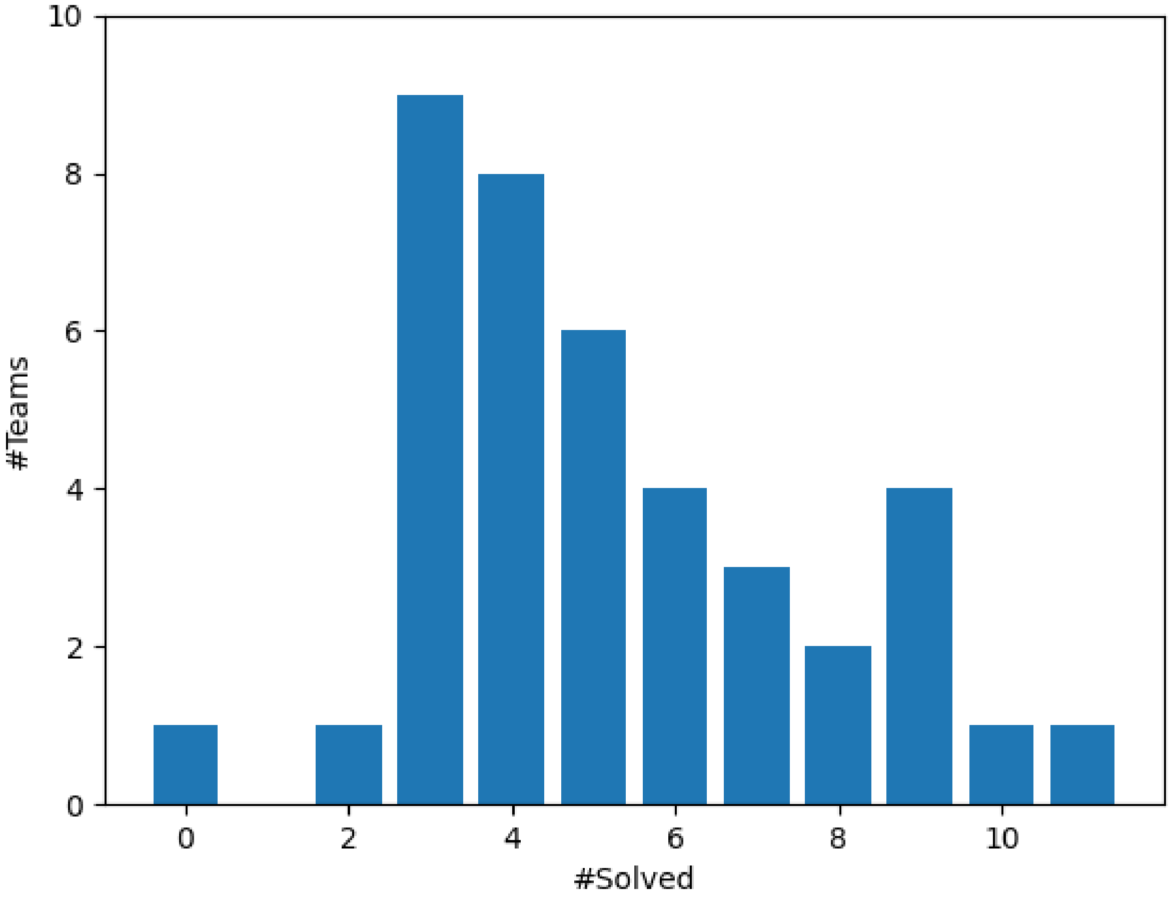
JUDGE TEAM

ICPC 2020 ASIA YOKOHAMA REGIONAL

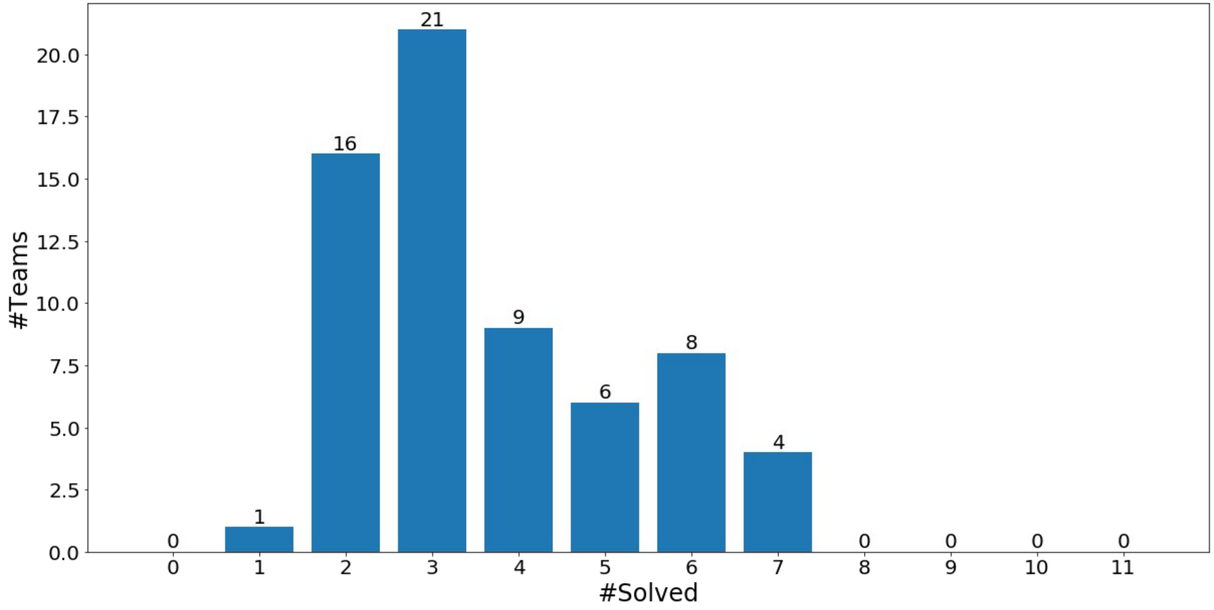
Problem vs. #Teams @Freeze

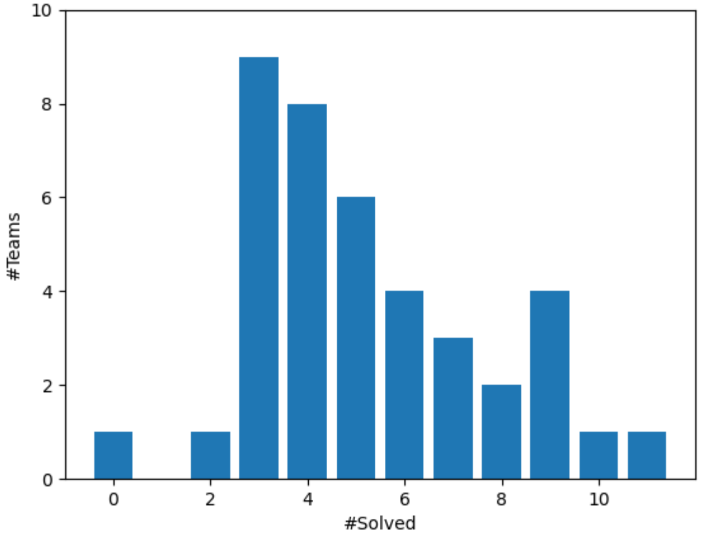


#Solved vs #Teams @Freeze

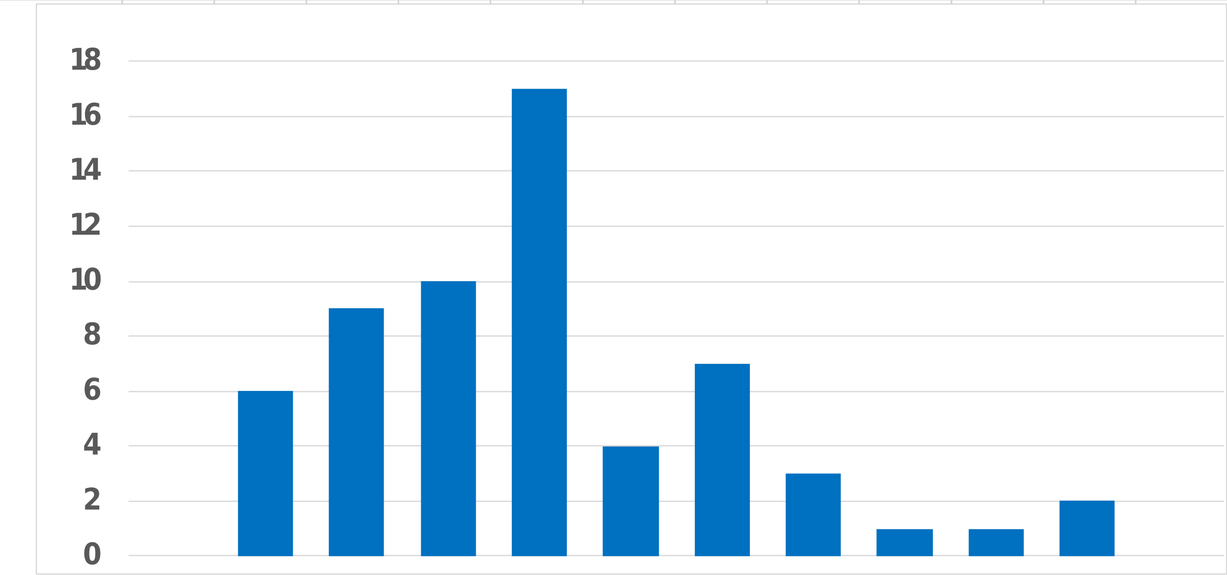


cf. 2019's #Solved vs #Teams

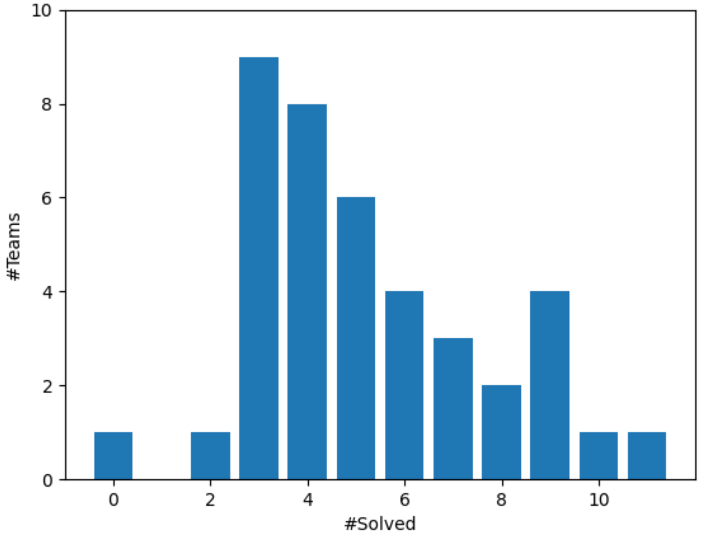




cf. 2018’s #Solved vs #Teams



#Solved	0	1	2	3	4	5	6	7	8	9	10	11
#Teams	0	6	9	10	17	4	7	3	1	1	2	0



Commentaries

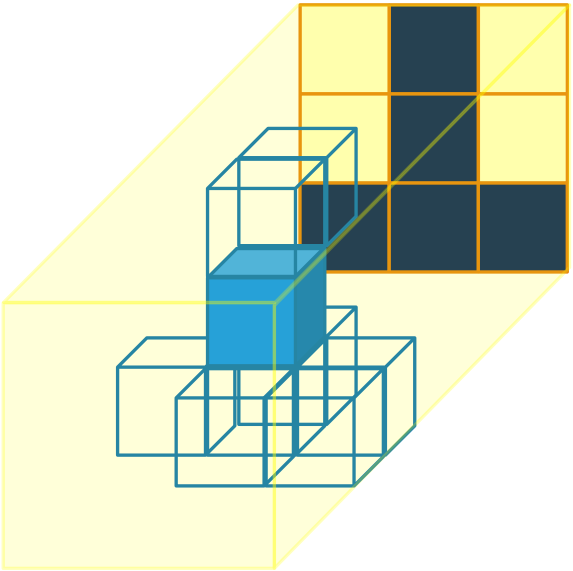
Decreasing order of #attempt teams
(tie-breaking by alphabetical order)

