香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Constructive Algorithms & Special Tasks (II)

Ethen Yuen {ethening}

2026-03-14

Outline

1. Communication Tasks
2. Output-only Tasks
3. Minicomp

Communication Task

Communication task (Two-steps task)

- Generally, you have to write two subprograms (or two modes) to collaboratively solve a problem.

- Judging flow:
  - [source input] → [program mode A] → [output A]
  - [input based on output A] → [program mode B] → [final output]

- The *means of communication* are limited in some way.
  - Limited in amount of data sent
  - Data may be reordered (e.g. IOI 2011 - Parrots)
  - Data may be corrupted (e.g. IOI 2019 Practice - Data Transfer)
  - Data must be in some specified form (e.g. M2332 - Collaborative Sudoku)

Communication task (Two-steps task)

- Typical Program mode A
  - Encodes or processes the original input into a concise Output A
  - **Aim:** build up a strategy that can transfer more information with shorter length
- Typical Program mode B
  - Interprets Output A' to produce the final result
  - **Aim:** build up a strategy to interpret the [output A] and extract some useful data

- Typical Scoring
  - The size of Output A or the correctness of Output B influences your final score

Communication task

- It is not necessary to have only 2 steps in a communication task.
  - e.g. APIO 2022 - Mars is a N-step communication task
  - The main idea is still the same, focusing on the data to be storing and transferring.

- In general, we could classify communication tasks into 2 types:
  - Encoding - **A** encode a message in some way -> send -> **B** decode it in some way
  - Labelling - **A** label a grid / graph structure -> **B** use those labels to navigate / answer some questions

- Generally a lot of fun to do communication tasks
  - Improve your thinking
  - A lot of real-life applications

IOI 2019 Practice - Data Transfer

Warm-up Question:

- **Alice** has **N** bits of data, she should attach **K** bits on the end and send to Bob
- At most one bit of the **(N + K)** bits might be corrupted
- **Bob** needs to recover the original **N** bits.

- **N = 255**
- **K <= 9** to get full marks.

- This task is actually testing you to come up with some kind of Error Correction Code (Hamming Code) in Information Theory.

IOI 2019 Practice - Data Transfer

A simple and brute-force solution: adding **redundancy**

K = N: Sending a copy of the given **N** bits, then we can use these **K** bits to check if the first **N** bits are corrupted.

- What if the last **K** bits are corrupted? How can we know whether the error happens in the data part or the attachment part? We can't.

K = 2N: Sending 2 copies of the given **N** bits. Resolves error by majority voting

- **K** is way too large to score any significant marks, we need a cleverer way.

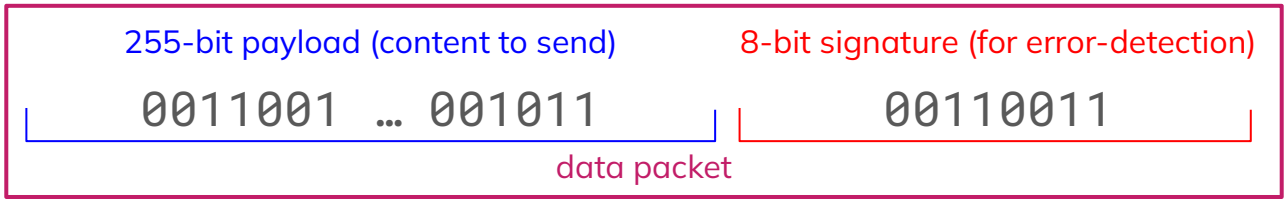
IOI 2019 Practice - Data Transfer

- **Notice that N = 255 = (2^8 - 1)**
- We can set up a 8-bit error detection scheme as follows:
  - Start with a 8-bit attachment **`att = 0`**. We construct the attachment by XOR-summing the 255-bit data like the code on the top right.

- If the **XOR-sum of the new data** do not match the old one, we can recover the corrupted position by XORing them.

```
int att = 0;
for (int i = 1; i <= 255; i++) {
  if (data[i]) {
    att ^= i;
  }
}
==============================
int natt = 0;
for (int i = 1; i <= 255; i++) {
  if (ndata[i]) {
    natt ^= i;
  }
}
int corrupted = (natt ^ att);
```

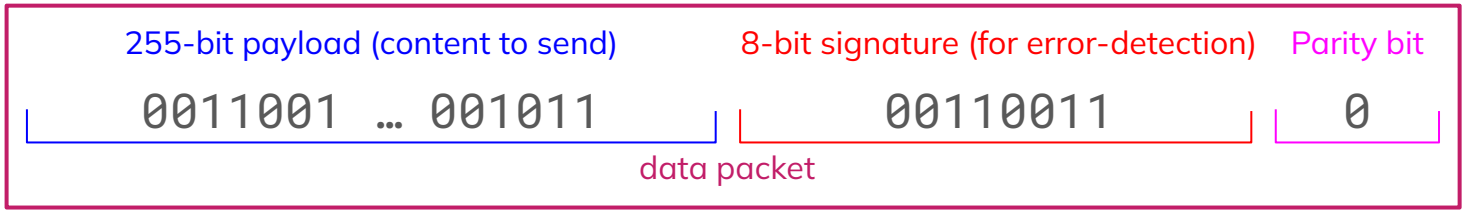
IOI 2019 Practice - Data Transfer



- For the received packet:
  - If XOR-sum of payload == signature: NO corruption (note: at most 1-bit of corruption in this task)
  - Else: we can detect the corruption in the payload with the signature!
- Any problem?
- The signature is still a part of the data packet → It can get corrupted too! What should we do?

IOI 2019 Practice - Data Transfer

- **K <= 9** to get full marks.
- We can add one more bit, denoting the parity of the signature.



- For the received packet:
  - If parity of signature != parity bit: Corruption with signature or parity, data is not corrupted
  - Else if XOR-sum of payload == signature: NO corruption
  - Else: we can detect the corruption in the payload with the signature!

