Commentaries on Problems

JUDGE TEAM

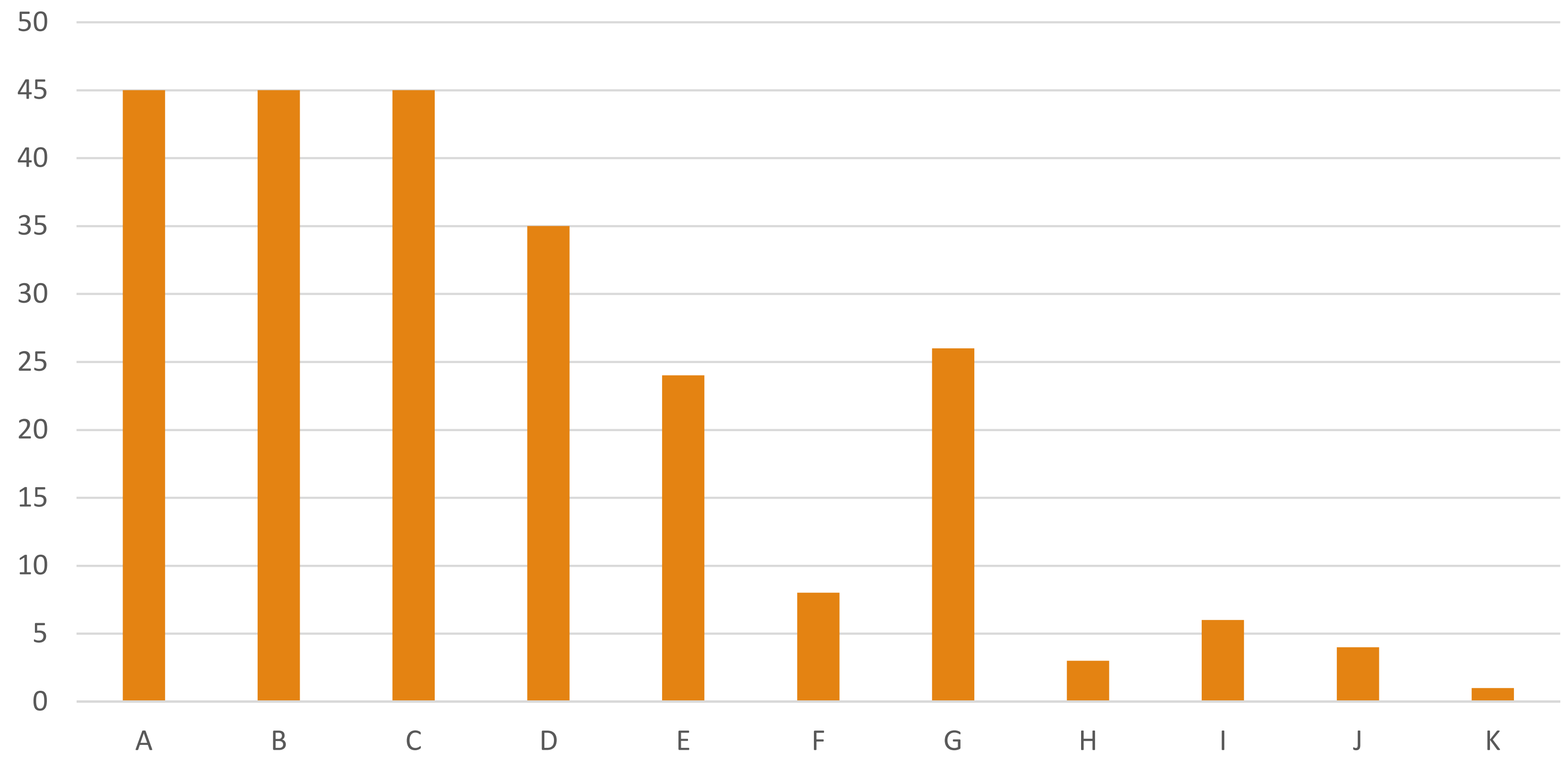
ACM ICPC 2016 ASIA TSUKUBA REGIONAL

Problem Set Design Objectives

- All the teams should be able to solve at least one problem
- All the problems should be solved by at least one team
- No team should be able to finish all the problems too early
- The problem set should demand for expertise in diverse areas

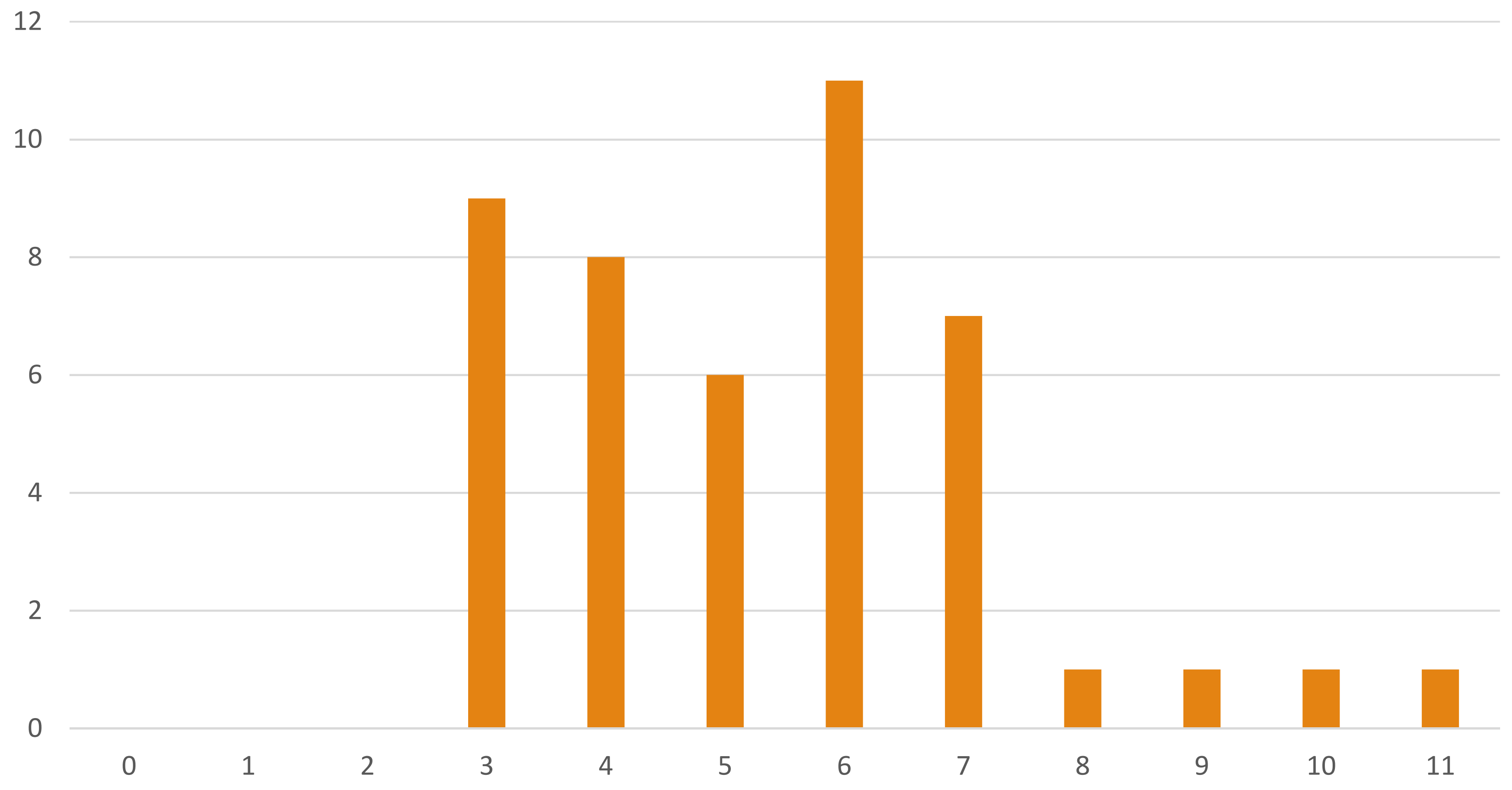
Problems and # of Teams Solved

A	B	C	D	E	F	G	H	I	J	K
45	45	45	35	24	8	26	3	6	4	1



# problems solved & # teams

0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	9	8	6	11	7	1	1	1	1



A: Rearranging a Sequence

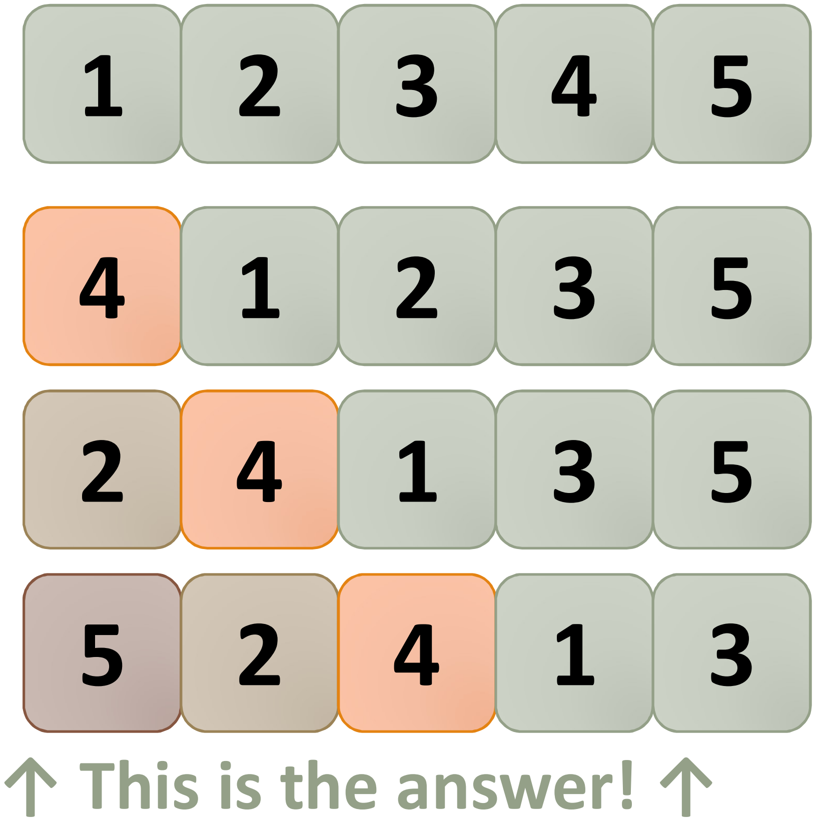
Problem:

Given an integer sequence, execute the requested operations to move elements to the head.