A □ B □ J □ G □ I □ E □ H □ C □ K □ F □ D

A:Three-Axis Views

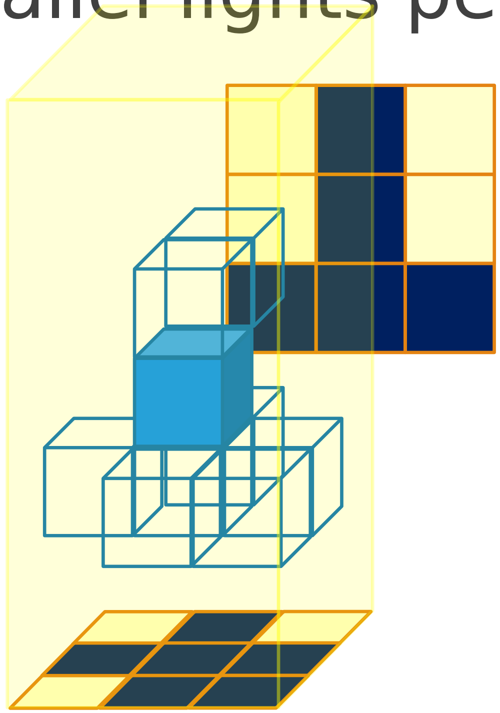
Story

An object, a subset of cubes, can make three silhouettes of squares by three parallel lights perpendicular to its faces



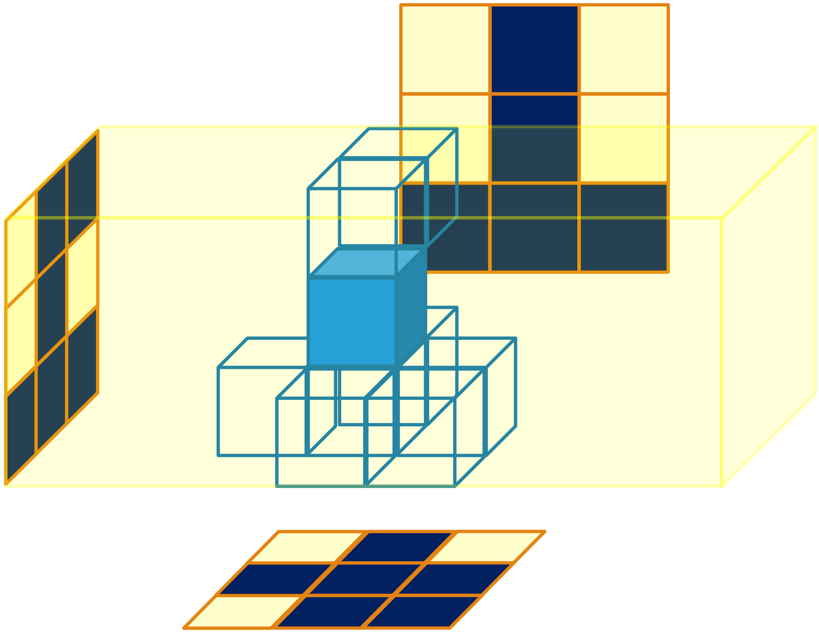
Story

An object, a subset of cubes, can make three silhouettes of squares by three parallel lights perpendicular to its faces



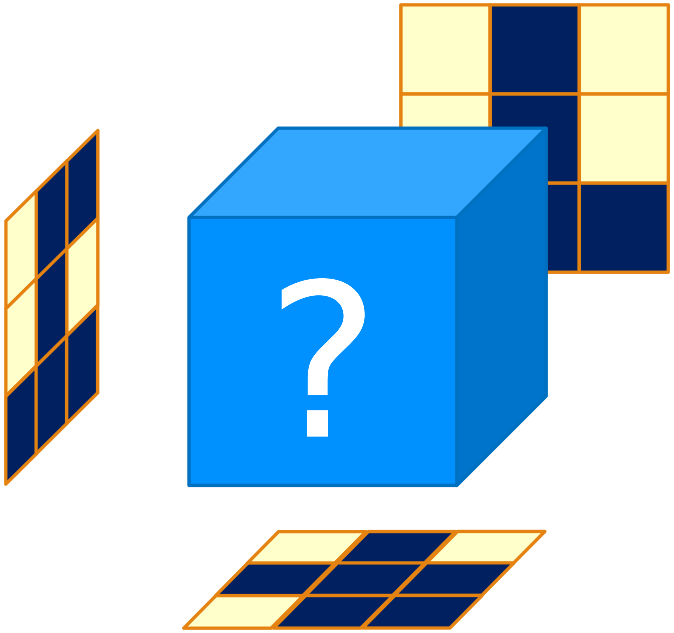
Story

An object, a subset of cubes, can make three silhouettes of squares by three parallel lights perpendicular to its faces



Problem

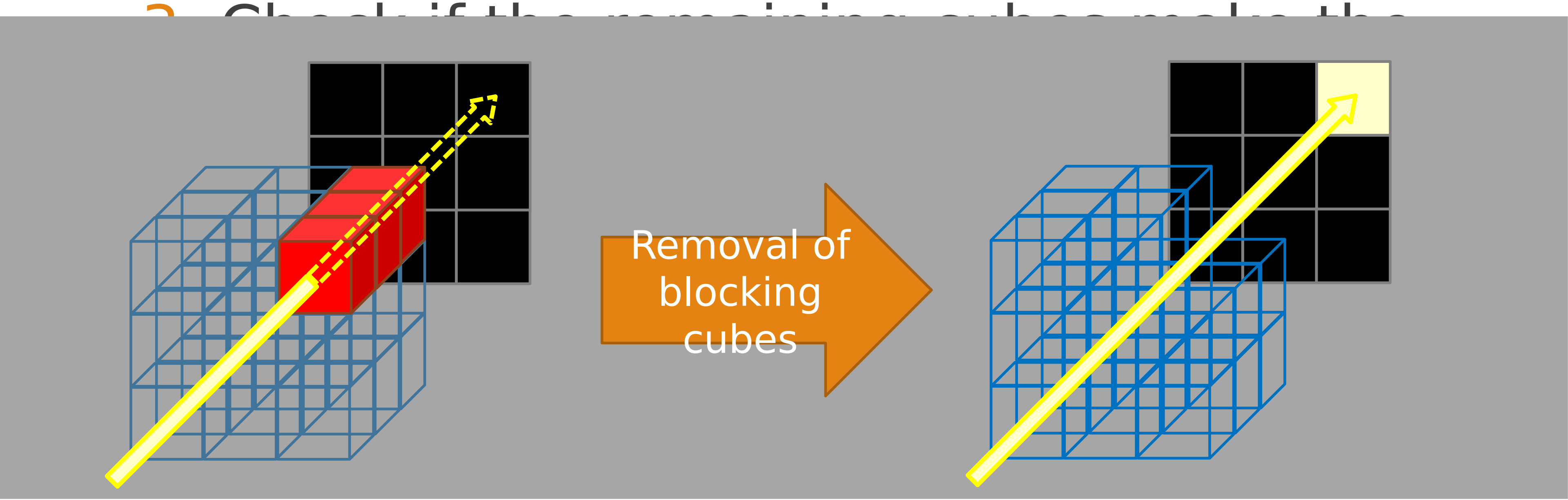
An object, a subset of cubes, can make three silhouettes of squares by three parallel lights perpendicular to its faces



Can you make such an object
to make the given silhouettes?

Solution ( )

1. Begin with the full cube
2. Remove cubes blocking the light to make a blight cell
3. Check if the remaining cubes make the



B: Secrets of Legendary Treasure

Problem (1/2)

Numbers from 1 to X were partitioned into two lists, each sorted in ascending order, but... ....

4	5	7	10	11	13

1	2	3	6	8	9	12

Problem (2/2)

Numbers from 1 to X were partitioned into two lists, each sorted in ascending order, but ……………………

	5				13

		3		8		12

some of the numbers were lost!

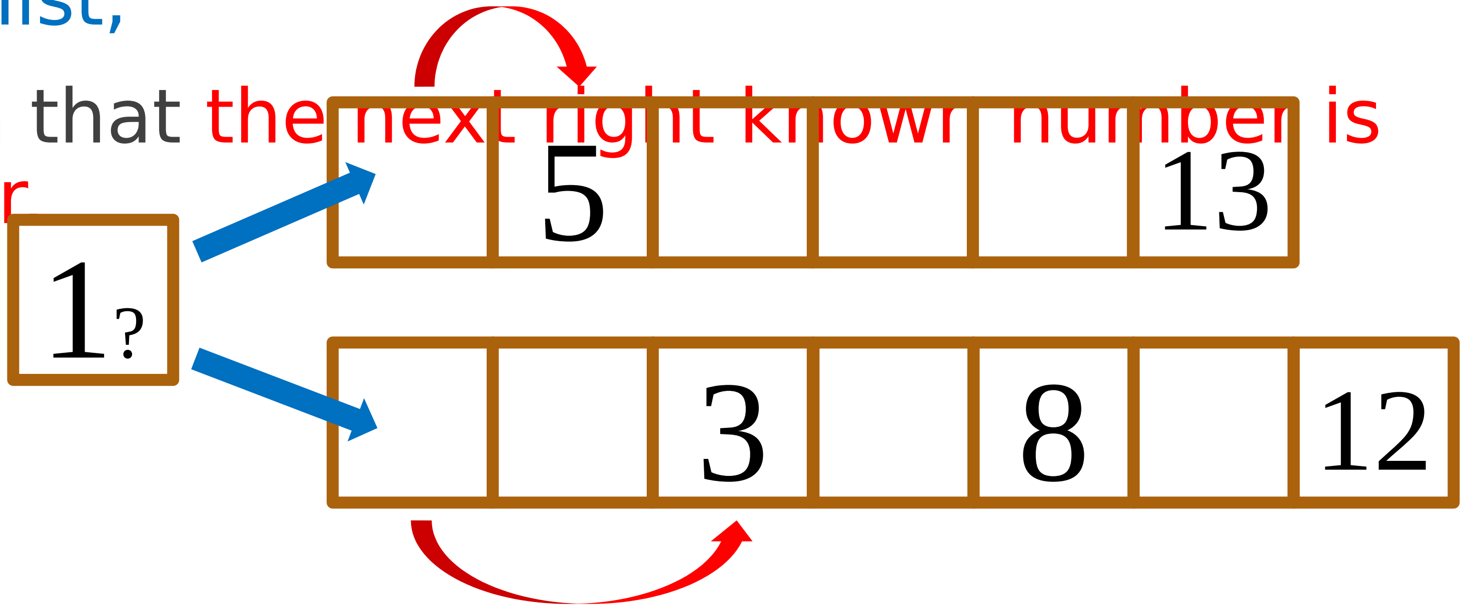
Please restore the original pair of lists.

(Any one of them is accepted when there are multiple possibilities.)