IOI 2019 Practice - Data Transfer

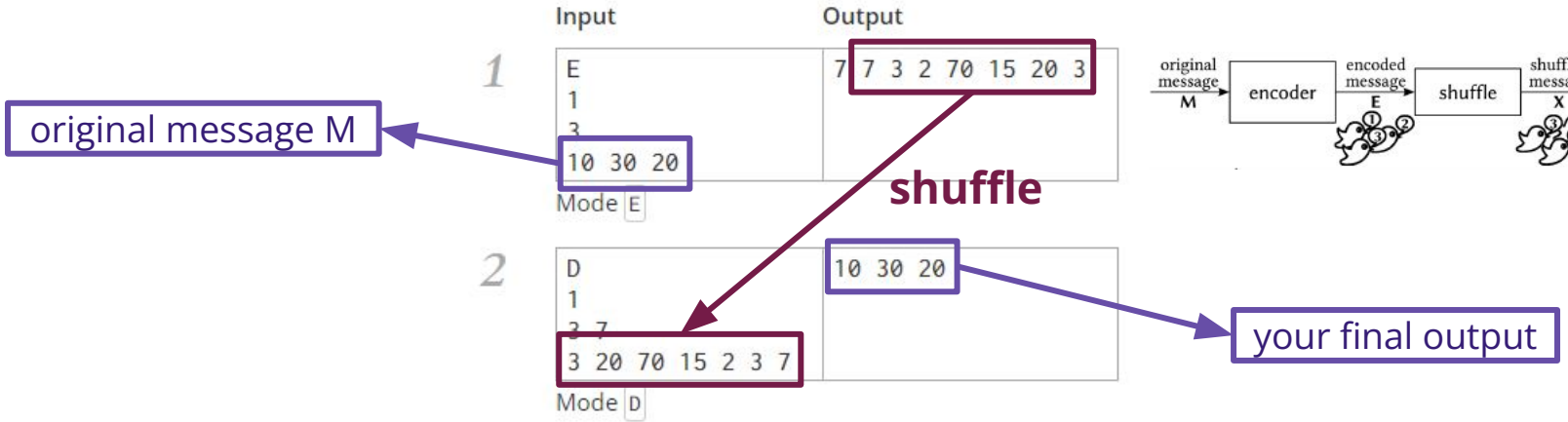
- This is a typical communication task: focusing on “How to encode bits with minimal overhead”.

- **Takeaway:** Considering in binary base (or other bases) is a important technique in Communication Task. You will see this over and over again.

I1123 Parrots

Problem:

- Message **M** consists of **N** (<= 64) integers within [0, 255] (not necessarily distinct).
- You can send **K** integers but they would be shuffled. Recover **M**.



I1123 Parrots

SUBTASKS

	Points	Constraints
1	17	$N = 8$, and each integer in the array M is either 0 or 1. Each encoded integer must be in the range from 0 to $R = 65535$, inclusive. The number of birds you may use is $K = 10 \times N$.
2	17	$1 \leq N \leq 16$ Each encoded integer must be in the range from 0 to $R = 65535$, inclusive. The number of birds you may use is $K = 10 \times N$.
3	18	$1 \leq N \leq 16$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 10 \times N$.
4	29	$1 \leq N \leq 32$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 10 \times N$.
5	19	$16 \leq N \leq 64$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 15 \times N$.

Partial score available
K = 5N for full marks

I1123 Parrots

Adapted From IOI 2011 Solutions

1	17	$N = 8$, and each integer in the array M is either 0 or 1. Each encoded integer must be in the range from 0 to $R = 65535$, inclusive. The number of birds you may use is $K = 10 \times N$.

I1123 Parrots

Adapted From IOI 2011 Solutions

1	17	$N = 8$, and each integer in the array M is either 0 or 1. Each encoded integer must be in the range from 0 to $R = 65535$, inclusive. The number of birds you may use is $K = 10 \times N$.

- Instead of sending the message **M**, send the positions of 1.

I1123 Parrots

Adapted From IOI 2011 Solutions

2 17 $1 \leq N \leq 16$
Each encoded integer must be in the range from 0 to $R = 65535$, inclusive.
The number of birds you may use is $K = 10 \times N$.

I1123 Parrots

Adapted From IOI 2011 Solutions

2 17 $1 \leq N \leq 16$
Each encoded integer must be in the range from 0 to $R = 65535$, inclusive.
The number of birds you may use is $K = 10 \times N$.

- Numbers in message ranged from 0-255 (8 bits)
- Numbers you sent ranged from 0-65535 (16 bits)
- Because the numbers you can output have a larger range, we can incorporate **positional information** and **data information** into the same encoded number: **256 * (position) + data**

I1123 Parrots

Adapted From IOI 2011 Solutions

3	18	$1 \leq N \leq 16$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 10 \times N$.

I1123 Parrots

Adapted From IOI 2011 Solutions

3 18 $1 \leq N \leq 16$
Each encoded integer must be in the range from 0 to $R = 255$, inclusive.
The number of birds you may use is $K = 10 \times N$.

- We do not have extra bits to store position, but we can utilize the extra numbers that we sent. We can send each bit of each integer one by one:

Position of the element (0 ~ 15) 4 bits	Position of the bit (0 ~ 7) 3 bits	Bit value (0 / 1) 1 bit

- **K = 8N**

I1123 Parrots

Adapted From IOI 2011 Solutions

4	29	$1 \leq N \leq 32$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 10 \times N$.

I1123 Parrots

Adapted From IOI 2011 Solutions

4	29	$1 \leq N \leq 32$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 10 \times N$.

- We need 1 more bit to represent the position,
- Inspired by Subtask 1, we do not need to send the bit value if we only send position of “1”.

Position of the element (0 ~ 31) 5 bits	Position of the bit (0 ~ 7) 3 bits	~~Bit value (0 / 1)~~ ~~1 bit~~

- **K = 8N**

I1123 Parrots

Adapted From IOI 2011 Solutions

5	19	$16 \leq N \leq 64$ Each encoded integer must be in the range from 0 to $R = 255$, inclusive. The number of birds you may use is $K = 15 \times N$.

I1123 Parrots

Adapted From IOI 2011 Solutions

5 19 $16 \leq N \leq 64$
Each encoded integer must be in the range from 0 to $R = 255$, inclusive.
The number of birds you may use is $K = 15 \times N$.

- **K = 12N** solution
- There are total of **64** (Number pos) *** 8** (Bit pos) **= 512** different bit position
- A single encoded integer can't even represent a position fully.

- Group the bits into pairs -> only **256** pairs of bits with **4** value **(00,01,10,11)**
  - Send the position **x** times, if the bit value is **x (0, 1, 2, 3)**.

I1123 Parrots

Adapted From IOI 2011 Solutions

5 19 $16 \leq N \leq 64$
Each encoded integer must be in the range from 0 to $R = 255$, inclusive.
The number of birds you may use is $K = 15 \times N$.

- **K = 6.25N** solution
- Group the bits into pairs -> only **256** pairs of bits with **4** value **(00,01,10,11)**
  - Send the position **x** times, if the bit value is **x (0, 1, 2, 3)**.
  - This encoding scheme is costly when all the bits are **1**, and no cost when all the bits are **0**
- Consider the opposite scheme, send the position **3-x** times for value **x**
  - Use the better scheme out of the 2
- Signify the decoder which scheme is used by sending **4** copies of **255**.

I1123 Parrots

Adapted From IOI 2011 Solutions

5 19 $16 \leq N \leq 64$
Each encoded integer must be in the range from 0 to $R = 255$, inclusive.
The number of birds you may use is $K = 15 \times N$.