Request: *“Move 4 to the head.”*

Request: *“Move 2 to the head.”*

Request: *“Move 5 to the head”.*



Key Points

The input is large.

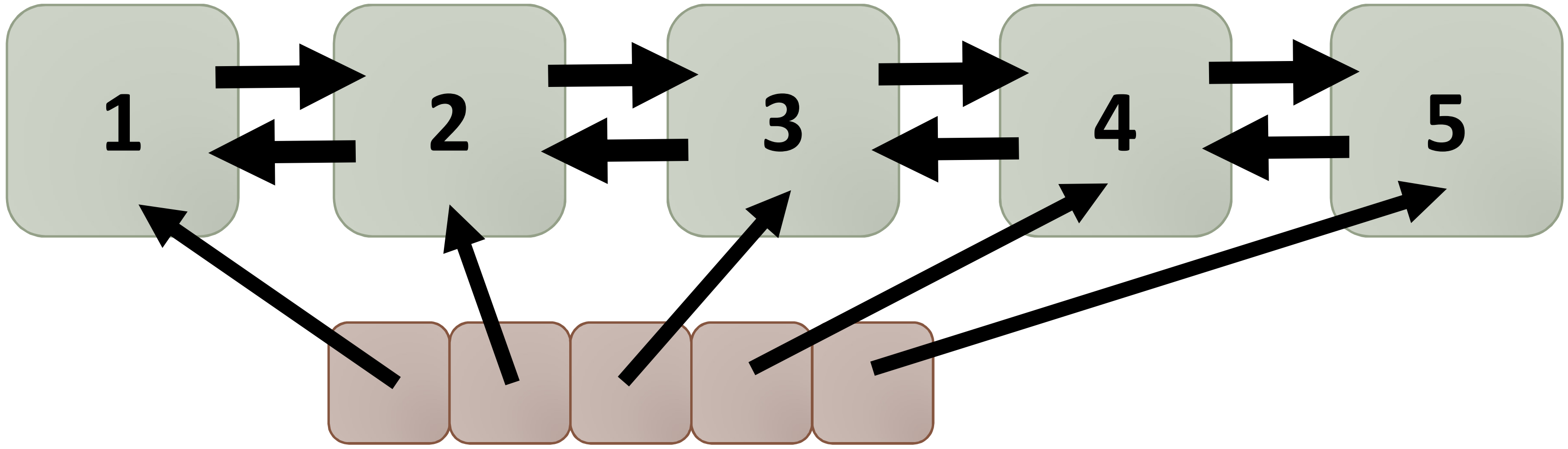
- N (≦ 200000) elements
- M (≦ 100000) requests

It is **too slow** if you represented the sequence as `int[]` or `std::vector<int>`, and really moved the elements.

Possible Solution

Use **a linked list**.

Each rearrangement can be done in O(1) time.

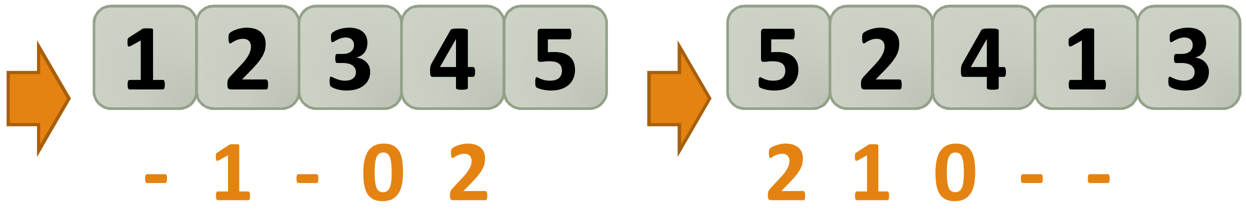


Other Possible Solutions

- Update only **timestamps** per each request. **Sort at the end.**

O(1) per each timestamp update, O(N log N) for sorting.

"Move 4, 2, and then 5."



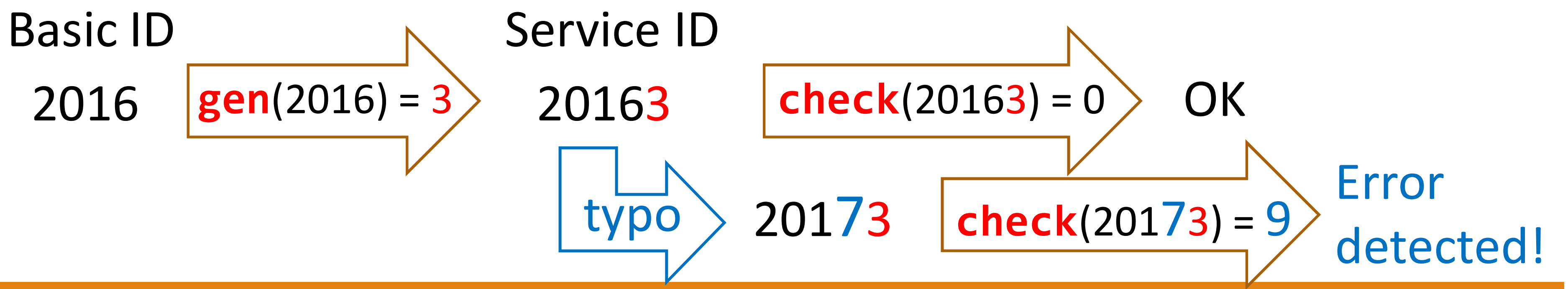
- **Reverse the list of requests**, prepend to the initial sequence, and remove duplicates. O(M+N).

Problem B:
Quality of Check Digits

Problem Summary

Count # of four-digit basic IDs 0000—9999 for which the specified check-digit system doesn't work well

- To detect human errors (typos) in IDs for a service
- Service ID = basic ID + a check digit
  - Cf. ISBN, "My Number", …
- A check digit = **gen**(basic ID)
- Error detection = **check**(service ID') != 0
- Error detection quality depends on **gen** and **check**



Solution: Brute-force!

Given: An operation (multiplication) table x : $i * j = x_{ij}$

gen($abcd$) = $(((0*a)*b)*c)*d$

- **check**($a'b'c'd'e'$) = $((((0*a')*b')*c')*d')*e'$

```
for bID in [0000, …, 9999]:
    sID = append(bID, gen(bID))
    if !(all([check(sID’) != 0 for sID’ in genErrors(sID) ])):
        cnt += 1
print(cnt)
```

genErrors generates 49 possibilities of common errors

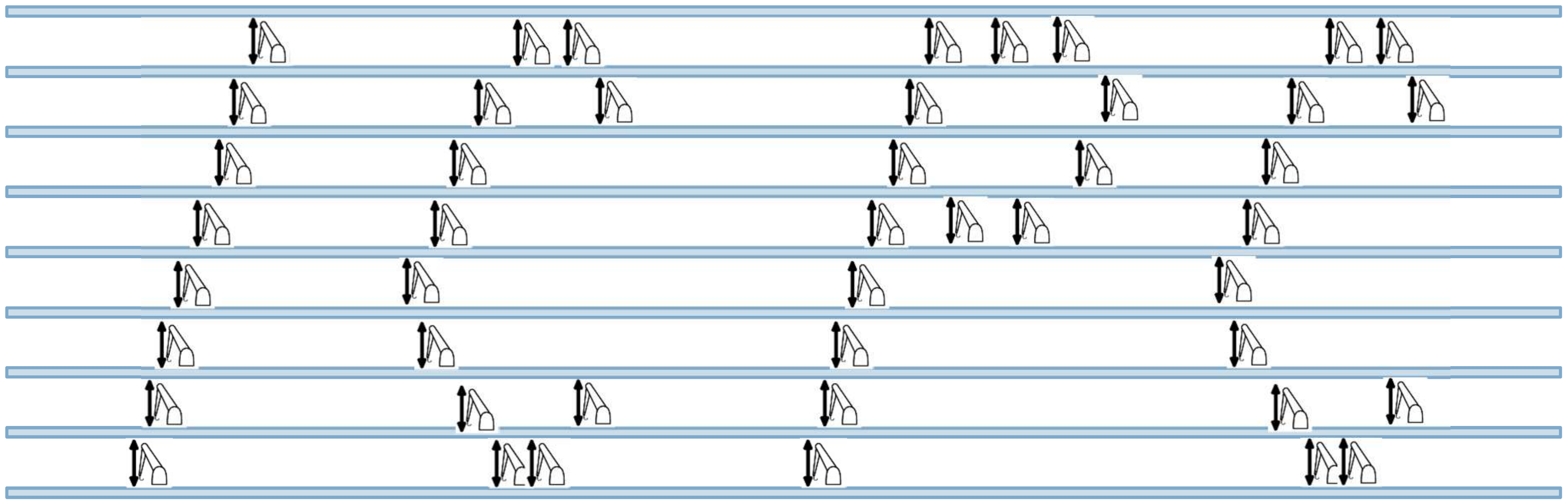
- Altering a single digit (45 ways): 20163 -> 29163
- Transposing adjacent digits (at most 4 ways): 20163 -> 20613

Take-home Message

The error detection used here is Damm algorithm

- Good check-digit algorithm
- Can detect all single-digit errors and all adjacent transposition errors
- But needs only decimal digits (0-9)
  - Cf. MOD11 used on ISBN needs 0-9 and X