Solution (Greedy algorithm)

Repeat finding where to put the least unused number:

- The leftmost unknown pos of the either of the list,
- such that the next right known number is smaller



Solution

[Proof outline of the greedy method] show that “if there’s a solution that fills y (y>x) to the greedy choice position, then filling the least value x also leads to another solution.” Rotating {x, x+1, …, y} in the former solution gives you the latter.

[Other approaches]

- Memoized search
- BFS-like search
  - State (a,b) reachable partitioning {1..a+b} into lists of length a and b is possible without contradiction

J: Formica Sokobanica

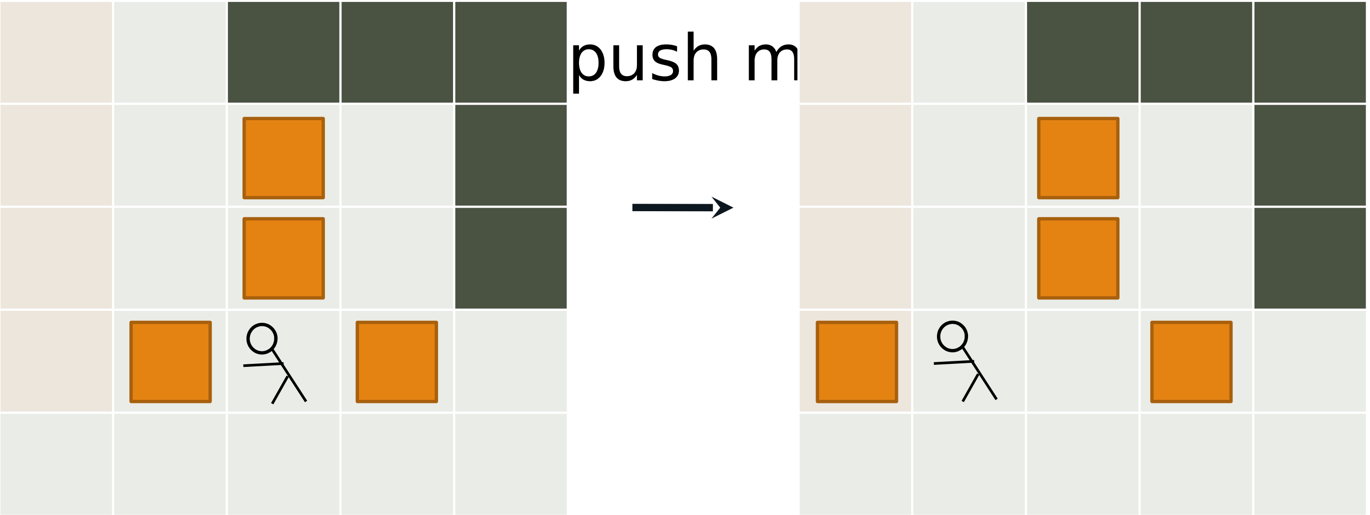
Formica Sokobanica

Formica Sokobanica is named after a computer game.

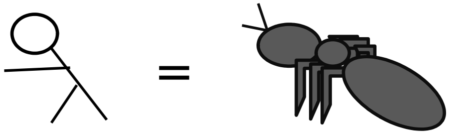
A worker arranges boxes in a warehouse by pushing them.

Tl push m

si

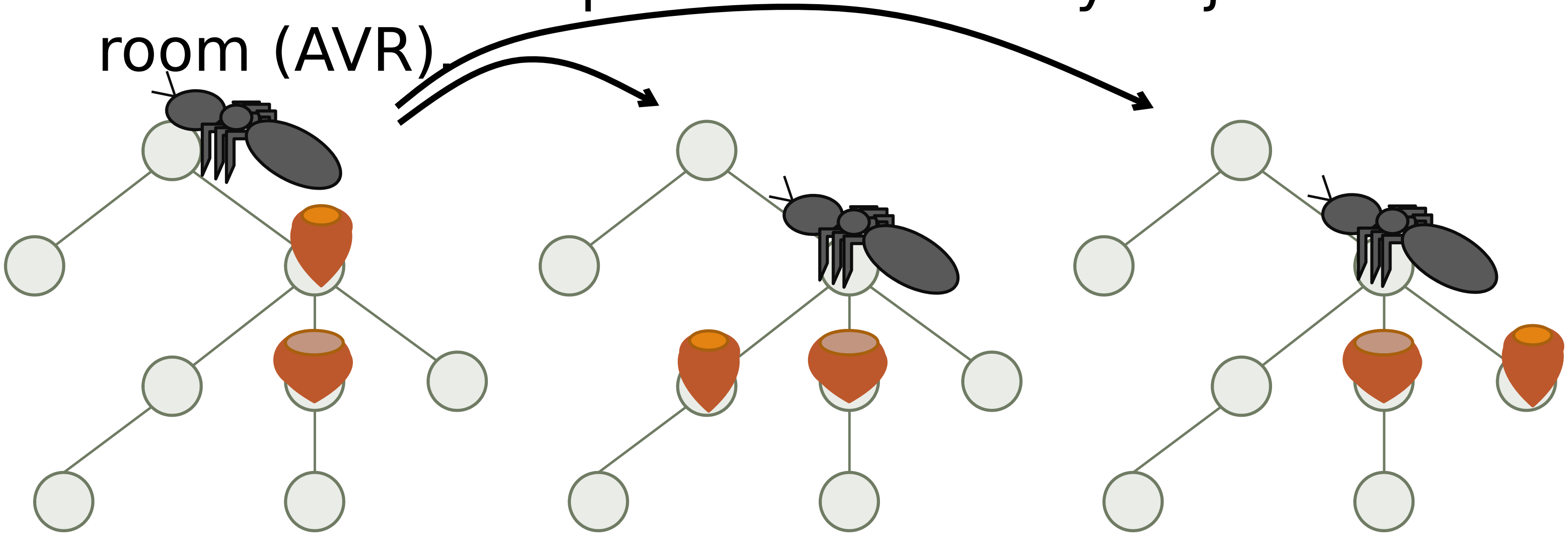


Problem Description



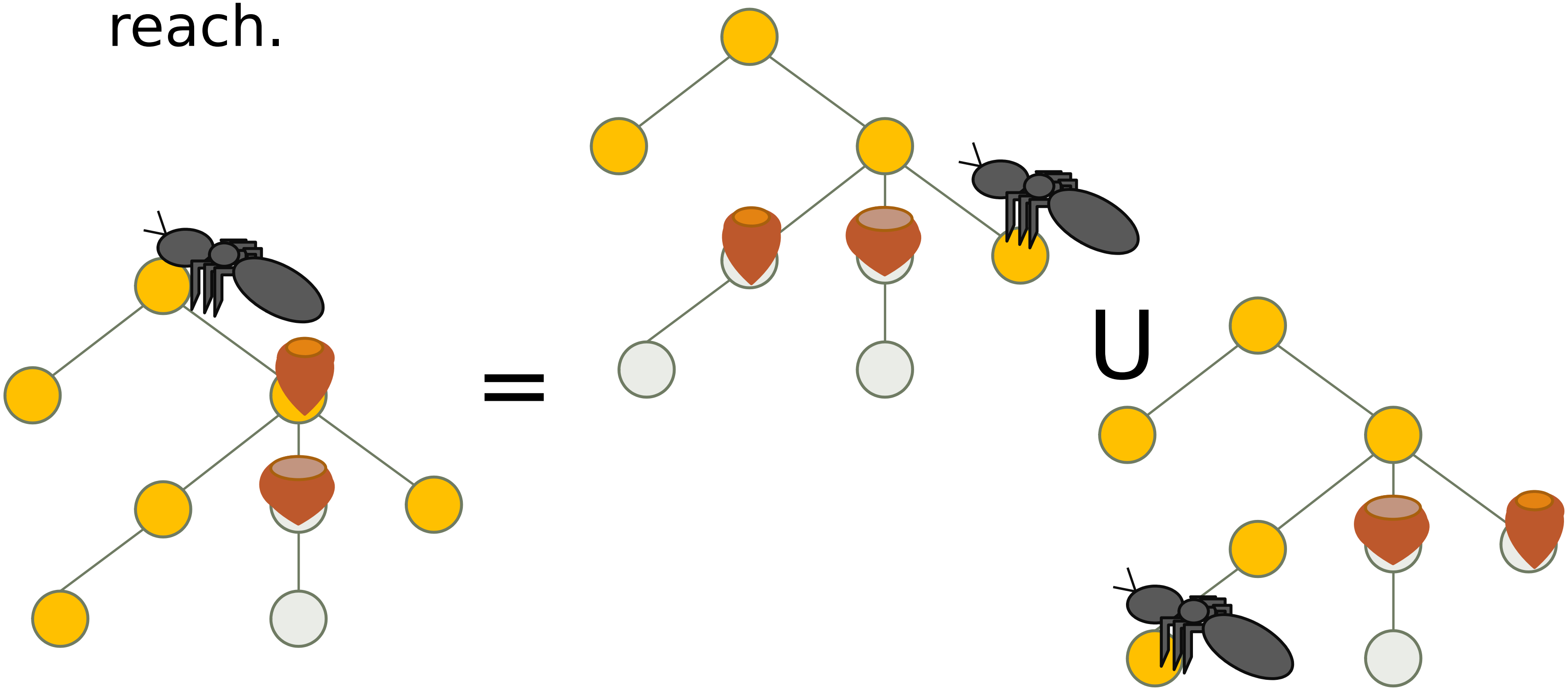
A variant of the computer game.

- The worker is an ant.
- Topology of the field is a tree.
- The worker can push a nut to any adjacent vacant room (AVR).



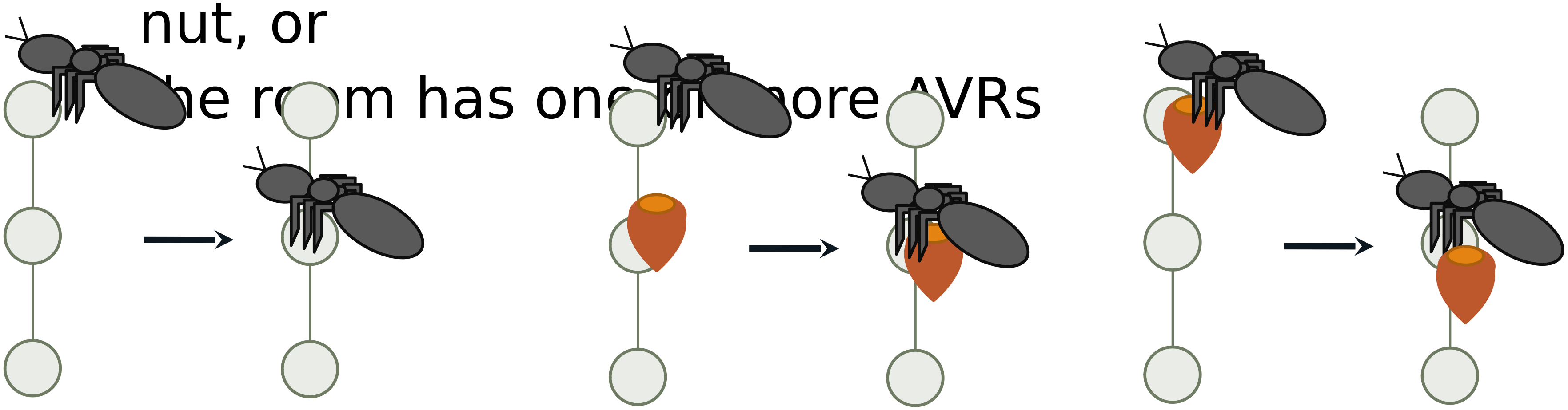
Problem Description: objective

Count up the number of rooms the ant can reach.