- **K = 4.08N** solution (Optimal Solution)
- What is the number of unordered sequence of 261 8-bit integers?
  - **(256+261) C (256) ≈ 1.47 * 10^154**
  - Same as the number of solution for **x_0+x_1+...+x_255 <= 261**
- What is the number of ordered sequence of 64 8-bit integers?
  - **256^64 ≈ 1.34 * 10^154**
- You can perform a mapping from the original message to the encoded sequence.

I1123 Parrots

- While brute-force exhausting every possibilities of information to send is a valid solution for **I1123 Parrots** (and some other communication task), the mapping is not easy to implement.
- Tasks nowadays also prevent you from easily doing so.
  - **M2332 Collaborative Sudoku** where you can send only valid sudoku boards.
  - There are **6,670,903,752,021,072,936,960** (around 72 bits) of sudoku. But could you iterate through all of them or develop a efficient mapping in 1 sec machine time?

- **Takeaway**
- The key to communication is still: What is important to send & How to send with good efficiency

M2332 Collaborative Sudoku

Problem

- Alice and Bob are given some hints of Sudoku.
- Bob needs to solve the Sudoku with their hints combined.
- Alice's only method of communication is sending valid Sudoku grids to Bob.

M2332 Collaborative Sudoku

What data should Alice send?

- (x, y, value) tuples -> 9^3 different values
  - 9.509 bits info each, needing up to 81 cells
  - ~**770 bits in total**
- 10^{81} information of status of each cell (empty, or 1-9)
  - **270 bits in total**
- A valid solution board and the a string of 01 denoting which cell is hint
  - A board + **81 bits in total**
- Sometimes you don't need to send complete information, sending enough information to solve the task can reduce the cost.
  - Remove redundant hints in the sudoku first then send: can reduce to **<= 40 useful hints** at best https://math.stackexchange.com/questions/3274039/sudoku-maximal-minimum-number-of-starting-clues

M2332 Collaborative Sudoku

How to send the data?

- Hardcode C valid sudoku boards
  - **log_2(C) bits** per board transmitted
- Sending sudoku with arbitrary permutation on first row
  - 9! Information per board
  - **18.4 bits** per board transmitted
- Sending sudoku with arbitrary permutation on 3 diagonal region
  - $(9!)^3$ information per board
  - **55.4 bits** per board transmitted

3	5	9						
2	7	4						
6	1	8						
			8	3	6			
			2	5	1			
			4	9	7			
						2	9	5
						6	7	3
						8	1	4

M2332 Collaborative Sudoku

Combining the best idea in both sides gives us a 3-board solution.

- A valid solution board and the a string of 01 denoting which cell is hint
  - A board + **81 bits in total**
- Sending sudoku with arbitrary permutation on 3 diagonal region
  - $(9!)^3$ information per board
  - **55.4 bits** per board transmitted

This example shows how hard constraints (like only valid Sudoku boards allowed) force us to embed data in creative ways.

More Practice Tasks on encoding

- CEOI 2014 D1Q3 - Question
- https://oj.uz/problem/view/CEOI14_question_grader(statement here: https://ceoi2014.informatik-olympiade.de/wp-content/uploads/2014/06/question.pdf)
  - There is a question with two choices, x (correct answer) and y (incorrect answer).
  - “A” knows the values of both x and y. (x, y ≤ 920)
  - “A” shouts a number to “B”, h.
  - “B” receives either x or y only. “B” should determine if it is the correct answer or not.
  - Score based on the maximum number shouted.

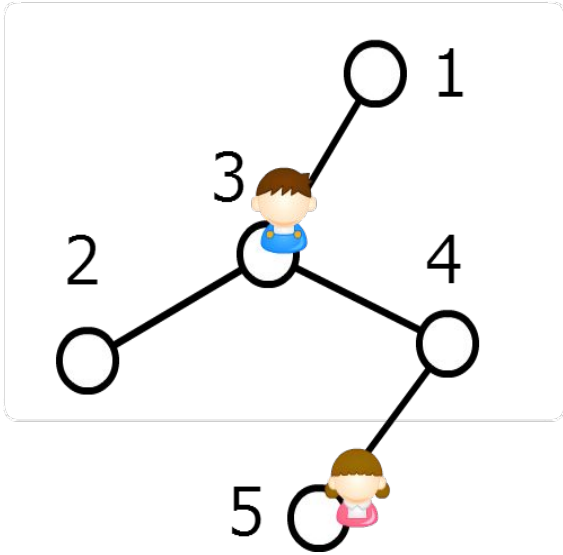
- BOI 2022 D2Q1 - Flight to the Ford (communication)
- https://oj.uz/problem/view/BOI22_communication
  - Encode a number N by sending 0 or 1.
  - Twist: everytime you send a signal, it may be corrupted (never corrupted twice in a row)
  - You know when it is corrupted and can adjust the signal sent afterwards.

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

Problem

- Given a tree with nodes numbered **1** to **N**.
- Anna is at node **T** and Bruno is at node **S**.
- Anna knows the whole tree structure and can label each node with an integer.
- Bruno only knows what the neighbours of the current nodes are, and their corresponding label. He must pick one step toward **T**. (or report if he is at **T**)

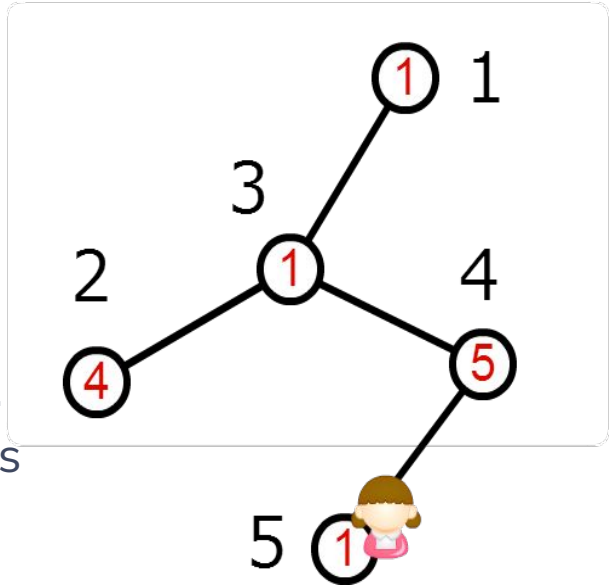


JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

Problem

- Given a tree with nodes numbered **1** to **N**.
- Anna is at node **T** and Bruno is at node **S**.
- Anna knows the whole tree structure and can label each node with an integer.
- Bruno only knows what the neighbours of the current nodes are, and their corresponding label. He must pick one step toward **T**. (or report if he is at **T**)