Problem C: Distribution Center

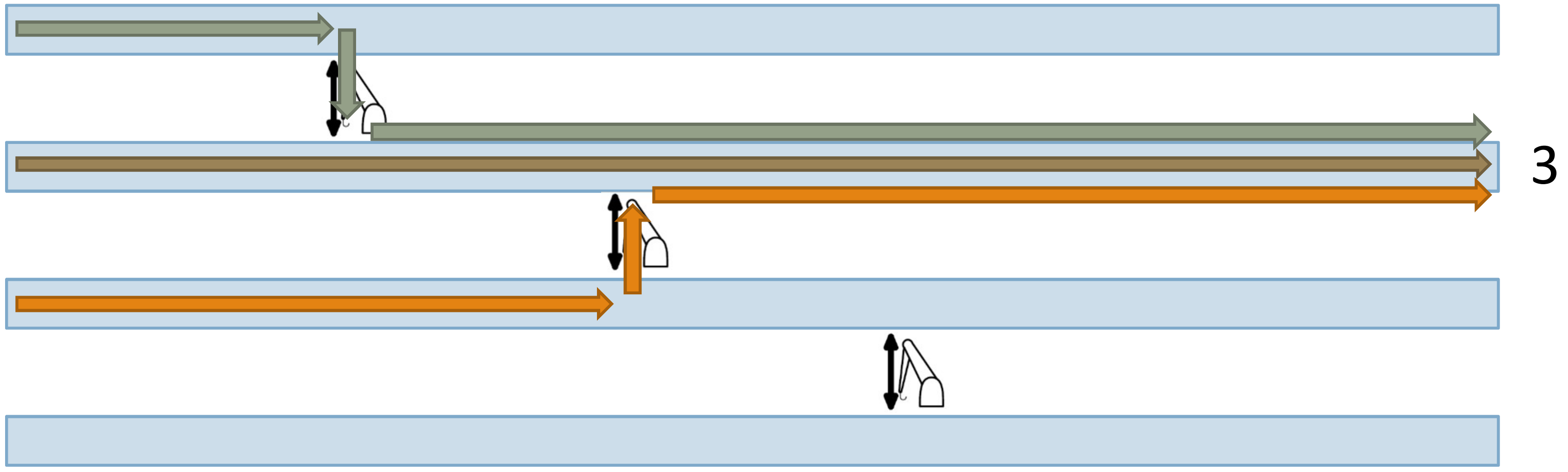


(This is a joke, not included in judge data.)

Problem C:
What you need to compute.

Count the number of entry points to each goal

- Goods go from left to right
- Goods can change its lane with cranes



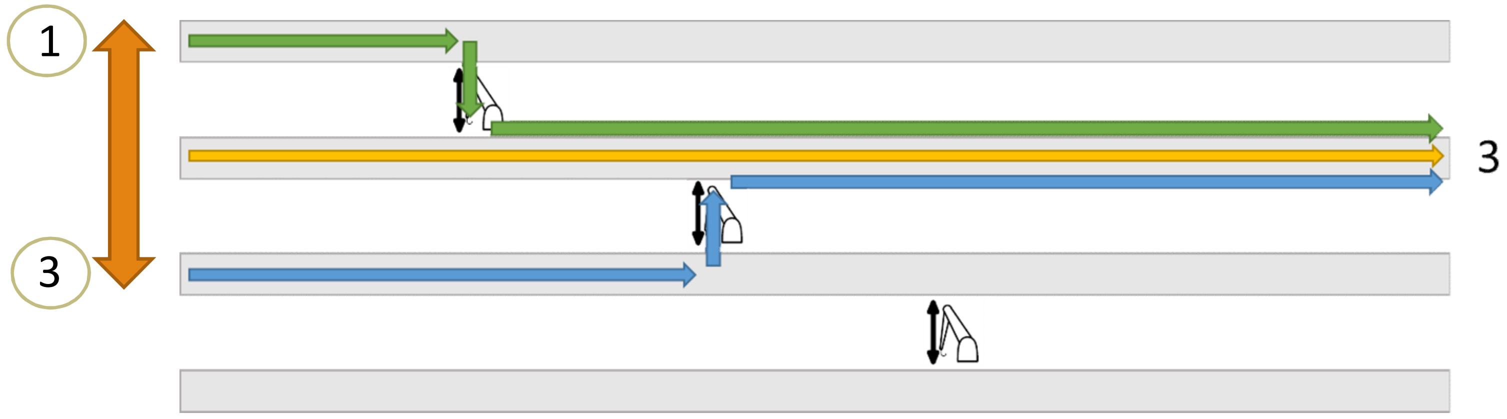
Problem C: Key points

The input size is large ➔ O(nm) algorithms will fail

- n (number of lanes) $\leq$ 200,000
- m (number of cranes) $<$ 100,000

The entry points are in continuous range

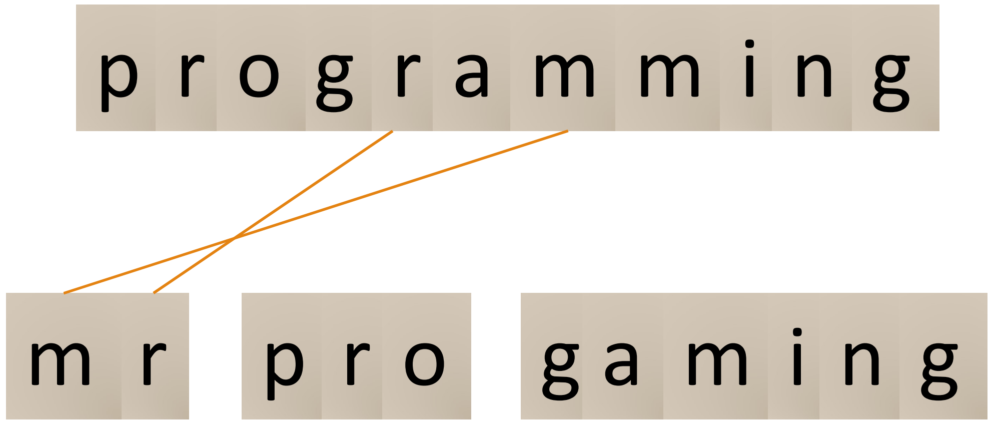
- Compute with the two numbers



D: Hidden Anagrams

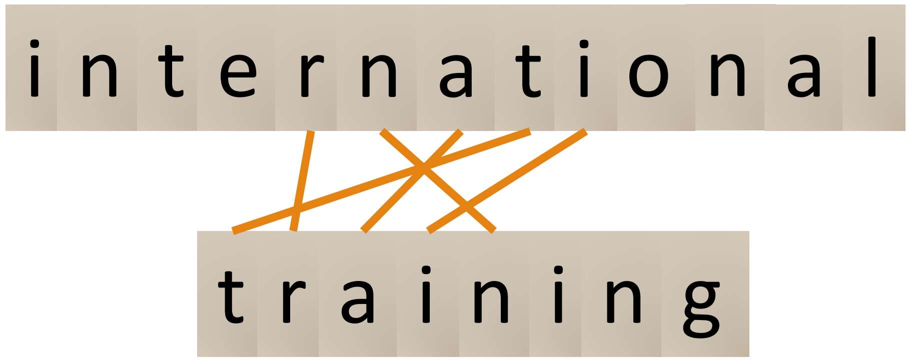
What’s anagrams?

A word or a phrase that is formed by rearranging the letters of another



What’s hidden anagrams?

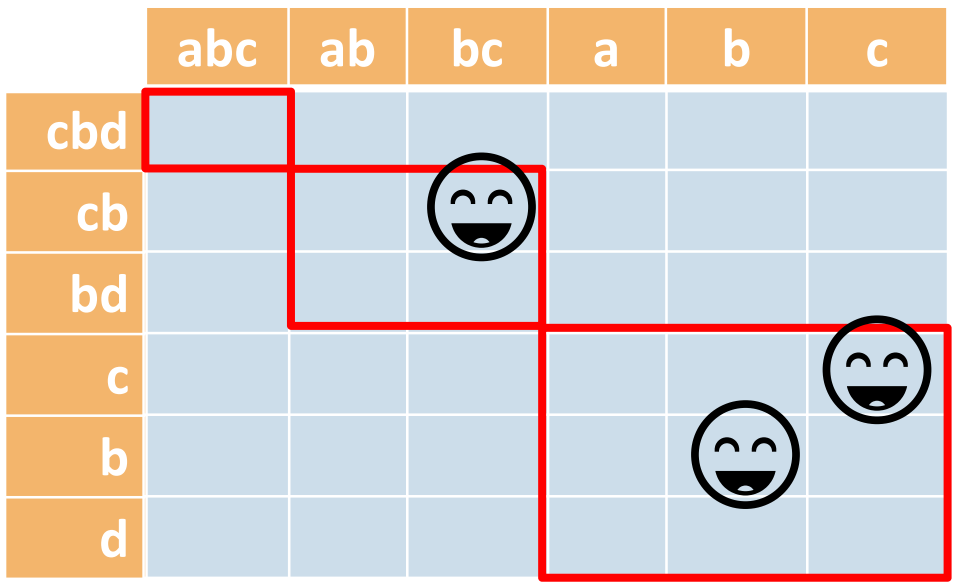
For given two strings, a substring of one that is an anagram of some substring of the other



A naïve solution: all-to-all comparison

abc

cbd



- Simple but slow
  Two strings of lengths 4,000 have more than 20 billion pairs of substrings of the same lengths

A better solution: compare only when substrings have the same summaries

a	31
b	90
c	67
d	43
⋮	
z	98

aabc	31+31+90+67=219
baca	90+31+67+31=219

- Record a summary of every substring of one string and examine every substring of the other
  - Use a hash table or a sorted list
- A naïve summary calculation may require O(length^3) operations that can be reduced to O(length^2) by removing duplicate ones

E. Infallibly Crack Perplexing Cryptarithm

Symbols possibly appear in the original equation are

()+-*01=

8 different symbols

- **Replace all alphabetical letters in the input to these symbols.**
  - **At most 8! (= 40,320) patterns** not too many
  - **If 9 or more different letters are in the input, the answer is 0.**

- **Parse each equation obtained by the replacement**
  - **By recursive descent parsing algorithm, perhaps.**
  - **Deal parse errors properly.**
  - **Count only strings which is successfully parsed and the equality holds.**

Example

Input: `ICPC`

Possible replacement (3 examples out of ${}_8C_3 \times 3! = 336$ patterns)

`0=1=  (I→0,C→=,P→1)`

...Syntax Error

`10=0  (I→1,C→0,P→=)`

...Successfully parsed,

but the values of the both side are not equal.