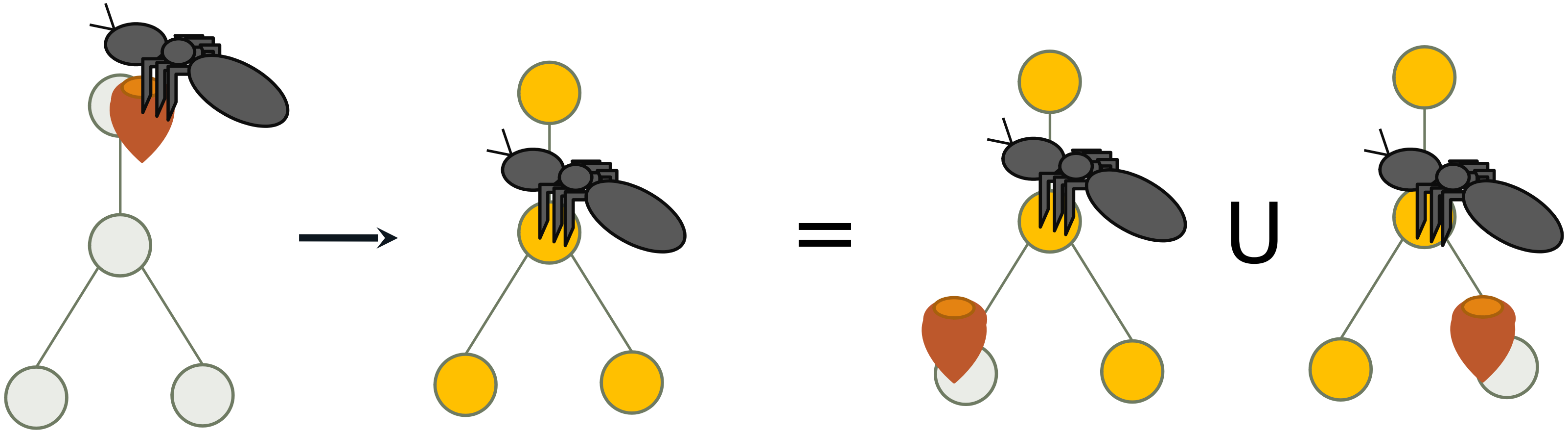
Solution

- Depth first search while keeping track of if the ant is pushing a nut or not
- Ant can enter a room if
  - neither the ant is pushing nor the room contains a nut, or
  - the room has one or more AVRs



Solution

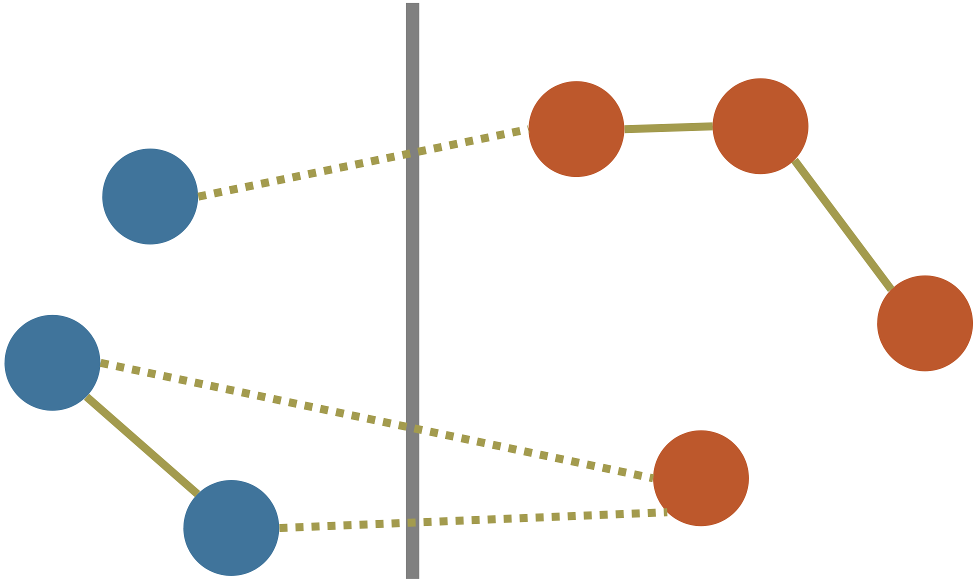
- The ant is considered to lose its nut when it enters a room
  with ≥2 AVRs



G:To be Connected,
or not to be,
that is the Question

Problem Summary

Divide the nodes into two groups (by a threshold) and remove edges connecting nodes in different groups

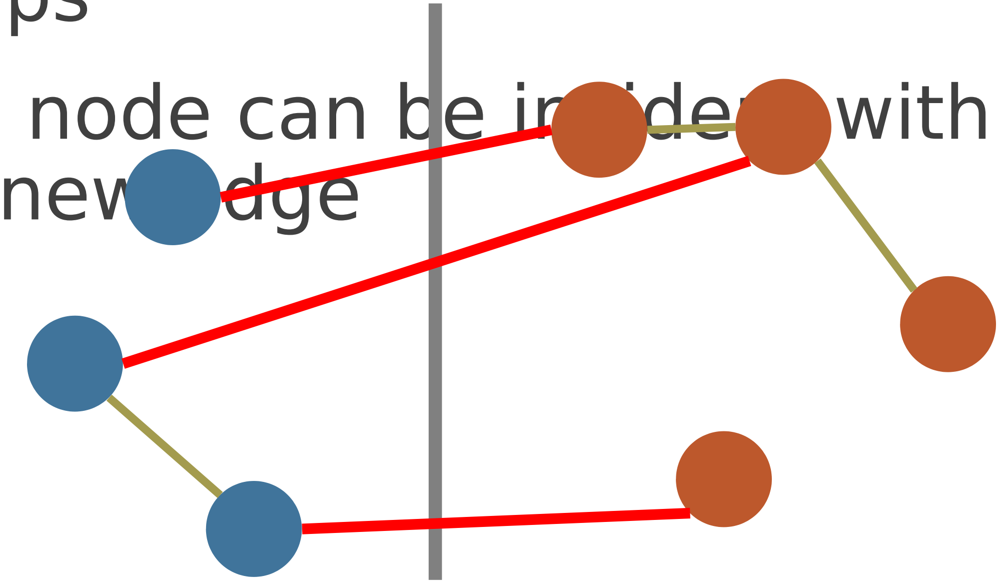


Problem Summary

Make the subgraph connected by adding new edges

1. New edges must connect nodes in different groups

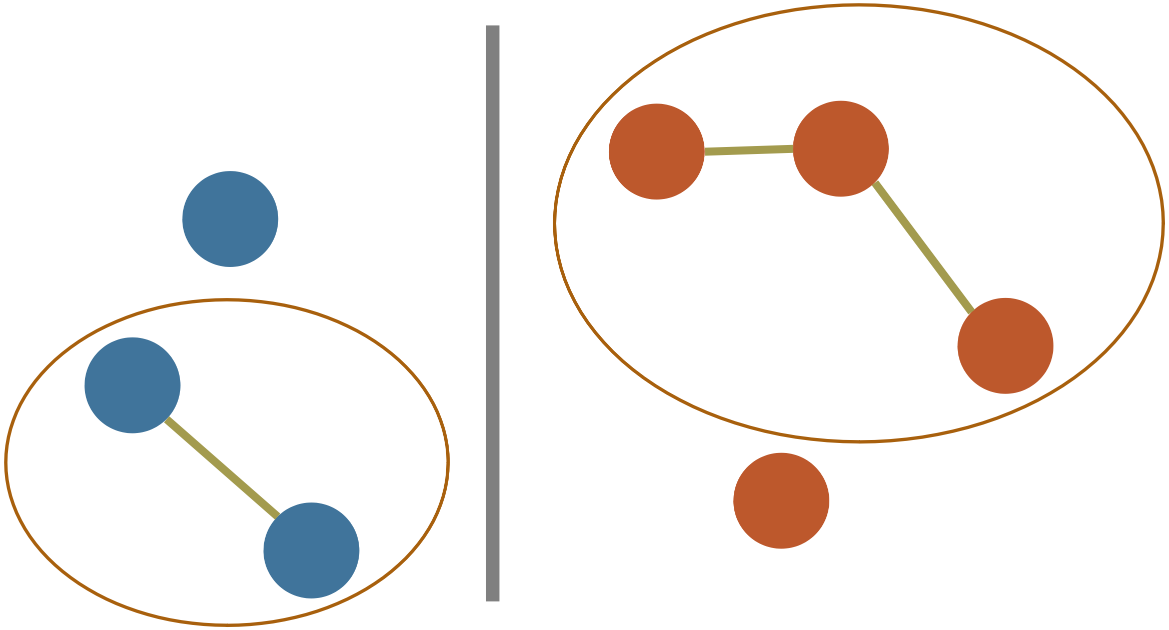
2. Each node can be incident with at most one new edge



Necessary Conditions

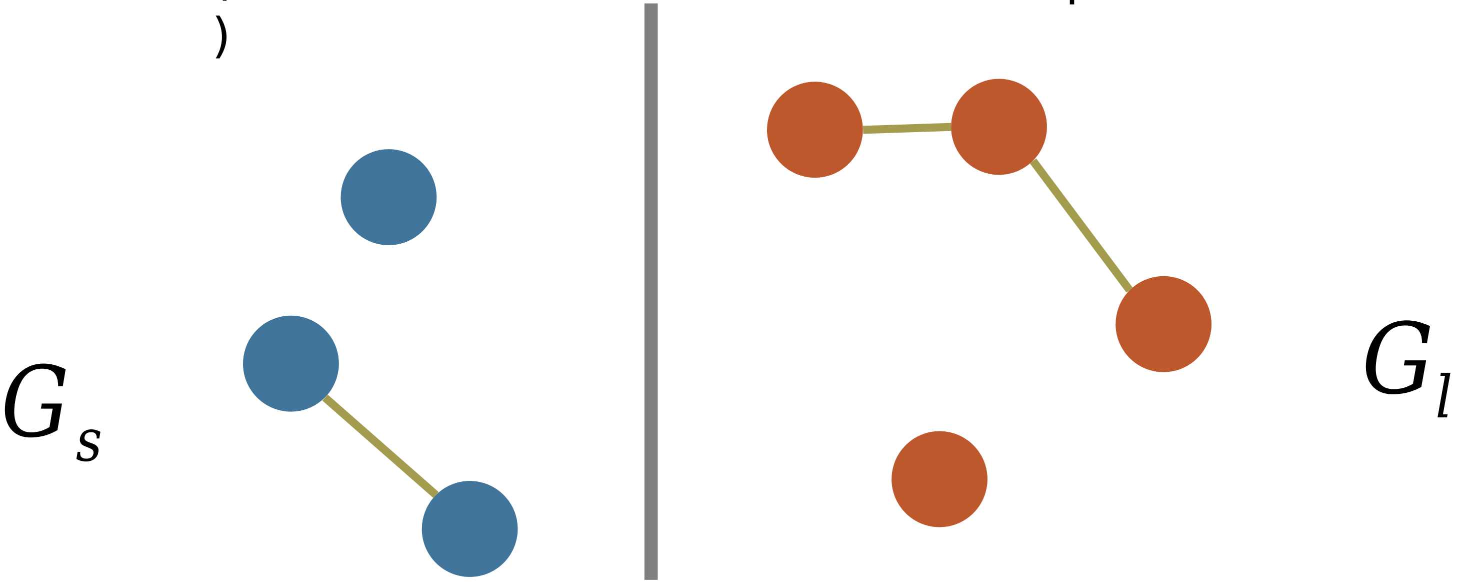
Let be the graph where each connected component corresponds to a node