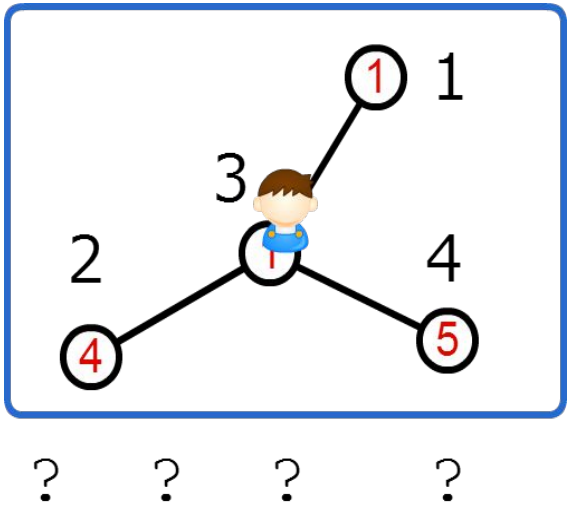


JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

Problem

- Given a tree with nodes numbered **1** to **N**.
- Anna is at node **T** and Bruno is at node **S**.
- Anna knows the whole tree structure and can label each node with an integer.
- Bruno only knows what the neighbours of the current nodes are, and their corresponding label. He must pick one step toward **T**. (or report if he is at **T**)

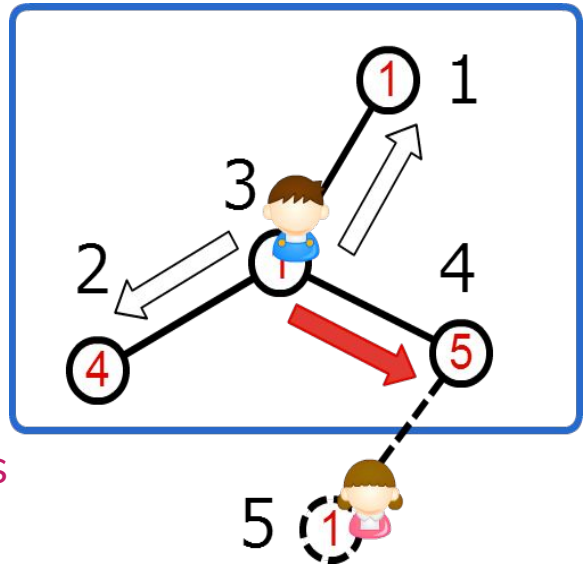


JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

Problem

- Given a tree with nodes numbered **1** to **N**.
- Anna is at node **T** and Bruno is at node **S**.
- Anna knows the whole tree structure and can label each node with an integer.
- Bruno only knows what the neighbours of the current nodes are, and their corresponding label. He must pick one step toward **T**. (or report if he is at **T**)



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

Subtask	Score	Constraints	Numbers written
1	10	--	0 ~ N
2	15	--	0 ~ 2
3	20	No island is directly connected to exactly two islands by bridges. S ≠ T	0 ~ 1
4	55	--	0 ~ 1

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 1:** Numbers written can be **0~N**

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

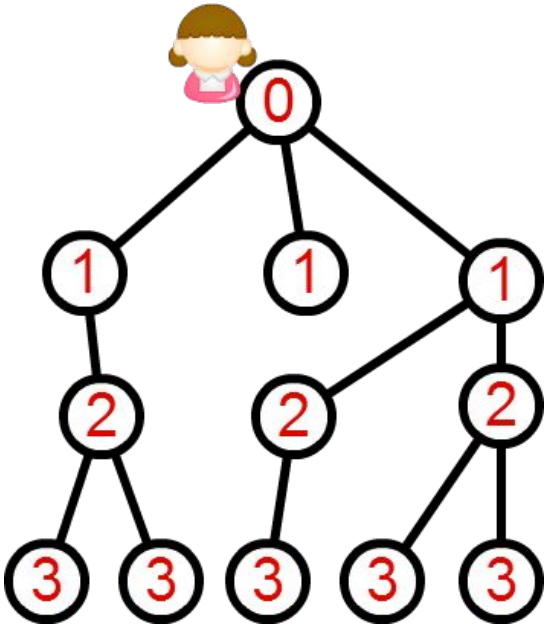
- **Subtask 1:** Numbers written can be **0~N**

Anna

- DFS from **T**.
- Write down the distances of all nodes from **T**.

Bruno

- Keep visiting the neighbour with a smaller number on it.
- Stop when reaches the node with number **0**.



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

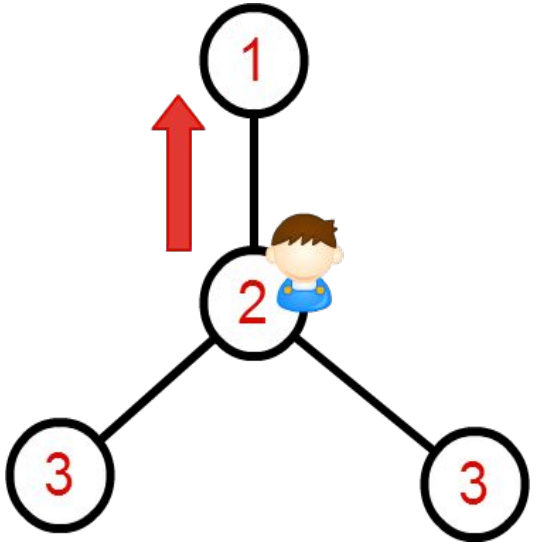
- **Subtask 1:** Numbers written can be **0~N**

Anna

- DFS from **T**.
- Write down the distances of all nodes from **T**.

Bruno

- Keep visiting the neighbour with a smaller number on it.
- Stop when reaches the node with number **0**.



JOI 2014/15 Spring Camp D3Q3 - Navigation

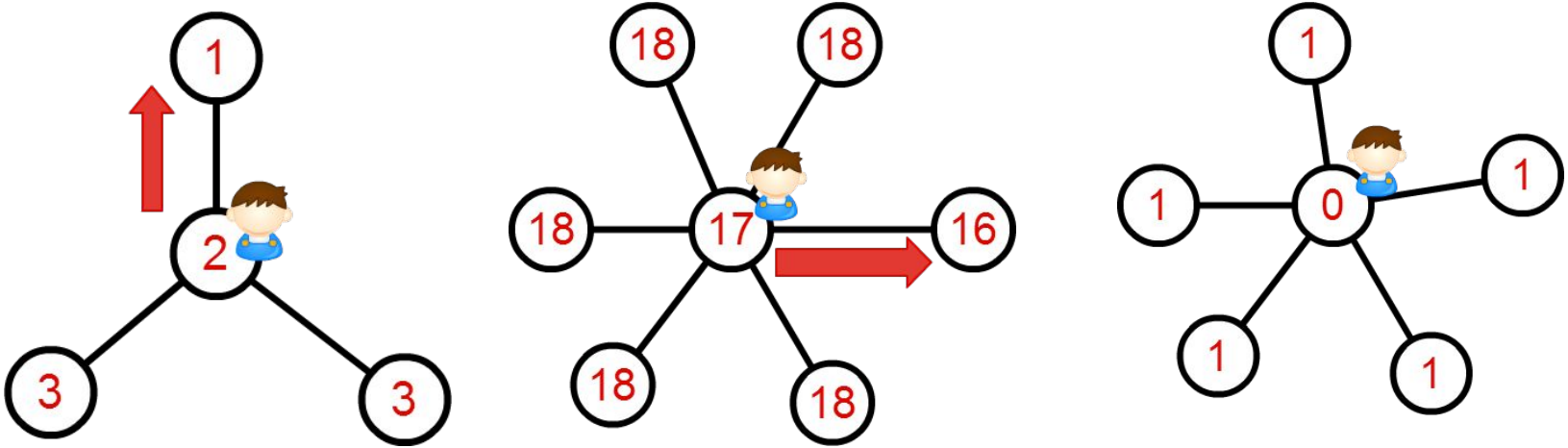
Adapted From JOI 2015 Solutions

- **Subtask 2:** Numbers written can be **0~2**

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 2:** Numbers written can be **0~2**
- In subtask 1, at any point of time, Bruno encounter at most 3 different numbers