`-0=0  (I→-,C→0,P→=)`

...Successfully parsed and the values are equal.

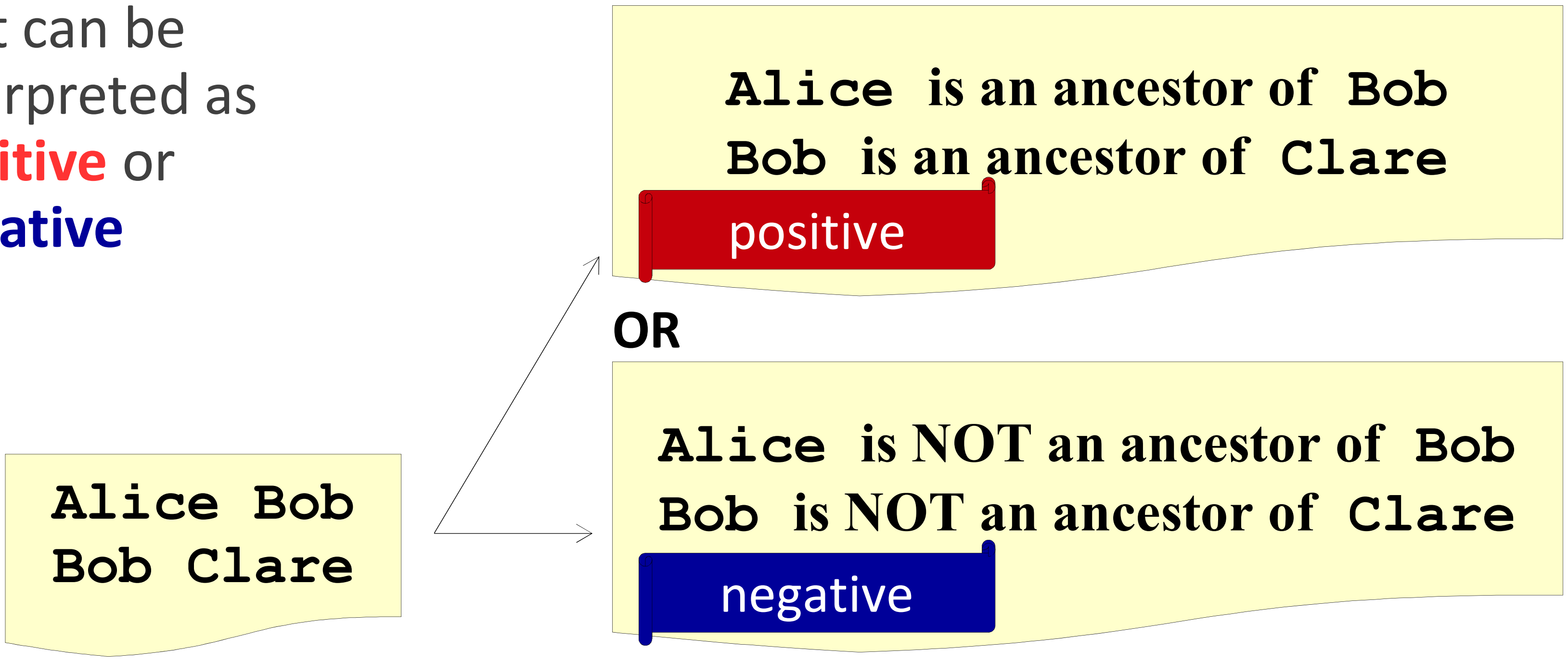
Problem F:
Three Kingdoms of Bourdelot

Input

- A **Hypothesis** that p is an ancestor of q

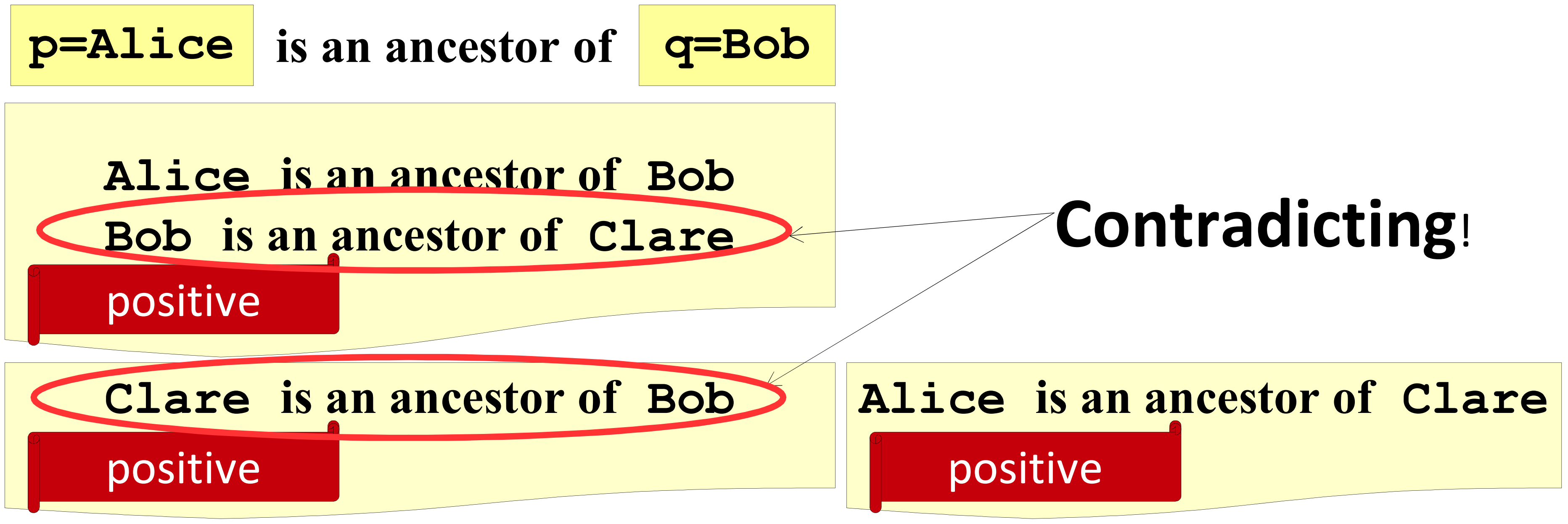
p=Alice **is an ancestor of** **q=Bob**

- **Documents** that can be interpreted as positive or **negative**


Alice Bob
Bob Clare
Alice is an ancestor of Bob
Bob is an ancestor of Clare
positive
OR
Alice is NOT an ancestor of Bob
Bob is NOT an ancestor of Clare
negative

Problem

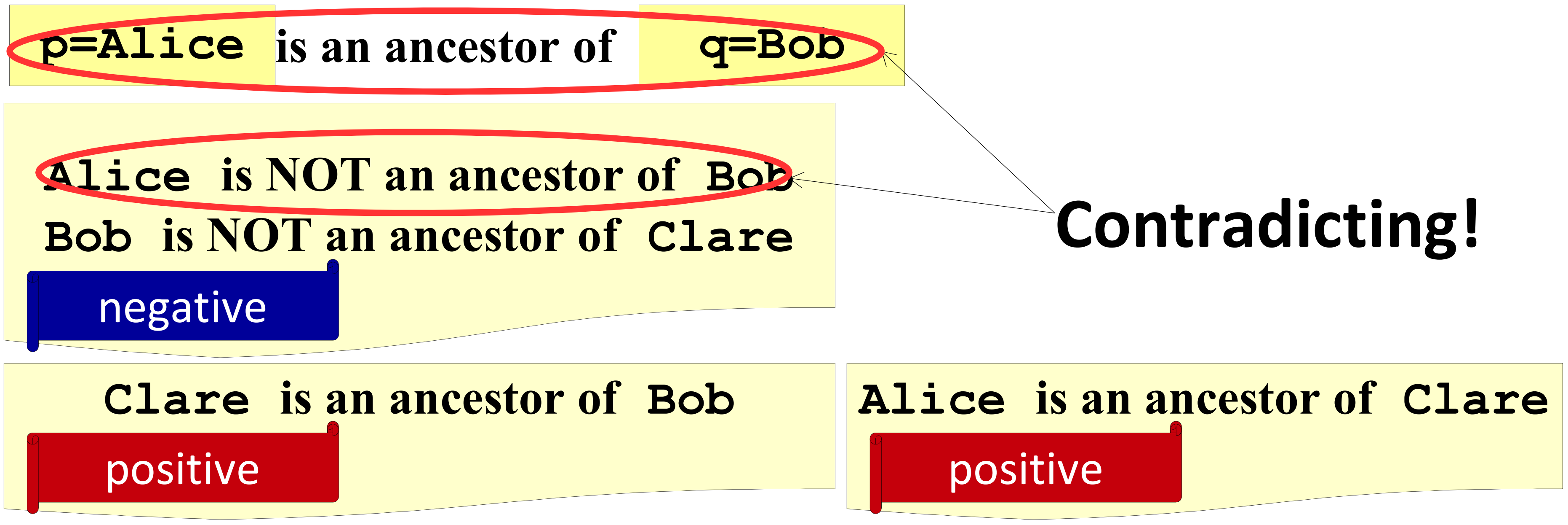
Can we assign 'positive' or 'negative' such that the documents and the hypothesis are not contradicting?



p=Alice is an ancestor of q=Bob
Alice is an ancestor of Bob
Bob is an ancestor of Clare
positive
Clare is an ancestor of Bob
positive
Alice is an ancestor of Clare
positive
Contradicting!