 is necessary for connectivity



Necessary Conditions

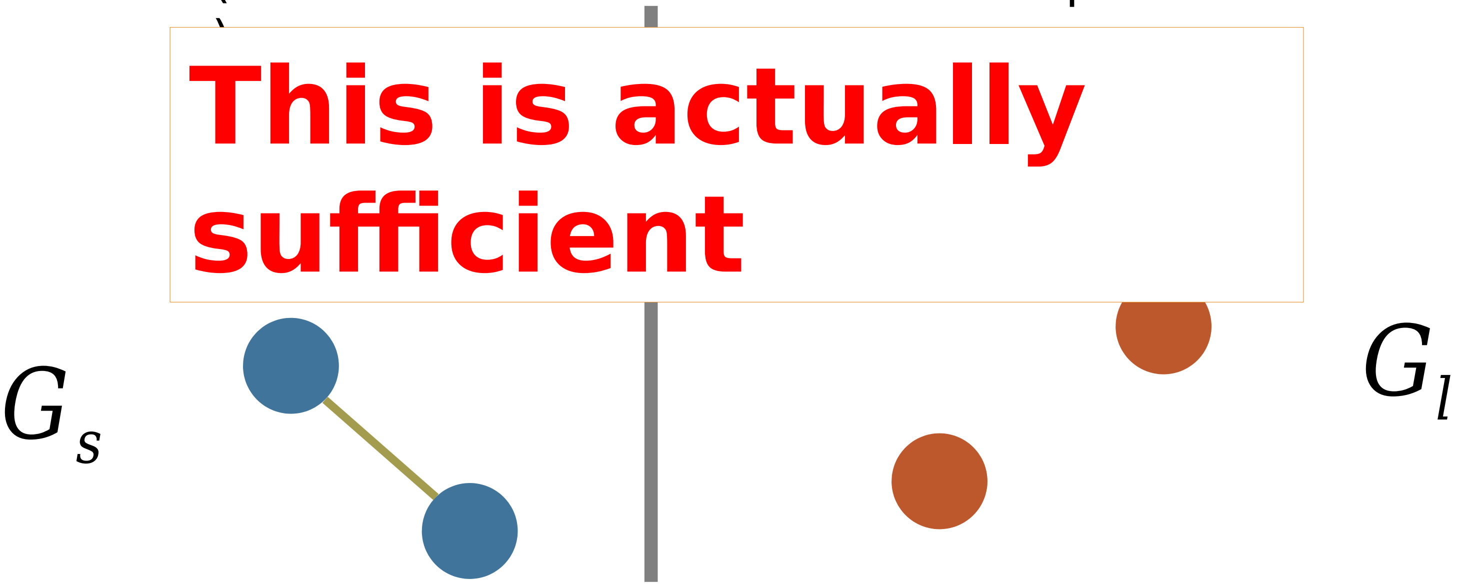
(* is the number of connected components of
)



Necessary Conditions

(* is the number of connected components of)

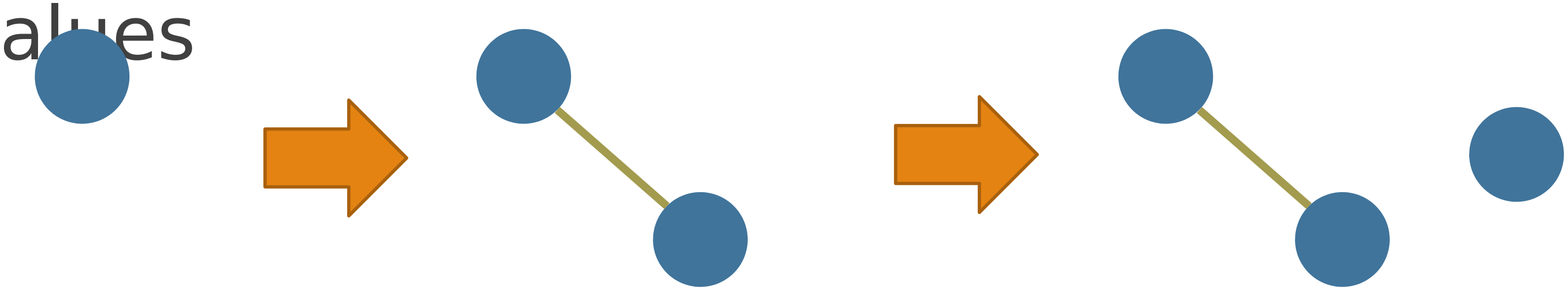
This is actually sufficient



Compute Values

The numbers of nodes and connected components for
can be calculated by checking nodes from smaller values

Same for by checking nodes from larger values



Summary

1. Calculate the numbers of nodes and connected components for two groups with each possible threshold

2. Find the minimum threshold satisfying the necessary condition

The total time complexity is

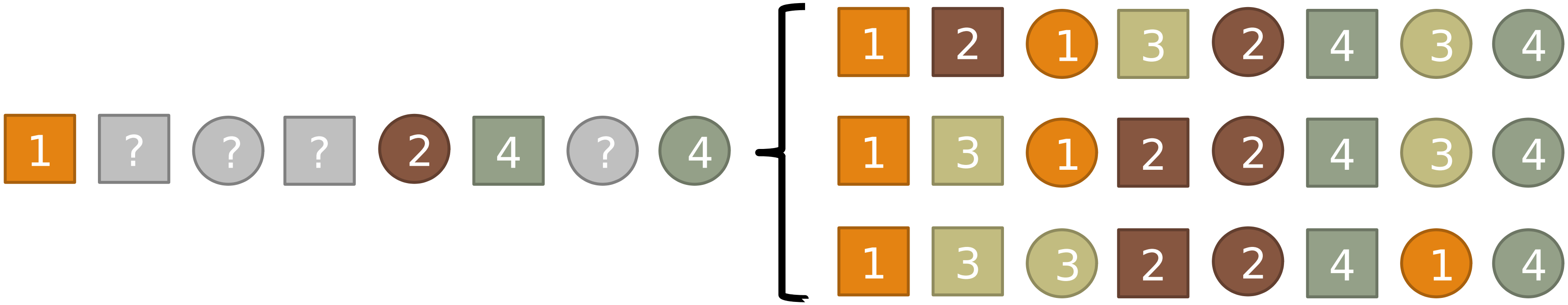
I:High-Tech Detective

Problem Description

You are given a list of events describing the entry and the exit of persons.

- Each ID is appeared once for its entry and once for its exit.
- Some of the ID(s) are missing.

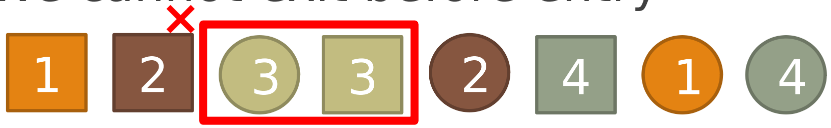
Your task is to calculate the number of consistent ways to fill the missing ID(s) modulo 1,000,000,007.



■ : Visitor entered.

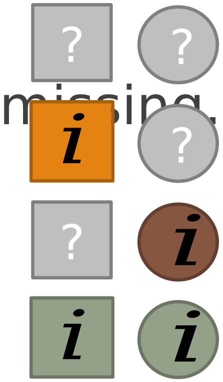
● : Visitor exited.

Note: We cannot exit before entry



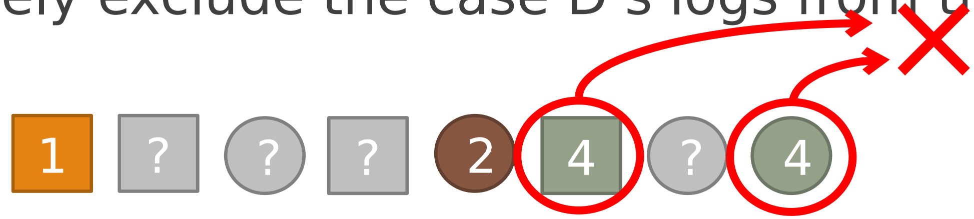
Solution

We can categorize ID(s) into the following four groups:



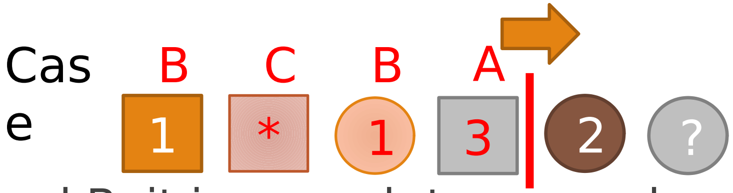
- Case A: Both the entry and the exit logs are missing.
- Case B: Only the entry log remains.
- Case C: Only the exit log remains.
- Case D: Both logs remain.

We can safely exclude the case D's logs from the input.



Solution

Basic idea is to determine the cases for missing IDs from left to right.



For the case A and B, it is enough to remember #(unclosed IDs) (having entered but not having exited yet) at i-th position.

- For the case A, we determine each ID at its entry.

For the case C, it is not necessary to determine each ID for its entry.

- We can determine its ID at its exit.

The following values can be calculated in O() time complexity by Dynamic Programming (DP).

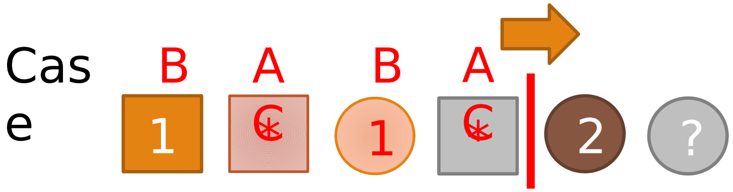
dp[i][I_caseA][I_caseB] := #combinations at the i-th event where

I_caseX = #(unclose IDs) of the case X (X ∈ {A, B, C})

(I_caseC can be uniquely determined by i, I_caseA, and I_caseB).

Solution

By determining the case A’s ID at its exit, we can combine the case A
with C.



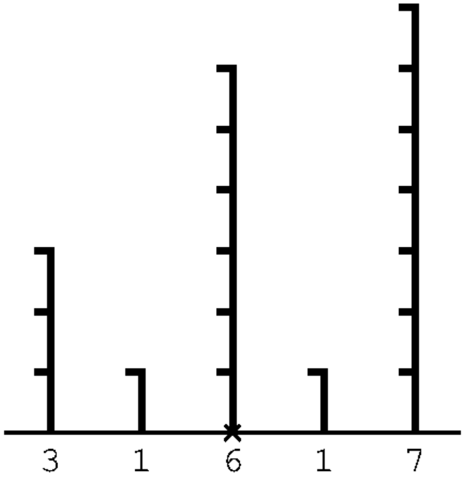
The following values can be calculated in O() time complexity by DP.

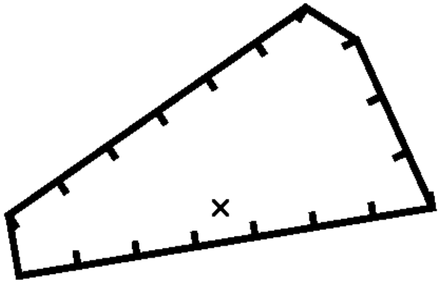
dp[i][I_caseAC] := #combinations at the i-th event where
I_caseAC = #(unclose IDs) of the case A and C.