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

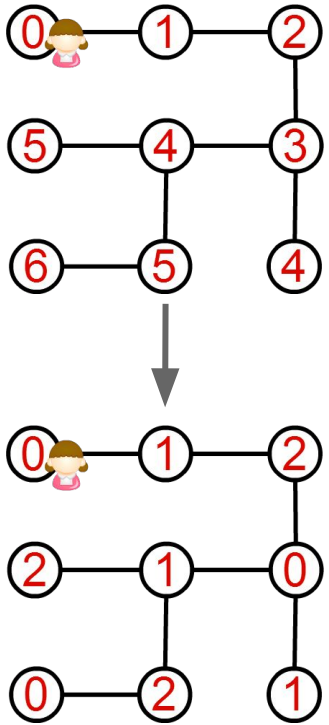
- **Subtask 2:** Numbers written can be **0~2**

Anna

- Write down distances **modulo 3**.

Bruno

- Current node 0 → Go to 2 (if exists)
  - Stop if not exists.
- Current node 1 → Go to 0
- Current node 2 → Go to 1



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

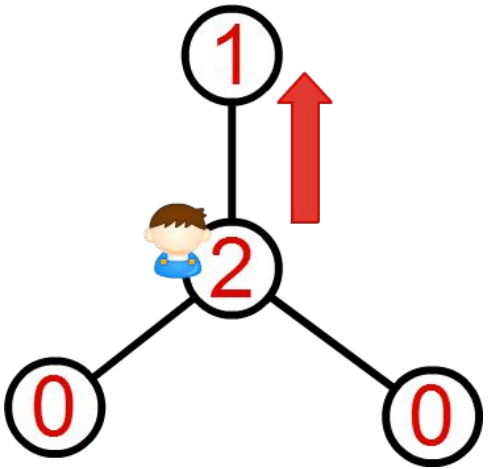
- **Subtask 2:** Numbers written can be **0~2**

Anna

- Write down distances **modulo 3**.

Bruno

- Current node 0 → Go to 2 (if exists)
  - Stop if not exists.
- Current node 1 → Go to 0
- Current node 2 → Go to 1



JOI 2014/15 Spring Camp D3Q3 - Navigation

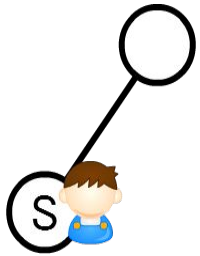
Adapted From JOI 2015 Solutions

- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.

JOI 2014/15 Spring Camp D3Q3 - Navigation

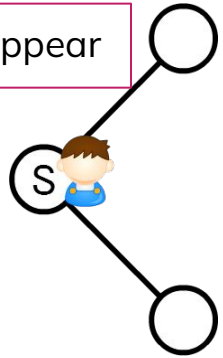
Adapted From JOI 2015 Solutions

- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- Node S can be divided into the following three types:

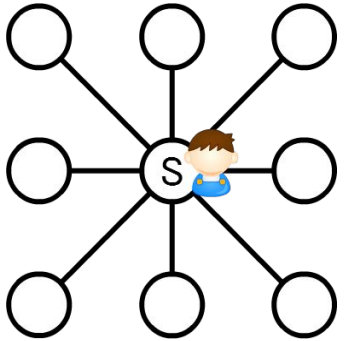


Degree = 1

Won't appear



Degree = 2

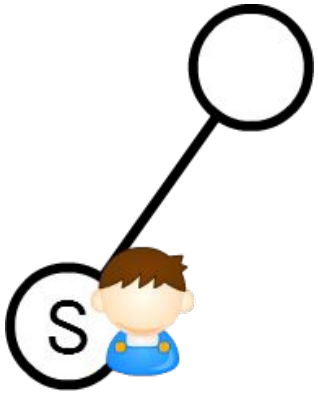


Degree > 2

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

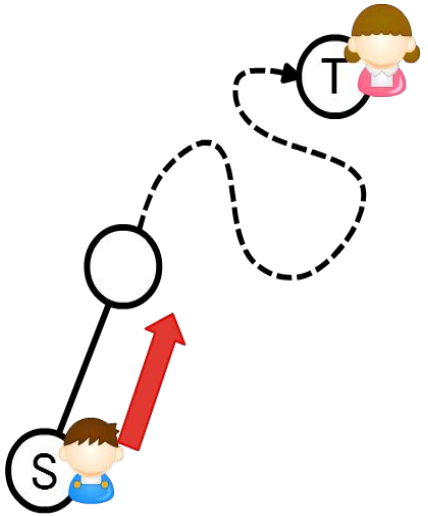
- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- **Case #1:** Degree = 1



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

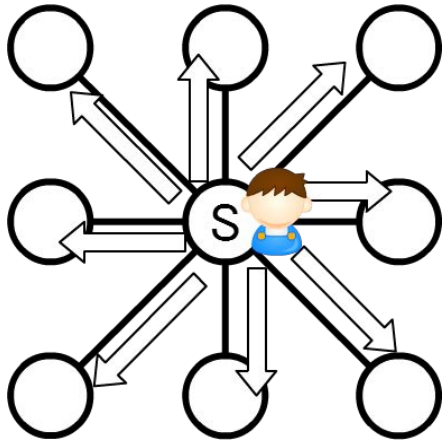
- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- **Case #1:** Degree = 1
  - In this subtask, **S ≠ T**.
  - Feel free to go!