Problem

Can we assign 'positive' or 'negative' such that the documents and the hypothesis are not contradicting?



p=Alice is an ancestor of q=Bob

Alice is NOT an ancestor of Bob
Bob is NOT an ancestor of Clare
negative

Clare is an ancestor of Bob
positive

Alice is an ancestor of Clare
positive

Problem

Can we assign 'positive' or 'negative' such that the documents and the hypothesis are not contradicting?

p=Alice **is an ancestor of** **q=Bob**

Alice is an ancestor of Bob
Bob is an ancestor of Clare
positive

OK!

Clare is NOT an ancestor of Bob
negative

Alice is an ancestor of Clare
positive

Solution

Calculate S = "set of true ancestor-descendant pairs" greedily.

1. Put the hypothesis to S.

S = {<p, q>}

p=Alice is an ancestor of **q=Bob**

S = {
<Alice, Bob>
}

Alice is an ancestor of **Bob**
Bob is an ancestor of **Clare**

Clare is NOT an ancestor of **Bob**

Alice is an ancestor of **Clare**

Solution

2. If an unlabeled document D has a pair in S, then D must be positive.
   - Label D positive and put the pairs in D to S.
   - Take the transitive closure of S.
   - Iterate until converges.

p=Alice **is an ancestor of** **q=Bob**

Alice is an ancestor of Bob
Bob is an ancestor of Clare
positive

Clare is NOT an ancestor of Bob

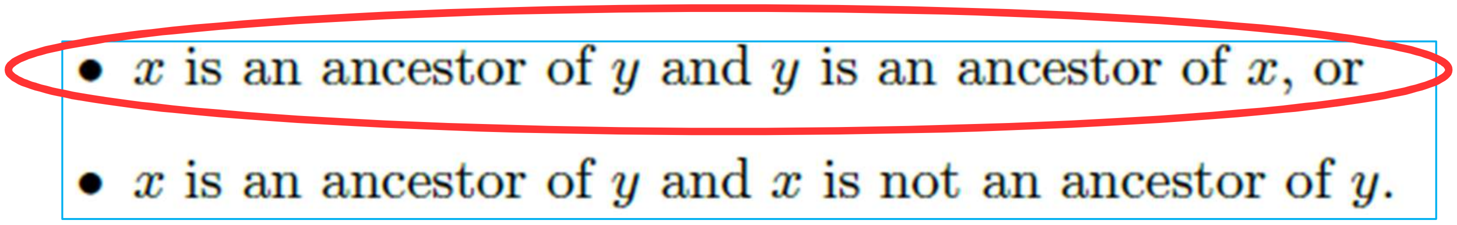
Alice is an ancestor of Clare
positive

S = {
<Alice, Bob>
<Bob, Clare>
<Alice, Clare>
}

Solution

3.If S contains <x,y> and <y,x> for some x and y, output "No".

Clearly contradicting (by the first type of the contradictions)!



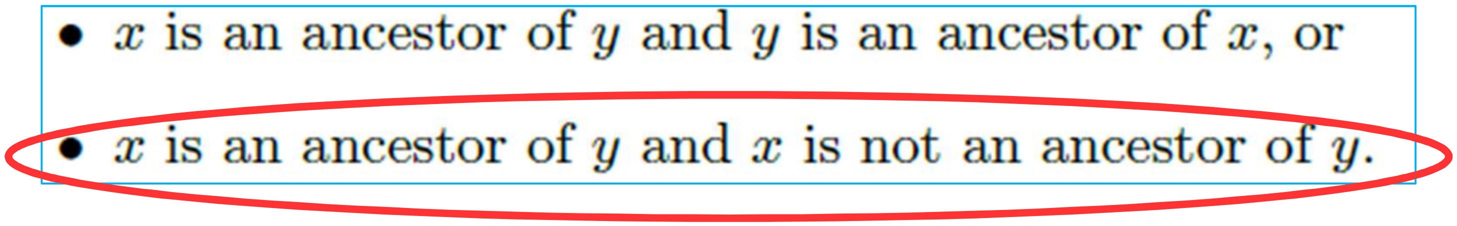
- x is an ancestor of y and y is an ancestor of x, or
- x is an ancestor of y and x is not an ancestor of y.

Solution

4. Otherwise, output "Yes".

We can make unlabeled documents negative.

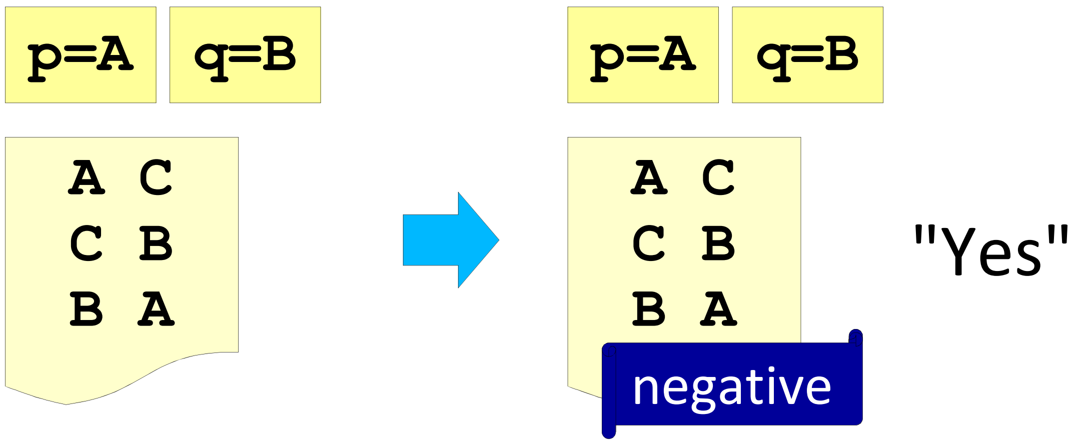
- For any pair <x, y> in such a document, it isn't contained in S (otherwise the document would be labeled positive in Step 2)
- Therefore it doesn't cause contradiction of the second type of the contradictions.



- x is an ancestor of y and y is an ancestor of x, or
- x is an ancestor of y and x is not an ancestor of y.

Solution

Do NOT take transitive closure of a document before Step 1.



Solution

Do take transitive closure in Step 2.

`p=A` `q=B`

```
A B
B C
C D
```

positive

```
B D
D A
```

positive

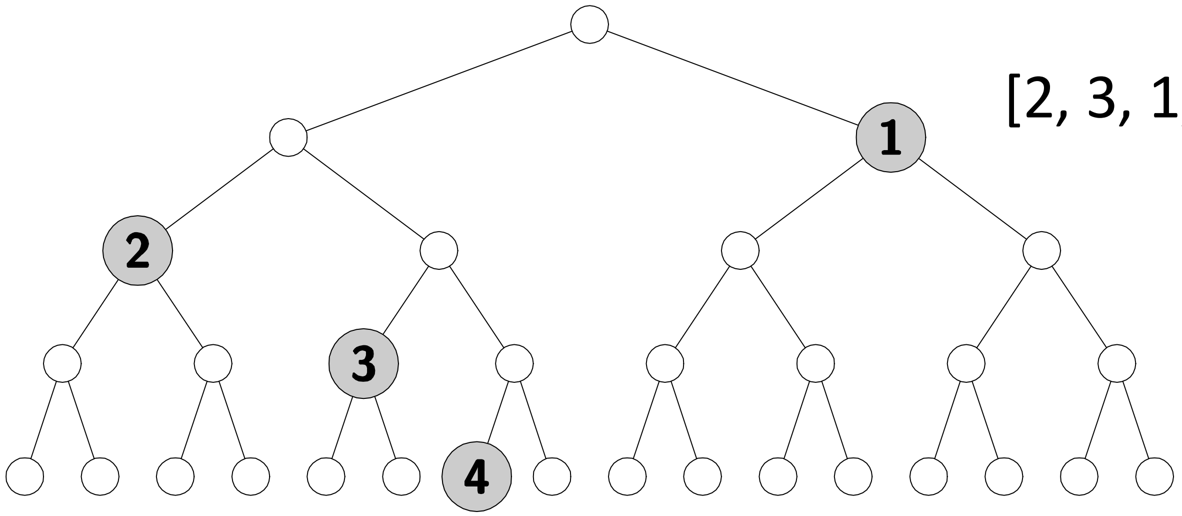
"No"

G: Placing Medals on a Binary Tree

Problem Summary

Can we place a set of medals on a perfect binary tree ?

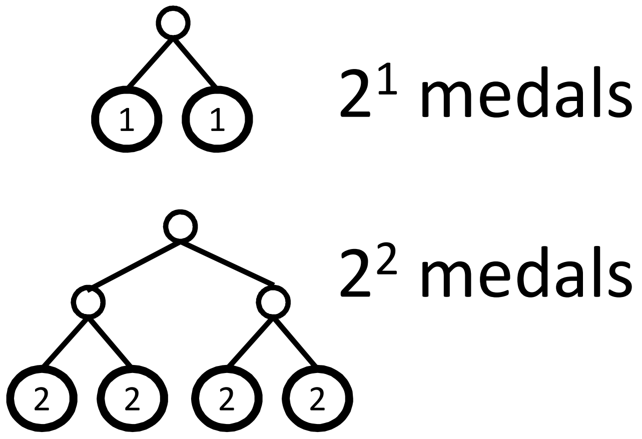
- A medal engraved with *d* should be on a node of depth *d*
- One medal per node
- At most one medal on the paths from any nodes to the root

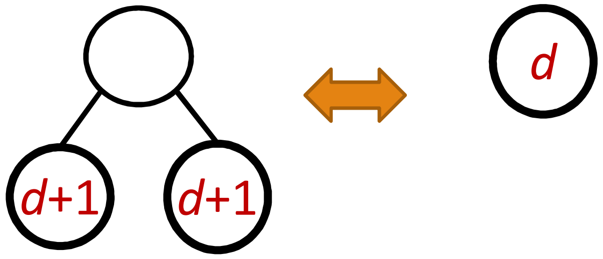


[2, 3, 1, 4] can be placed

Properties (1)

- If all medals are engraved *d*, 2^d medals can be placed.
- Two *d*+1 medals can be placed in place of one *d*.