E: Jewelry Size

Story

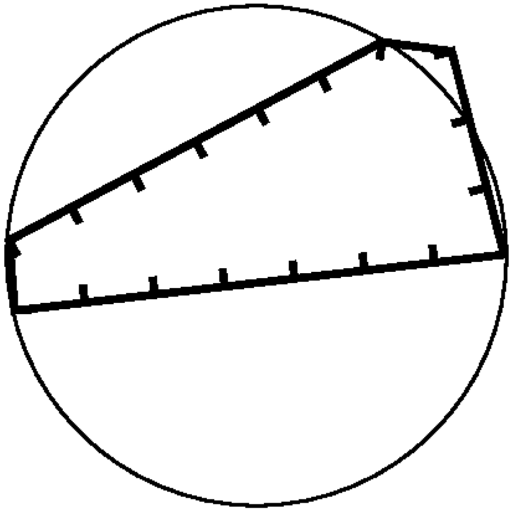
- Given lengths of the edges of a polygon.

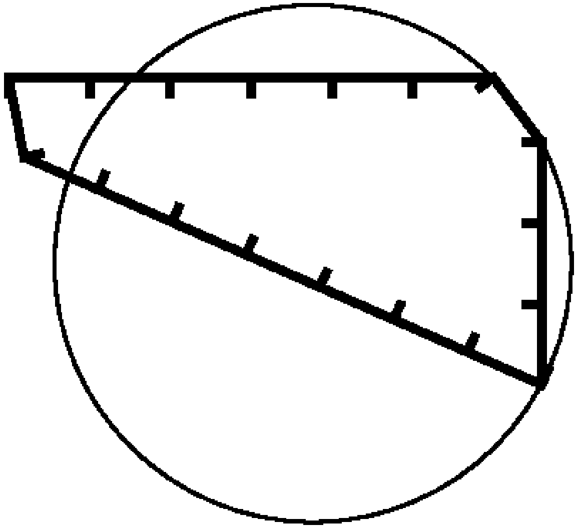




Story

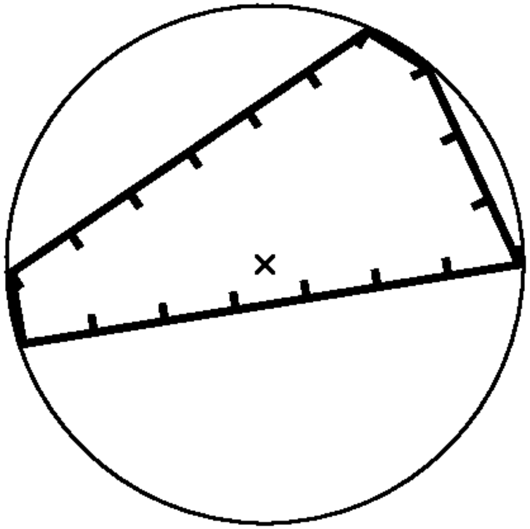
- There are many polygons each of which has the edges with the given lengths.





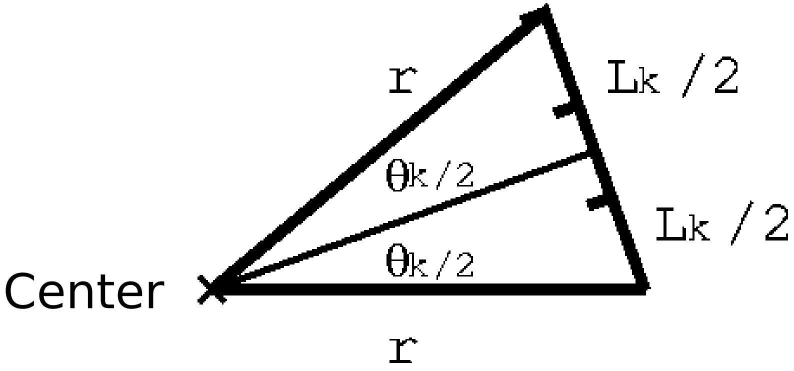
Story

- Your task is to find a polygon that has the circumscribed circle and print its radius.



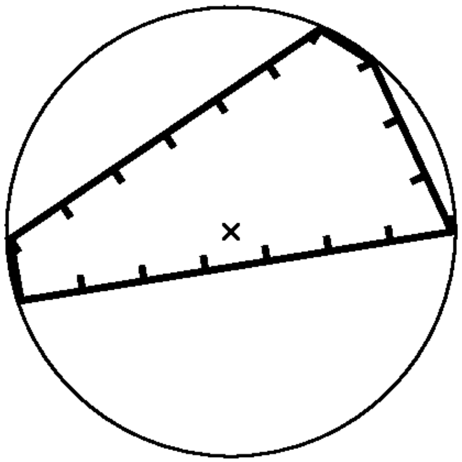
Solution

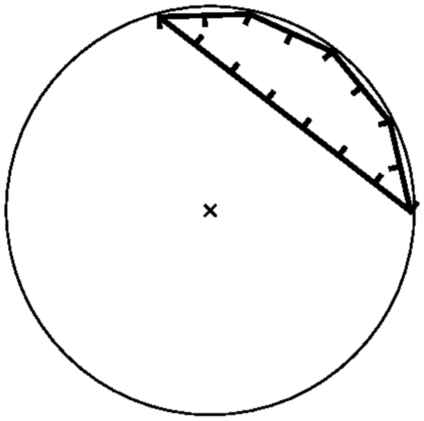
Fixing radius to an approximate value, compute angle



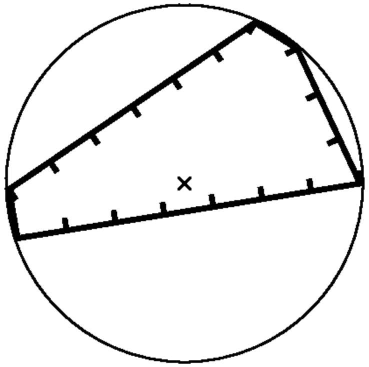
Solution

There are two patterns:

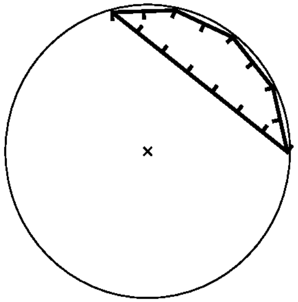




Solution



Type 1 All the sum of angles is :



Solution

Type 2 The maximum angle is equal to the sum of the other angles:

Where the maximum length of the given edge lengths:

Solution

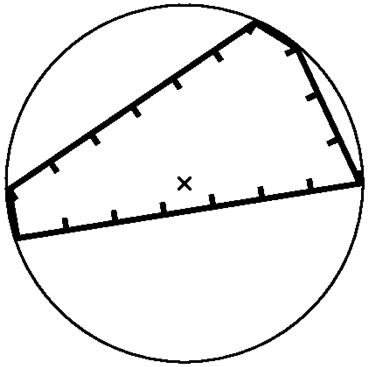
The minimum candidate radius:

Solve one of the nonlinear equations:

or

with the bisection method, the Newton method, etc.

Solution



Type 1: minimum radius

maximum radius

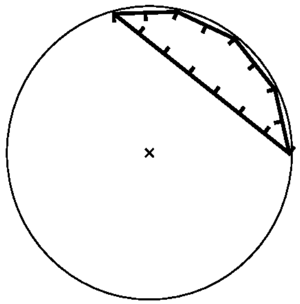
max length of circumference

max radius

input example

1000

6000 6000 6000 … 6000



Solution

Type 2: minimum radius

maximum radius

and

input example

1000

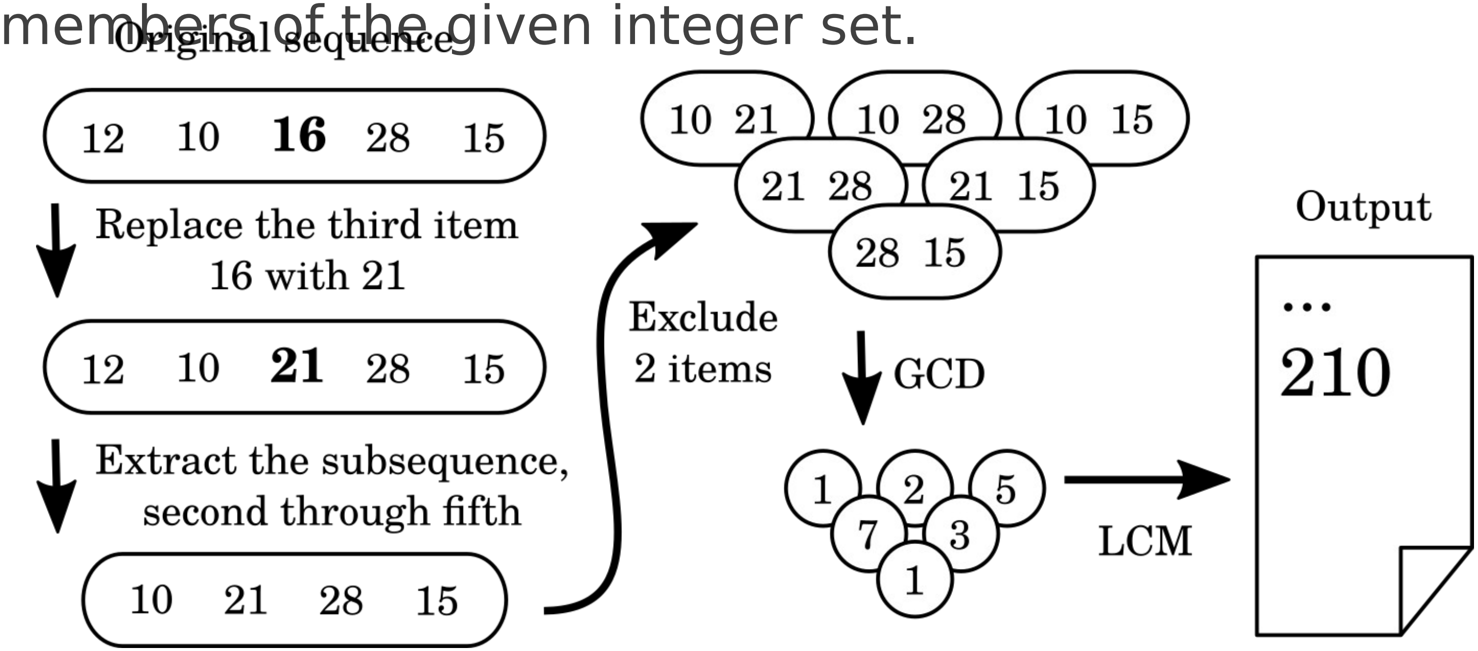
6000 7 7 7 7 7 7 7 6 6 6 … 6

H: LCM of GCD s

Problem

Find LCM of GCDs of the subsets of a given multiset of integers.

The subsets are those obtained by excluding some of the members of the given integer set.



Its naïve implementation requires taking GCD times, which is too computationally heavy to meet the time constraint.

GCD, LCM, and Factorization

Assume that the members of a set are factorized as

 where is the -th prime. With

then GCD and LCM of all the members of the set are

When GCDs of subsets excluding members are

, factorized as , then the value to computed is .

is the smallest among the members of   Thus, those subsets excluding members with the largest have the smallest   So, is the -th largest among all the members of .

Finding the -th Largest Factor Based on a Specific Prime

When the largest factors based on prime among are kept in an array, say “top[+1]”, a new member can be incorporated to it by the following procedure.

```
tmp =
for x in 0..k:
  top[x], tmp ← min(top[x], tmp), max(top[x], tmp)
```

As the elements of the array “top” and are powers of , the max and min operations can be substituted by GCD and LCM.

```
tmp =
for x in 0..k:
  top[x], tmp ← gcd(top[x], tmp), lcm(top[x], tmp)
```

Finding the -th Largest Factors Based on All the Primes

The procedure described above can be applied to all the factors based on all the primes simultaneously, as GCD and LCM work for factors based on different primes independently.

```
tmp = 
for x in 0..k:
  top[x], tmp ← gcd(top[x], tmp), lcm(top[x], tmp)
```

Applying this procedure through all the elements leaves the desired value in “top[k]”. This algorithm requires computing GCD and LCM only times.