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

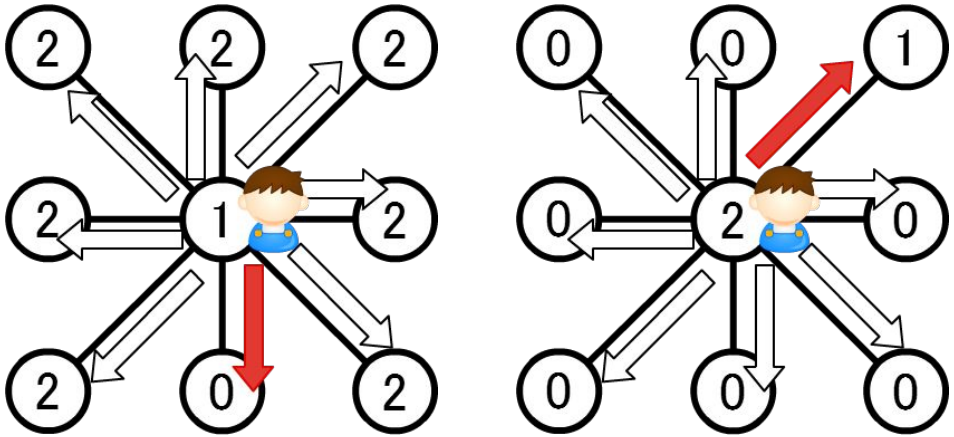
- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- **Case #3:** Degree > 2



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- **Case #3:** Degree > 2
  - Recall Subtask 2.

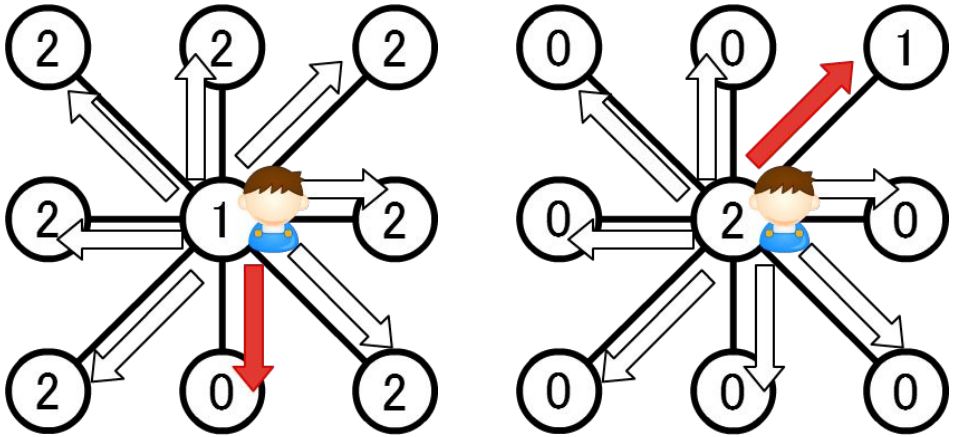


We can easily find the **only different** number in the neighbour.

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- **Case #3:** Degree > 2
  - If the distance from T to the correct exit is **x**, then the wrong exit is **x + 2**.



We can easily find the **only different** number in the neighbour.

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.
- **Case #3:** Degree > 2
  - If the distance from T to the correct exit is **x**, then the wrong exit is **x + 2**.
  - We can find a 01 string where **f(x) != f(x + 2)** to label the nodes:
  - One that works is: 0, 0, 1, 1, 0, 0, 1, 1, ...

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

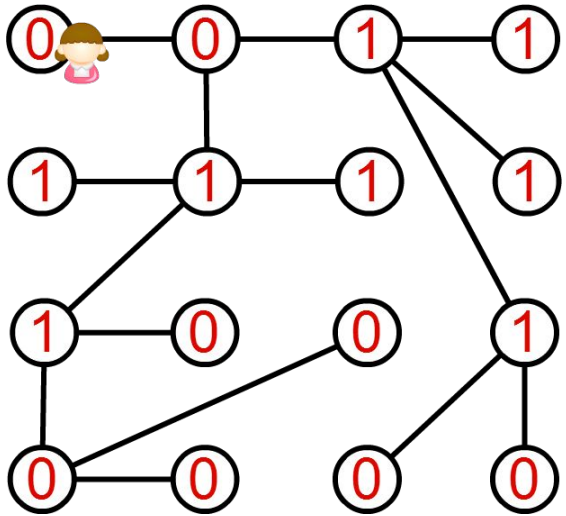
- **Subtask 3:** No island is directly connected to exactly two islands by bridges, **S ≠ T**.

Anna

- Write down 0, 0, 1, 1, 0, 0, 1, 1, ... according to distance from T.

Bruno

- If degree = 1, output the only neighbour.
- If degree > 2, output the only neighbour with a different number written.



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.
- There is too little information that Bruno gets: it is impossible to determine the answer with only 0,1.
- There must be some information we haven't used yet
  - Read the question carefully one more time

- Given a tree with nodes numbered **1** to **N**.
- Anna is at node **T** and Bruno is at node **S**.
- Anna knows the whole tree structure and can label each node with an integer.
- Bruno only knows what the neighbours of the current nodes are, and their corresponding label. He must pick one step toward **T**.

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.
- There is too little information that Bruno gets: it is impossible to determine the answer with only 0,1.
- There must be some information we haven't used yet
  - Read the question carefully one more time

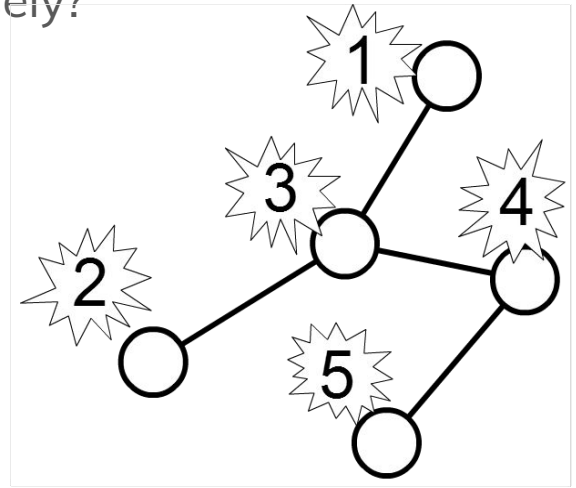
- Given a tree with nodes numbered **1** to **N**.
- Anna is at node **T** and Bruno is at node **S**.
- Anna knows the whole tree structure and can label each node with an integer.
- Bruno only knows what the neighbours of the current nodes are, and their corresponding label. He must pick one step toward **T**.

JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.
- How can we use the original IDs on the nodes wisely?

- Properties of original IDs:
  - Unique
  - That's all



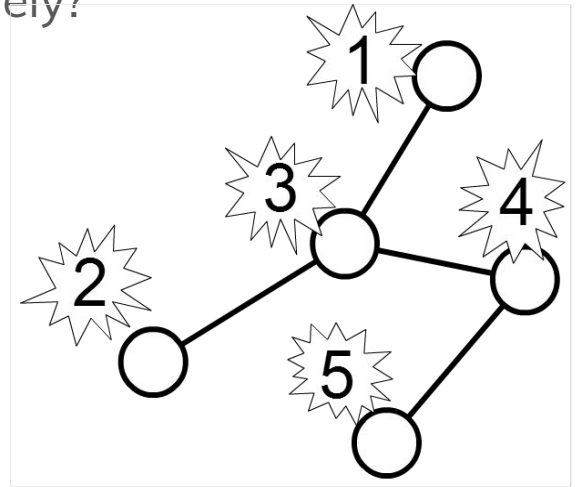
JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.
- How can we use the original IDs on the nodes wisely?