Properties (2)

1. We can place $[x_1, x_2, \ldots, x_n]$ when

$$\frac{1}{2^{x_1}} + \frac{1}{2^{x_2}} + \cdots + \frac{1}{2^{x_n}} \leq 1$$

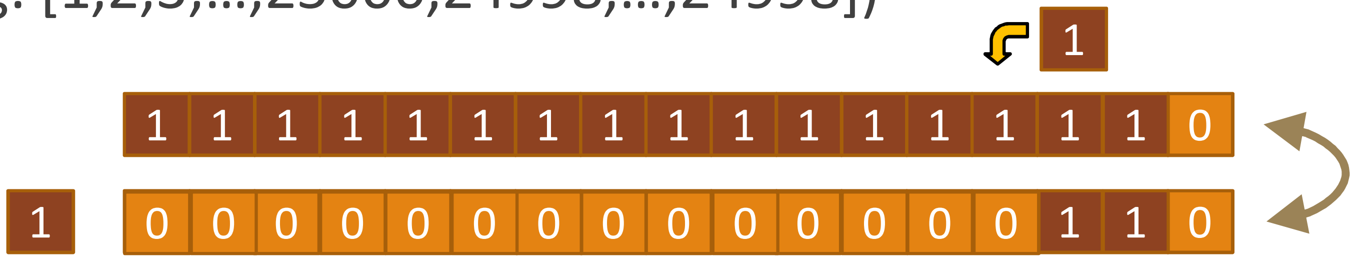
2. For $x_i > n$, we can place $[x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n]$ iff we can place $[x_1, \ldots, x_{i-1}, n, x_{i+1}, \ldots, x_n]$

   ⬄ We can use $\min(n, x_i)$ instead of x_i

Naïve implementations

Using a bit array to represent the sum

- $\frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^6}+\frac{1}{2^6} = \frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^5}$ = 0.11001 (in binary)
- A bit array of length n + 1, [0,1,1,0,0,1,…..] for 0.11001...
- Add $\frac{1}{2^{x_i}}$ and check if the result exceeds 1.0 or not
- *O*(n) for carry propagation (then cancelled) in the worst case (e.g. [1,2,3,…,25000,24998,…,24998])



- *O*(n^2) in total ➔ Too slow

Using BigDecimal or BigInteger (muliplied by 2^n)

- *O*(n^2) in total ➔ Too slow

Refinements

Recording the position of the first zero bit and the max number on medals placed

- If the number on a new medal is less than the position of the first zero and less than the max number on medals placed, you can tell, without computing the sum, that the medal cannot be placed
- When a new medal is placed, sum computation and updating of the first zero bit position is needed
- *O*(n) for one addition in the worst case, but as carry propagation clears consecutive one bits, the amortized total cost remains *O*(n)

Representing consecutive ones as its range

- Consecutive ones represented by a pair [start, end]
- Update is in *O*(log n); Total cost is *O*(n log n)

H: Animal Companion in Maze

Problem Summary

Chase the monkey in a maze-like building

Repeat moving to an adjacent room randomly

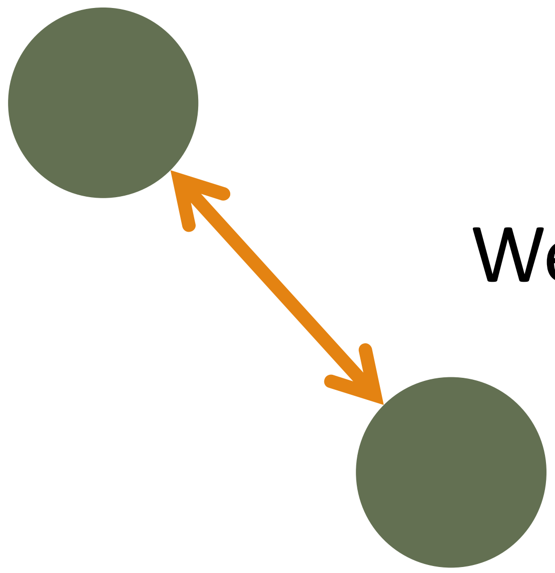
Compute the longest time before he will be confined
(He may have possibilities not to be confined)

- # of rooms: $2 \leq n \leq 100000$
- # of doors: $1 \leq m \leq 100000$
- Each door is one-way or two-way

Compute the length of the longest path in the graph, or find a cycle in the graph

When the graph has no cycle

The graph is **Directed ~~Acyclic~~ Graph (DAG)**



We don’t consider this as a cycle

When the graph has no cycle

The graph is **almost a DAG**

↓

After decomposing the graph into
Strongly-Connected Components,
each SCC is a **tree** with two-way edges only

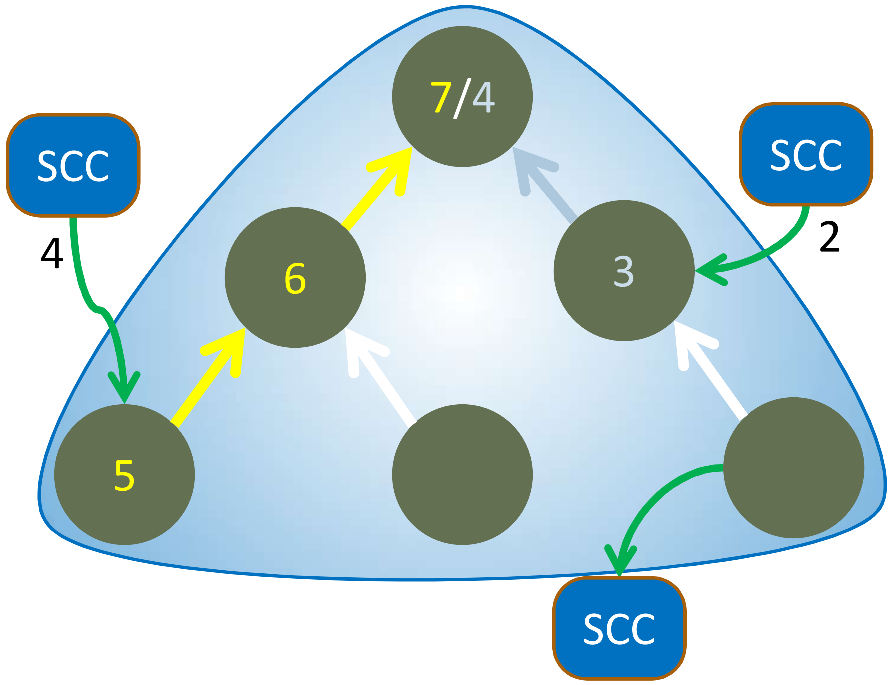
↓

We can compute the longest distance using **DP**

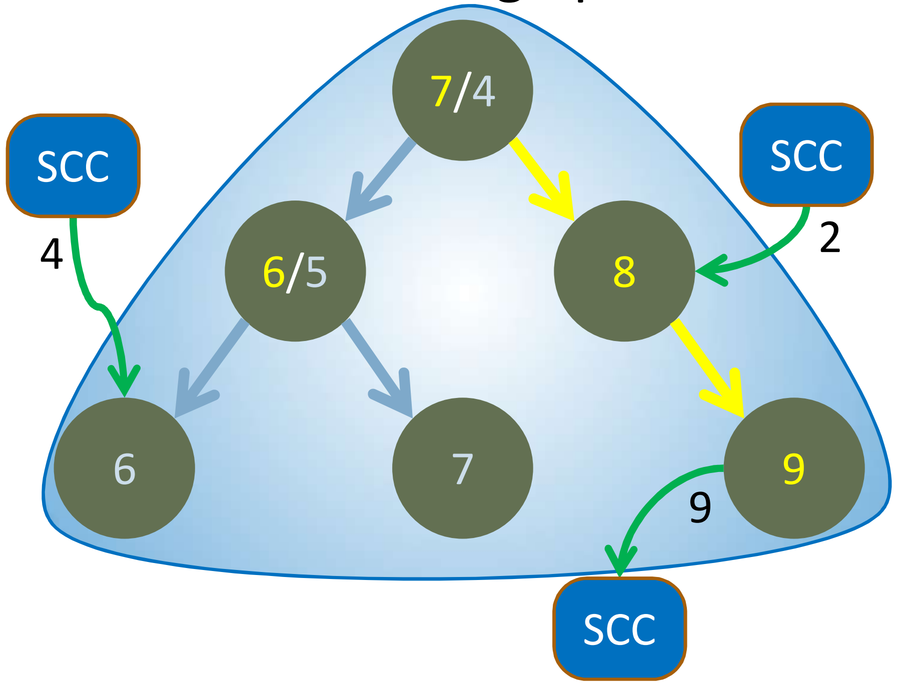
Path lengths through SCCs

The longest distance through a strongly-connected component can be computed with **two DFSs**

1. Take the max. of longest distance through children



2. Propagate the longest distance through parent



How to find a cycle

Easily detected by **the decomposition to SCCs**

If a one-way edge is in a SCC, it is in a cycle

If # of edges is more than or equal to
# of vertices in a SCC, it has a cycle

Otherwise no cycle exists

Summary

- Decompose into SCCs
- Detect a cycle if exists
- Otherwise compute the longest distance by DP
- The time complexity is $O(n + m)$

I: Skinny Polygon

Problem Summary

You are given two integers x_{bb} and y_{bb} ($<=10^9$). Find one simple polygon such that:

Its bounding box is $[0, x_{bb}] \times [0, y_{bb}]$.

The number of vertices is 3 or 4.