This operation is associative on sets of integers. As the sequence of integers remains almost the same for all the queries, building a segment tree is beneficial. Building the tree requires calls of GCD and LCM, and only   calls are required for each query and update.

C:Short Coding

Problem 1/2

Find a program with the fewest possible number of lines to solve a given maze.

```
.##S...##.
..#...#...
..#...#...
.###...##.
..........
..........
.##....##.
.#.#..#...
.##...#...
.#.....G#.
```

S : Start
G : Goal
. : Vacant cell
# : Filled cell

Problem 2/2

You can use only the following commands:

Command	Description
GOTO l	Goto the l-th line in the program.
IF-OPEN l	Goto the l-th line in the program if can move forward
FORWARD	Move forward
LEFT	Turn left
RIGHT	Turn right

Solution

Any maze can be solved by the left-hand rule algorithm.

```
LEFT
IF-OPEN 5
RIGHT
GOTO 2
FORWARD
```

The number of lines of a solution

Brute-force search + simulation

Estimation

How many valid programs?

- GOTO 1, …, GOTO 4
- IF-OPEN 1, …, IF-OPEN 4
- FORWARD, LEFT, RIGHT

11 kinds of commands per line

How many states in a simulation (to check one program.)

Where
in maze

Which
direction

Which
line in the program

K:Suffixes may Contain Prefixes

Problem

Given a target string

Bullet string s has n suffixes, s(1) ... s(n)

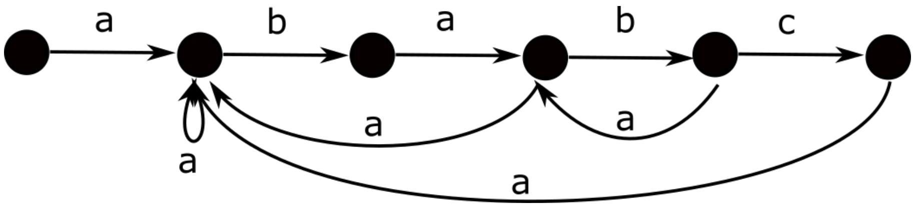
Score = sum of LCP(target, s(i))

Find bullet string, print the maximum score

LCP = length of longest common prefix

Solution

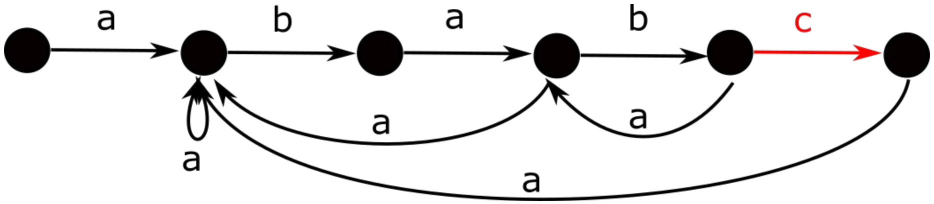
1. Build automaton

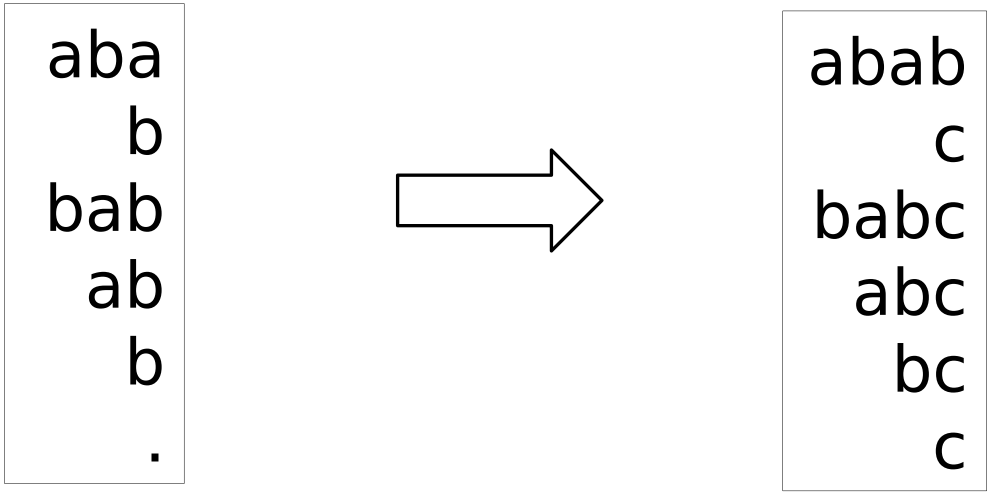


2. Dynamic programming

Automaton

Forward edge makes suffixes longer



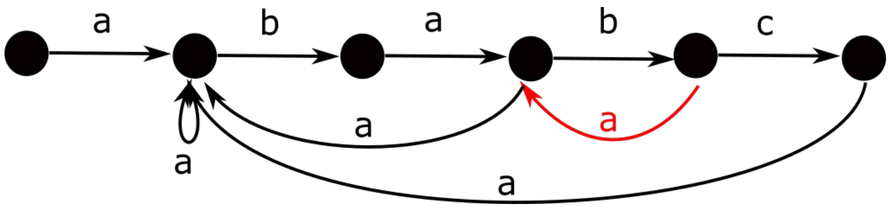


aba
b
bab
ab
b
.

ababc
babc
abc
bc
c

Automaton

- Backward edge fixes some LCP scores
  - Precalculate backward edge scores



aba
b
bab
ab
b
.

aba
ba
baba
aba
ba
a

Fixed score
4

Dynamic programming

- dp(i, j) = maximum score
  - i letters of the bullet string
  - j-th node on the automaton

- Forward edge from j to j+1
  - dp(i+1, j+1) ← max(dp(i+1, j+1), dp(i, j))

- Backward edge from x to y
  - dp(i+1, y) ← max(dp(i+1, y), dp(i, x) + score(x, y))

Dynamic programming

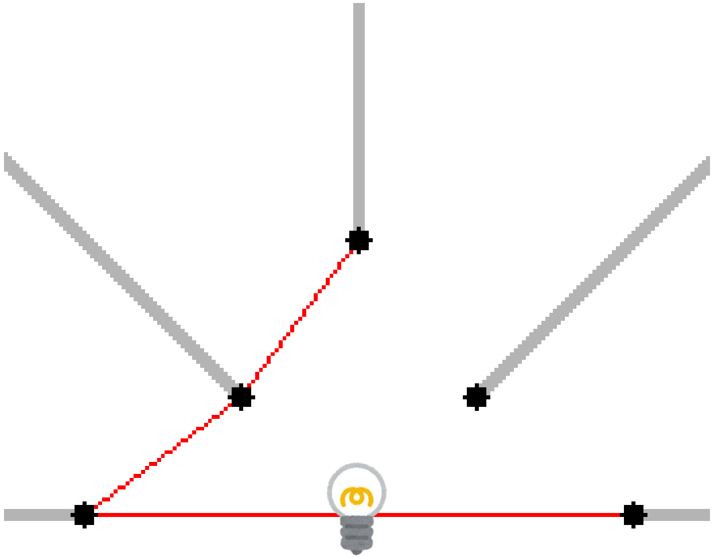
- O(anm)
  - a : kinds of letters, 26
  - m : length of target string
  - DFA has O(am) edges

- O(nm)
  - NFA has O(m) edges

F: Solar Car

Problem Description

Number of poles are placed at the field.
Drives a car from pole **s** to **t**, and then from **t** to **u** within shortest route.
Poles cast infinitely long shadows, and the car cannot go across them.



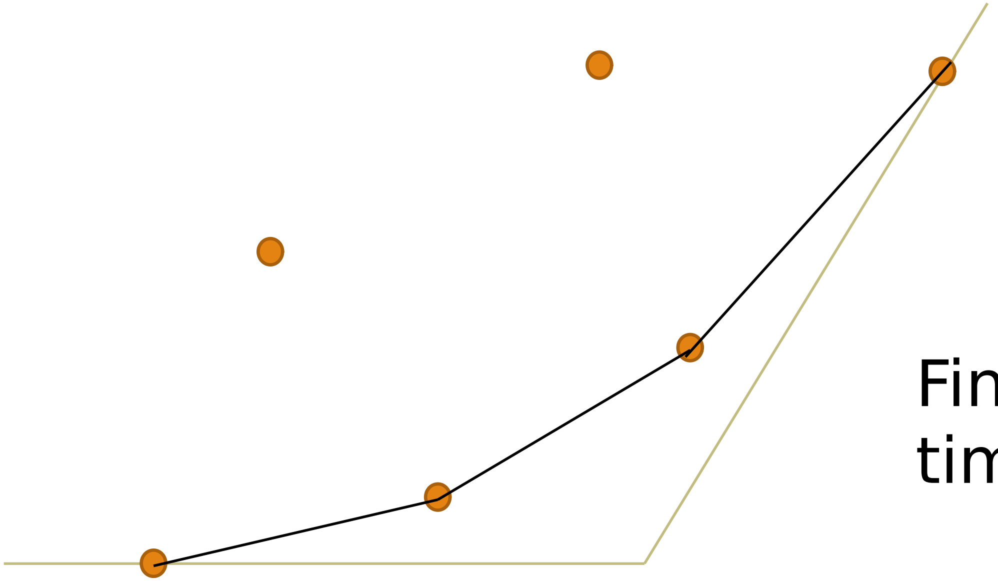
First, **s** and **u** are chosen from each range, and **t** is chosen to maximize the length of the path.

Solution

1. Find the shortest path between each pair of poles.
   - $O(n^2)$
2. Find **t** for all pairs (**s**, **u**).
   - $O(n^2)$
3. Answer the queries.
   - Find cumulative sum and answer each query in O(1)
   - $O(n^2+q)$

Step 1. (slow version $O(n^3)$)

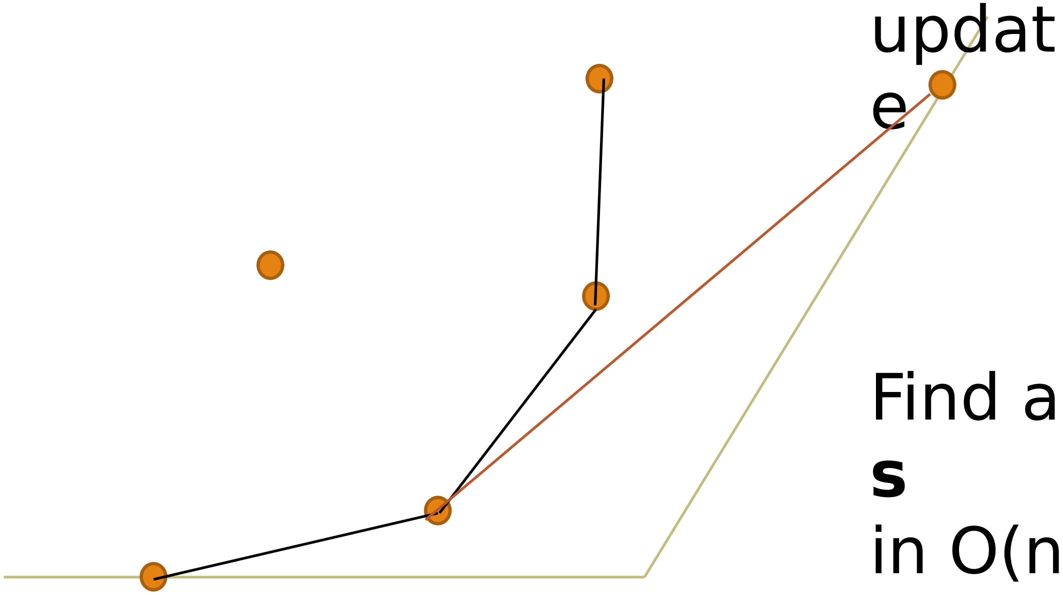
Find the shortest path between each pair of poles (**s**, **x**).