- Properties of original IDs:
  - Unique
  - That's all

- We can **compare the magnitudes** of the IDs to obtain useful information.



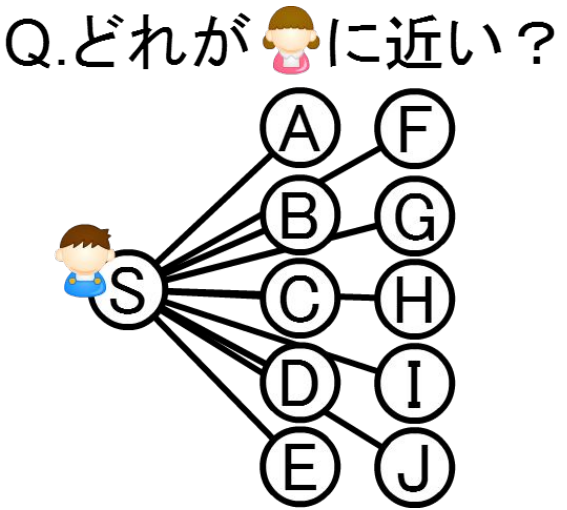
JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.

- **Original Approach:**
- (Too many possibilities)

"Which node is closer to Anna's home?"



JOI 2014/15 Spring Camp D3Q3 - Navigation

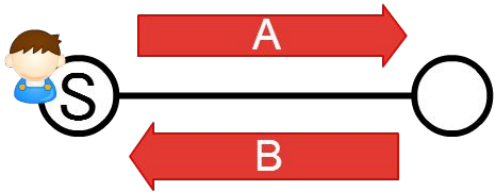
Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.

- **Modified Approach:**
- (Only 2 possibilities)

"Which direction should we go?"

Q.どっちに進むべき？



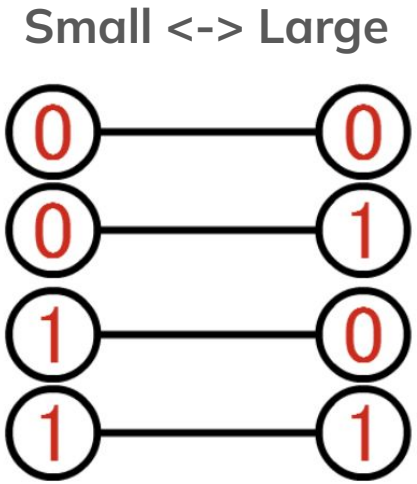
JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.

- **Modified Approach:**
- There are only two directions of travelling.
  → Only two possible answers.

- The information that characterizes the edge is
  - The IDs of the nodes.
  - The numbers written on the nodes.

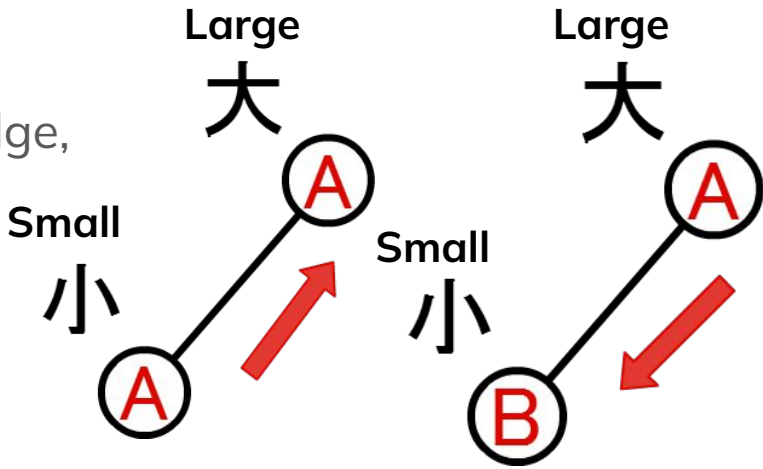


JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.

- **Modified Approach:**
- Let's number the nodes such that for each edge,
  - The travelling direction should be **"Small → Large"** if the numbers are **the same**.
  - The travelling direction should be **"Large → Small"** if the numbers are **different**.

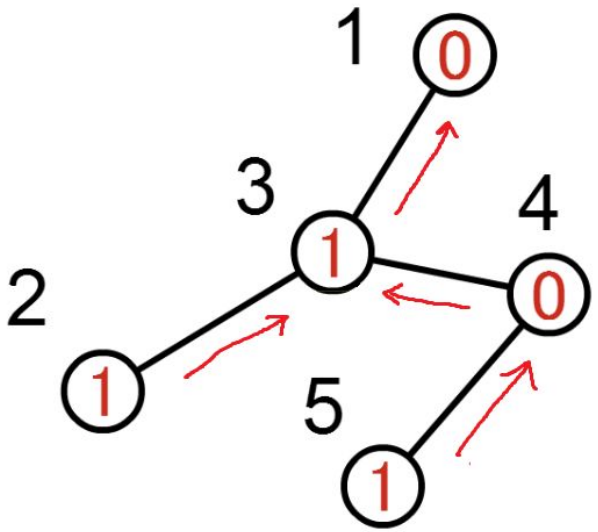


JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- **Subtask 4:** No additional constraints.

- **Modified Approach:**
- The numbering can be easily done with a DFS from T.



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

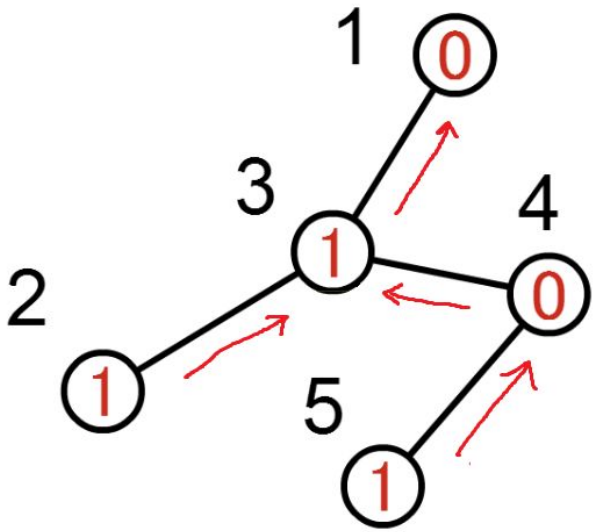
- **Subtask 4:** No additional constraints.

Anna

- DFS once from T and number the nodes by the method stated above.

Bruno

- Check each neighbouring edge and determine the correct direction of travel for each edge.
- If there exists some edge that should be travelled outwards, use that edge.