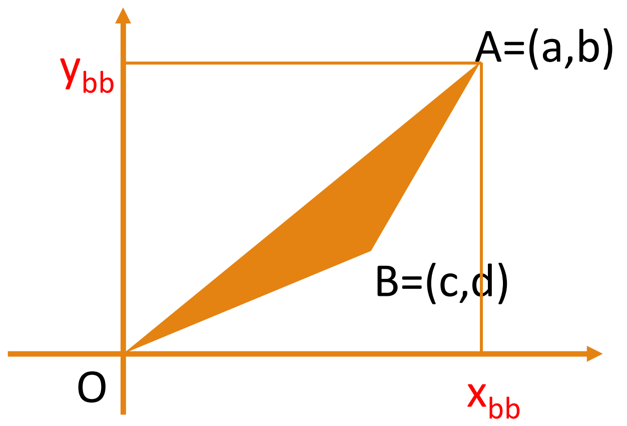
Its area is <= 25000.

NOTE: This is quite smaller than x_{bb} and y_{bb}.

When x_{bb} and y_{bb} are relatively prime

Let's start from easy cases.

The area of triangle △OAB is |ad-bc|/2.



Suppose that $a=x_{bb}$ and $b=y_{bb}$.

If xbb and ybb are relatively prime, there exists c and d s.t. |ad-bc|=1.

You can find such c and d using extended Euclidean algorithm.

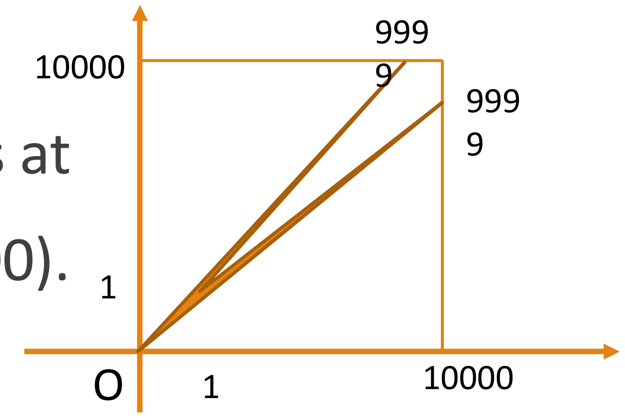
When x_{bb}=y_{bb}

What if x_{bb} and y_{bb} are not relatively prime?
Say, how about the case when x_{bb} = y_{bb}=10000?

In this case, the area of the quadrangle

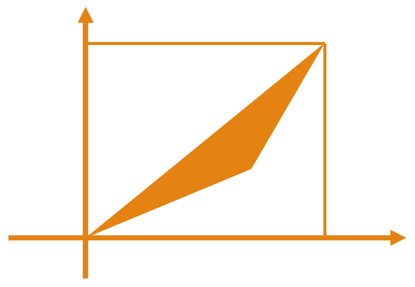
in the figure to the right is 1, with vertices at

(0,0), (10000,9999), (1,1), and (9999,10000).

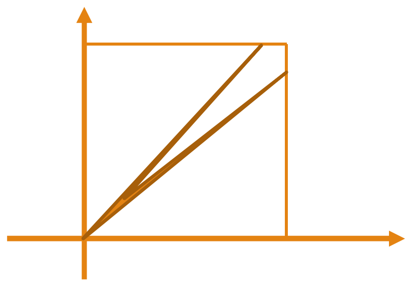


In general

Let $g = GCD(x_{bb}, y_{bb})$. We have two candidates.



There exists c and d s.t. $|ad-bc|=g$.
The area is $g/2$.



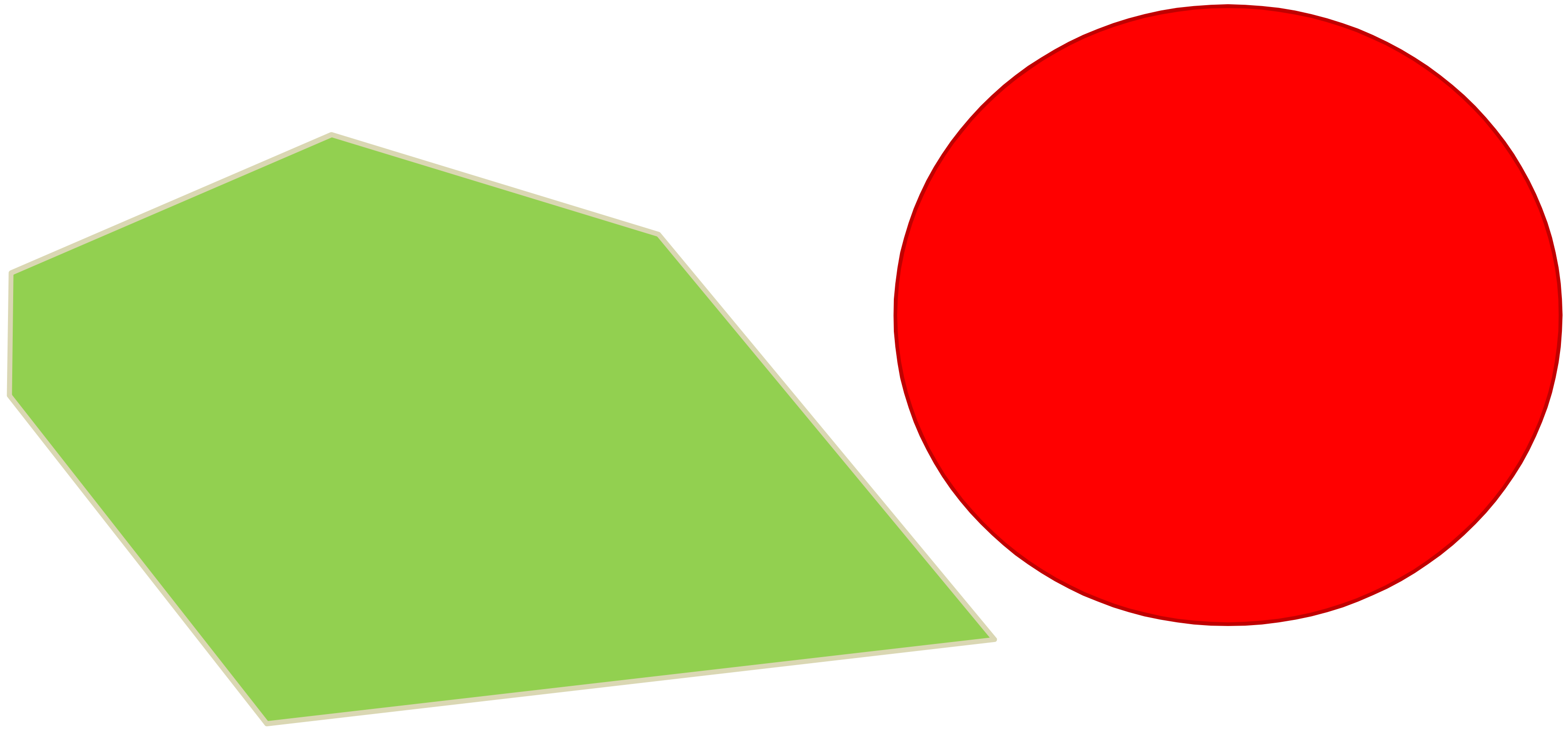
Put a vertex to $(x_{bb}/g, y_{bb}/g)$.
The area is $(x_{bb}/g+y_{bb}/g)/2$.

Choose smaller one. The area is at most

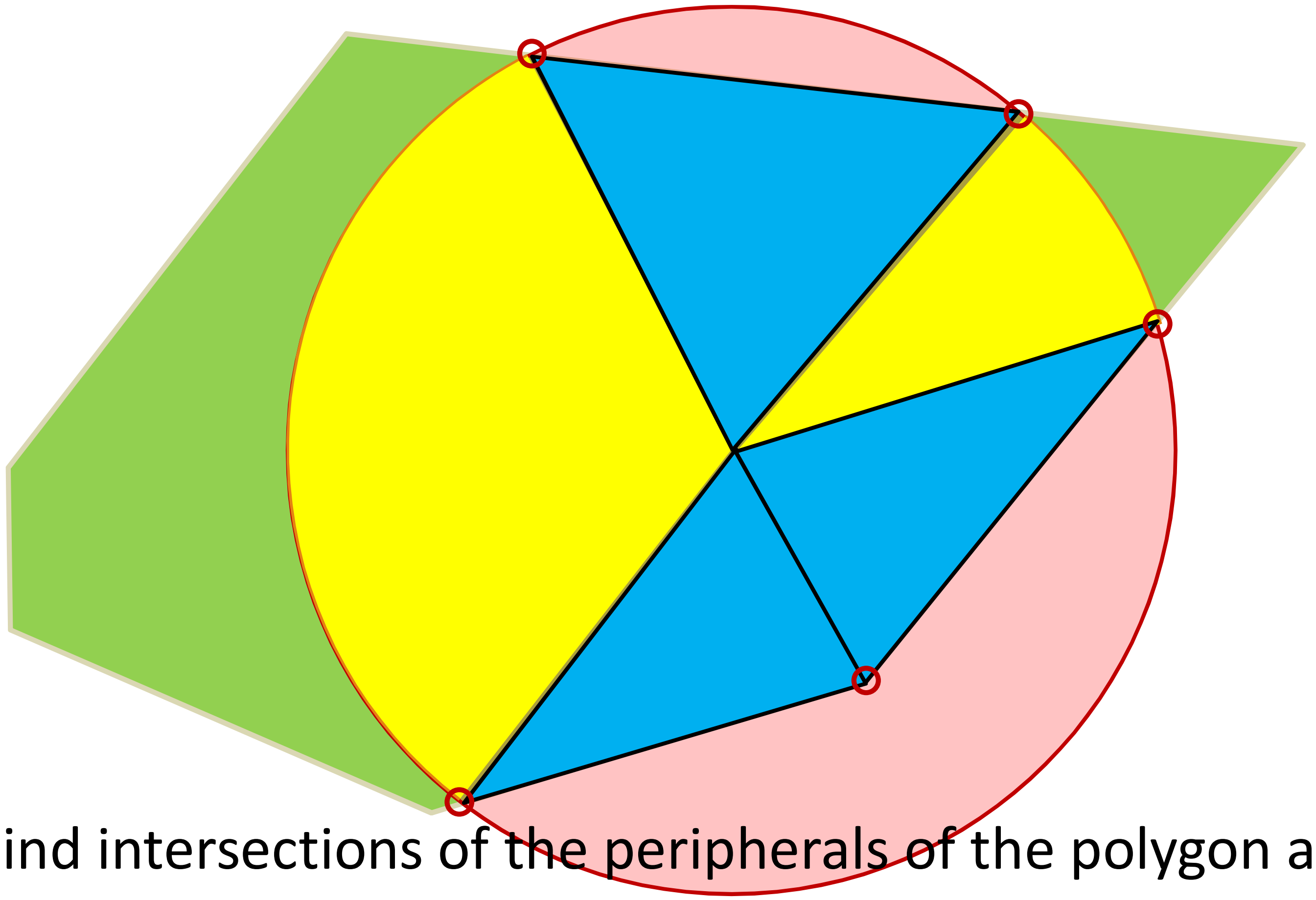
$$\sqrt{(g/2 * (x_{bb}/g+y_{bb}/g)/2)} = \sqrt{(x_{bb} + y_{bb})}/2 <= 25000.$$