Find convex hull $O(n^2)$ times

Step 1. (fast version $O(n^2)$)

Find the shortest path between each pair of poles (**s**, **x**).



Find all convex hull from
s
in O(n) times

Step 2. O(n^2)

Find **t** for all pairs (**s**, **u**).

Let f(s, u) = t such that

- The shortest route of **s** -> **t** -> **u** is clockwise.
- **t** is chosen to maximize the shortest route

The answer is

max(dist(**s**, f(**s**, **u**)) + dist(**u**, f(**s**, **u**)),
dist(**s**, f(**u**, **s**)) + dist(**u**, f(**u**, **s**)))

Property: f(**s**, **u**-1) <= f(**s**, **u**) <= f(**s**+1, **u**)

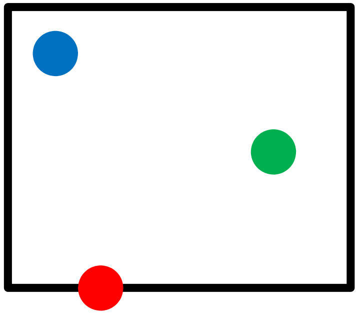
- Can be speeded up to O(n^2)

D:Colorful Rectangle

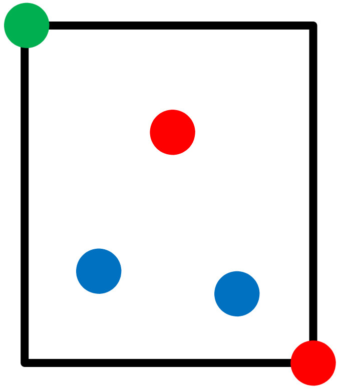
Task

There are red, blue, green points on a plane.

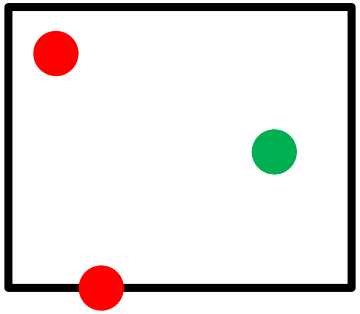
Find the shortest perimeter of a colorful rectangle.



Colorful



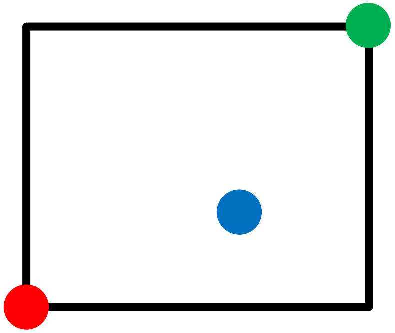
Colorful



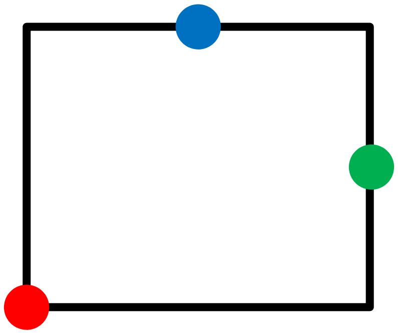
Not Colorful

Two types

By considering rotations and color swapping, we only need to consider the following two types of arrangements.

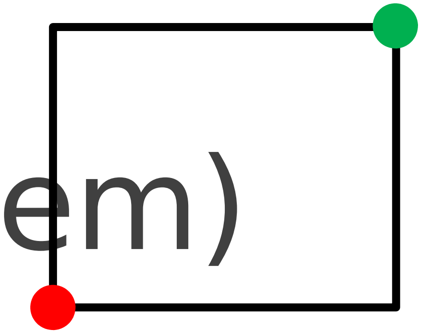


Type1



Type2

Type 0 (Easy problem)



Sort points by x

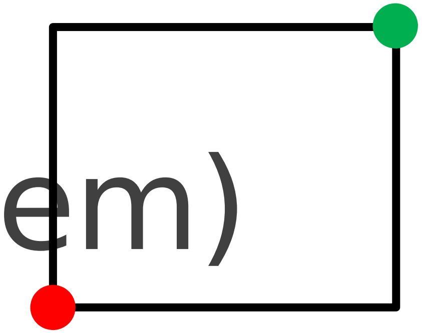
For each p

    maximize q.x + q.y
        s.t.   q.y p.y

Segment Tree (Range maximum query)

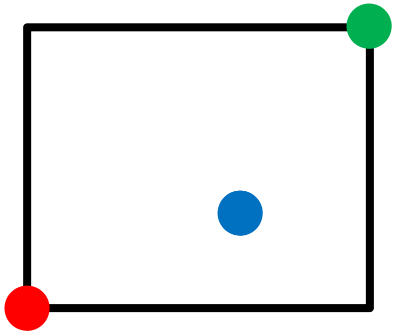
O(n log n)

Type 0 (Easy problem)



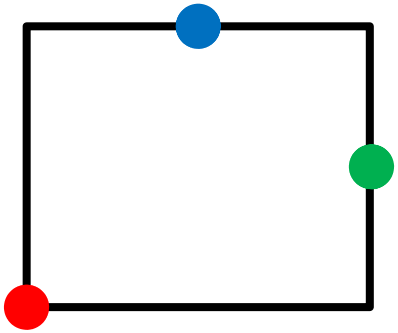
```
Sort points by x
Initialize segtree T
For each p
    if p is red
        T.insert(p.y, p.x+p.y)
    if p is green
        p.x+p.y-T.query(p.y)
```

Type 1



```
Sort points by x
Initialize segtree R and B
For each p
    R.insert(p.y, p.x+p.y)
    B.insert(p.y, R.query(p.y))
    p.x+p.y-B.query(p.y)
    O(n log n)
```

Type 2 (Difficult)



Sort points by x

For each g

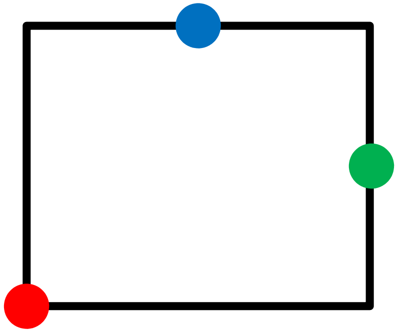
    minimize b.y – r.y – r.x

        s.t. r.y g.y b.y and r.x b.x

Segment Tree (Range Update + Point Query)

    O(n log n)

Type 2 (Difficult)

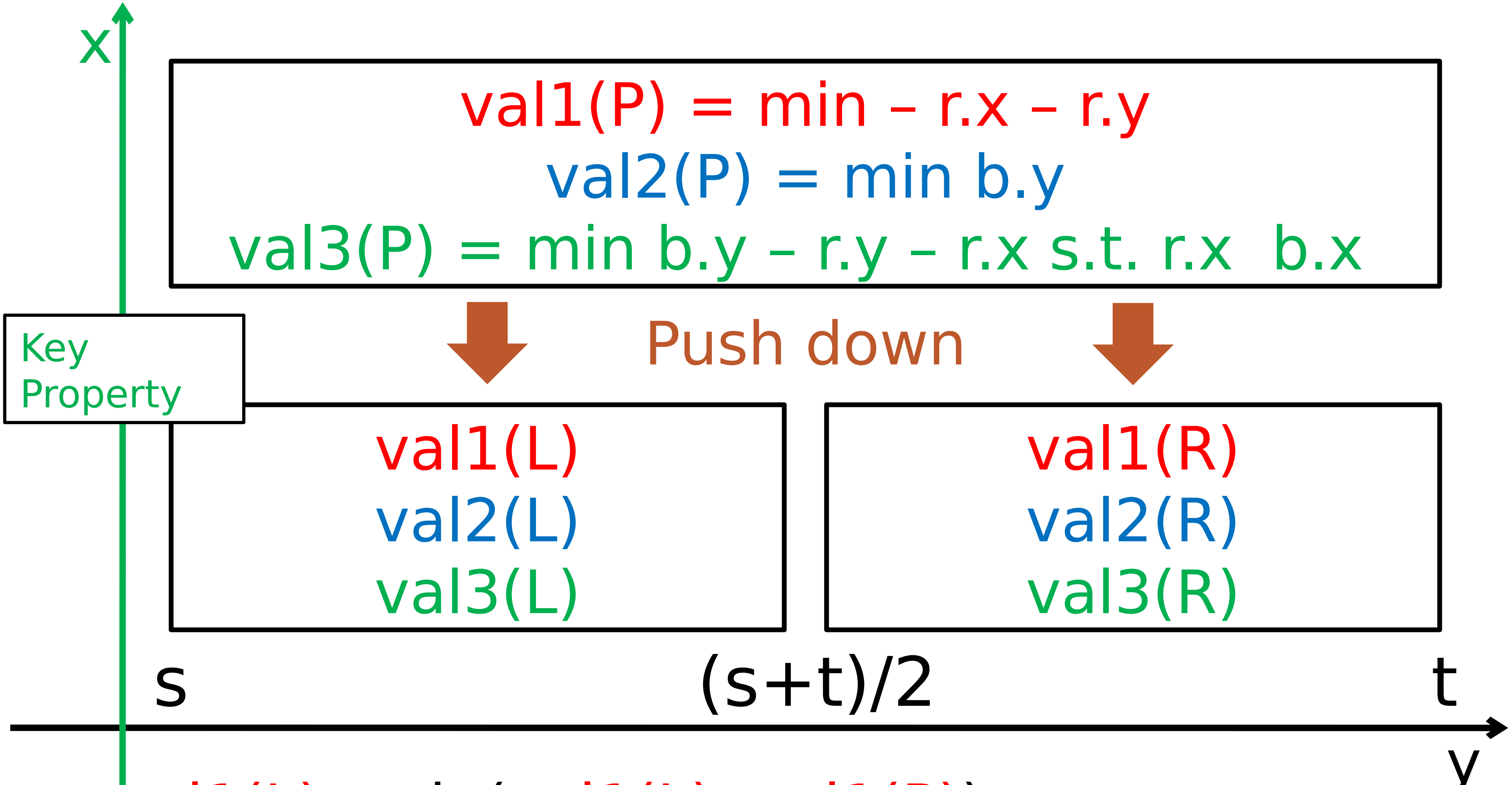


Each segment [s,t) keeps three values

1. minimum – r.x – r.y s.t. r.y s
2. minimum b.y s.t. t b.y
3. minimum b.y – r.y – r.x

   s.t. r.x b.x and r.y s and t b.y

Answer: g.x + min val3 s.t. s g.y < t



val1(P) = min – r.x – r.y

val2(P) = min b.y

val3(P) = min b.y – r.y – r.x s.t. r.x b.x

Push down

Key Property

val1(L)

val2(L)

val3(L)

val1(R)

val2(R)

val3(R)

val1(L)=min(val1(L), val1(P))

val2(L)=min(val2(L), val2(P))

val3(L)=min(val3(L), val3(P), val1(L)+val2(P))