Computational complexity is logarithmic to x_{bb} and y_{bb}. (very quick)

J: Cover the Polygon with Your Disk



Calculating Intersection Area Size



1. Find intersections of the peripherals of the polygon and the disk
2. Find polygon vertices covered by the disk
3. Sum up the area sizes of the fan shapes and triangles

Finding the Disk Position Giving the Max Area

Several efficient algorithms are known to find the maximum of unimodal functions in 2-D spaces

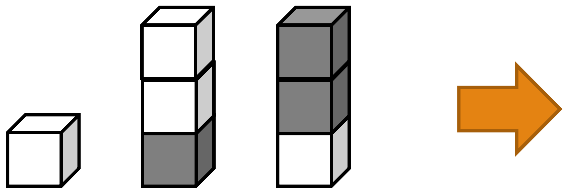
- Steepest descent
- Downhill simplex (Nelder–Mead)
- Quasi-Newton (BFGS)
- Evolution strategy (CMA-ES)

As the time limitation is not so severe, any of the methods listed above are OK

Problem K:
Black and White Boxes

The Game of Black and White Boxes

- Given piles of black and white boxes
- Two players: Alice and Bob
- First player is decided by a fair random draw
- Alice selects a black box and removes the box with the above
- Bob selects a white box and remove the box with the above
- If no box to remove is left, one loses the game



Alice (black) plays first ➔ Bob wins
Bob (white) plays first ➔ Bob wins

Four Possibilities

The game is a perfect information game

➔ The winner is determined by the configuration & the first player

1. Alice First ➔ Alice Wins & Bob First ➔ Alice Wins: **Alice-Wins**
2. Alice First ➔ Alice Wins & Bob First ➔ Bob Wins: **First Player-Wins**
3. Alice First ➔ Bob Wins & Bob First ➔ Alice Wins: **Second Player-Wins**
4. Alice First ➔ Bob Wins & Bob First ➔ Bob Wins: **Bob-Wins**

Configuration is Fair ⇔ First Player-Wins or Second Player-Wins

Problem K

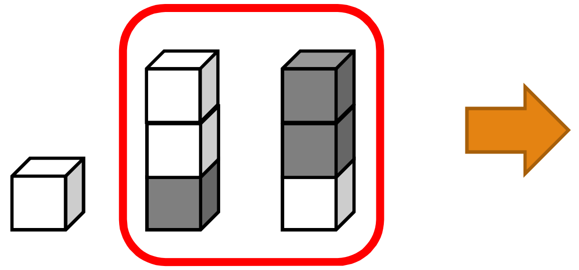
Given a candidate set of piles

Pick a number of piles to arrange an initial configuration:

- The configuration is fair
- The number of boxes is maximized.

Constraints:

- The number of piles ≤ 40
- The size of each pile $|p| \leq 40$

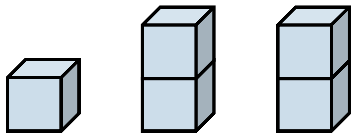


Alice (black) plays first ➔ Bob-Wins
Bob (white) plays first ➔ Alice-Wins

➔ Fair configuration of size 7

Colorless version = Nim

If there are no color, this is the Nim



The winner of the Nim is computed by bit-wise XOR
(01 xor 10 xor 10 = 01 ≠ 0 ➔ second player wins)

We have to seek a similar relation
for our colored problem!

Idea

1. If we have a number $s(i)$ for each pile i such that $\sum_{i \in X} s(i) = 0$ iff configuration X is fair,
2. the problem is solved by the subset sum problem:

$$\text{maximize} \sum_{i \in X} l(i) \ \text{ subject to} \sum_{i \in X} s(i) = 0,$$

3. which can be solved in $O(2^{n/2})$ time by enumerating the half of candidates ➔ OK for $n \le 40$

Question: Does there exist such number $s(i)$?
How to compute it?

Play with Small Examples 1

■ +1 (black wins)

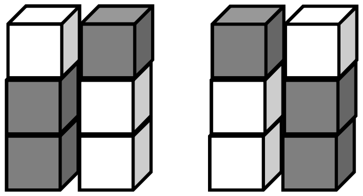
□ -1 (white wins)

(empty) = 0 (second player wins)

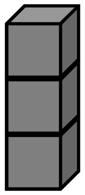
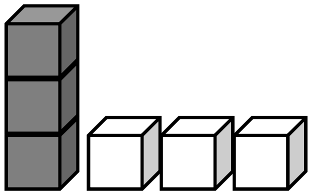
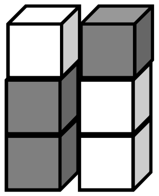
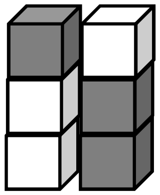
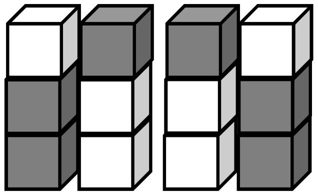
■□ 0 since it is fair

In general, alternate color ➔ alternate sign

(Proof: strategy stealing)

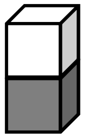
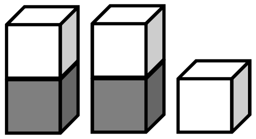
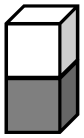
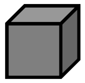
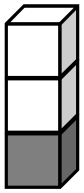
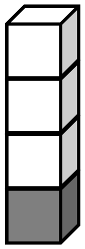


Play with Small Examples 2

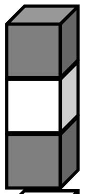
-  = +3 because  is fair
- If  = x then  = -x because  is fair

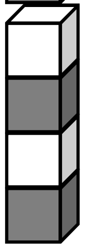
(proof by strategy stealing;
if the first player plays the first part,
the second player plays the alternate second part)

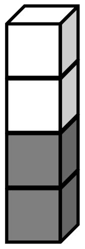
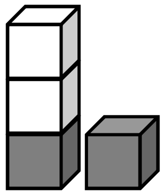
Play with Small Examples 3

-  … Black wins, but, how significantly black wins?
-  is fair ➔  = 1/2 (i.e., weaker than )
- Similarly,  = 1/4,  = 1/8, …,

Play with Small Examples 4

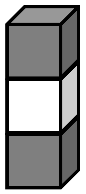

= 3/4

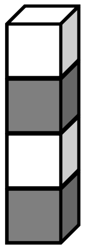

= 5/8

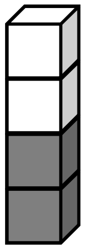
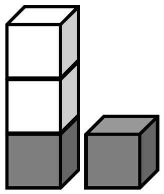

=

= 5/4

You have to guess the formula from the examples!

Solution

 $= 1 - 1/2 - 1/4 = 3/4$

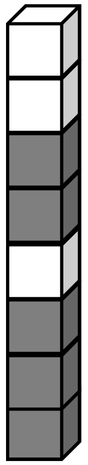
 $= 1 - 1/2 + 1/4 - 1/8 = 5/8$

 $=$  $= 1 - 1/2 - 1/4 + 1 = 5/4$

Formula

Add ±1 for each contiguous black/white box from the bottom

Then add ±1/2, ±1/4, ±1/8, … for each black/white box



$= 1 + 1 + 1 – 1/2 + 1/4 + 1/8 – 1/16 – 1/32 = 89/32$

number of boxes, number of piles $\leqq$ 40
➔ |numbers| are in $[2^{-40}, 2^{11}]$ ➔ represented in **double** type or **long long** type multiplied by 2^{40}

Conclusion of Analysis

The configuration can be mapped to a real number $s(i)$ of the form $a/2^b$ such that

- $s(i) > 0$ ➔ Alice wins
- $s(i) < 0$ ➔ Bob wins
- $s(i) = 0$ ➔ Second player wins

i.e., **This game has no "First Player-Wins" configurations**

The union of two states i, j is mapped to $s(i) + s(j)$

By computing this number, the problem is easily solved by Subset-Sum

Take Home Message:
General Theory for Two Player Game

If a game has no “First Player-Wins” configurations, the configuration of the game can be mapped to real numbers $s(i)$ of the form $a/2^b$ such that

- $s(i) > 0$ ➔ Alice-Wins
- $s(i) < 0$ ➔ Bob-Wins
- $s(i) = 0$ ➔ Second Player-Wins

The union of two states i, j is mapped to $s(i) + s(j)$

Such numbers are called the **Surreal Numbers**

If you are interested in,
read Conway: “On Numbers And Games”