JOI 2014/15 Spring Camp D3Q3 - Navigation

Adapted From JOI 2015 Solutions

- While the observation needed for this task is difficult, the implementation is extremely light and no specific hard knowledge is needed.
- Anyone with sufficient graph knowledge and know how to do communication task could do.

More Practice Tasks on labelling graph

- JOI 2019/20 Spring Camp D3Q3 - Stray Cat
- https://oj.uz/problem/view/JOI20_stray
  - Similar settings with Navigation
  - This time you are required to label edges instead of nodes, and you are given some quotas to go a wrong direction.

- IOI 2020 Stations
- https://oj.uz/problem/view/IOI20_stations
  - Multiple queries on source and destination

Output-only Task

Output-only task

- Formal definition
  - Input files are given to you
  - You are not required to upload any source codes, just the output files
- Actual meaning
  - No need to worry about failing some unknown cases, all cases are revealed :D
  - No time limits / memory limits (actually there are... TL = 5hrs, ML = your machine)
  - You can even solve the cases manually :D :D :D
- P.S. Google Hash Code is an annual competition of output-only task (It was discontinued)

Output-only task

- Common stuffs?
  - not expecting optimal solutions (or not even exist)
  - some formulas to determine how good your outputs are (and how much you score)
  - good-enough solutions can get good-enough scores

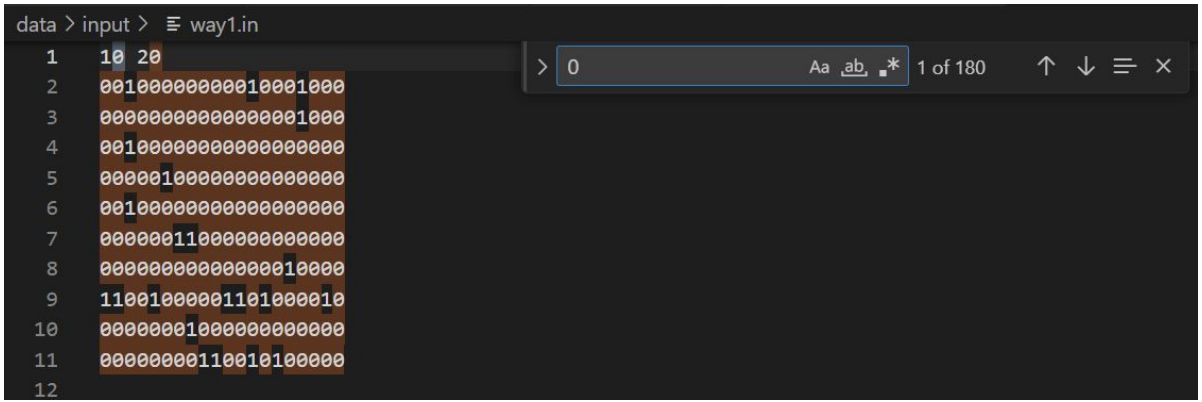
- What you can do?
  - Usually there exists some small cases (can be manually solved)
  - You can write programs to analyze the cases / solve the cases
  - You can even solve cases separately with different approach and codes

Test Cases

- The input files given is equally important to the problem statement.
  - **Not understanding them is like not reading the whole statement before doing a task.**
- Sometimes, there are “hidden” properties that do not reveal to you in the statement.
  - e.g. **IOI 2017 Nowruz**: the grid given to you is mostly empty
  - e.g. **IOI 2019 Broken Line**: in many cases the given points lies in a line
  - e.g. **M2454 The Great Awakening:** points are denser in the middle.
- Or the case itself would help you figure out some good solution
  - e.g. **M2334 A Way Out**: what is the optimal solution to solve a chain?
- You should visualize them to help you figure these out

Visualizer

- In some of the tasks, a visualizer program is given to you
  - Usually it is a Python script, and they will tell you how to run. **You should learn how to use it.**
- Sometimes, you will need to write your own visualizer.
  - If you have the skills to write Python visualizer on your own, that's great!
  - Sometime you may depend on the power of IDE to help you.
    (example from M2334 - A Way Out)



Scoring

- Output-only tasks is not usually seen in IOI, but from the few times it appears:
  - The average score is usually high
  - Similar level skill contestants' performances can be very diverse (like when some getting 30 marks while the other getting about 90)
- From contestants who get low score:
  - "I thought it was a hard question where I have already reached a limit, and my score could not be improved easily unless I put in a lot of time."
- From contestants who get high score:
  - "I just put some time in it and get pretty ok score so I keep on doing it."

E.g. **IOI 2017 Nowruz, IOI 2019 Broken Line**

Rank		Country	Sh	Sp	R	L	V	W
38	H	Hong Kong	100	40	100	39.84	81	33
46	M	Hong Kong	100	40	72	88.11	66	10
106	K	Hong Kong	65	40	37	82.95	52	24
132	L	Hong Kong	100	40	25	39.16	66	10

Rank		Country	N	W	T	P	S	B
28	Y	Hong Kong	87.90	20	37	98.15	51	50
64	T	Hong Kong	55.06	30	27	100.00	51	12
79	H	Hong Kong	29.26	100	16	20.00	13	70
201	K	Hong Kong	30.85	13	5	20.00	13	12

Scoring

- You are advised to attempt output-only task first within a problem set
  - You can utilize machine time to run solutions in the background when attempting other tasks later (passive income of marks)
  - More importantly, you may get a deeper understanding on the difficulty to get marks in it, so you can make better decision later.
- Something that you should do:
  - Write some obvious solutions first (like greedy, random, random + greedy, greedy + random)
  - Investigate how sensitive the scoring function is.

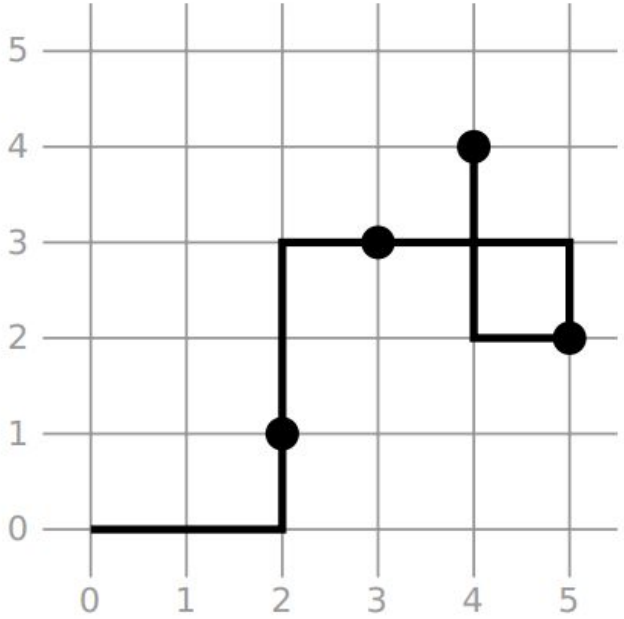
Grader

- You should be able to give a score to your answer locally.
  - In many output-only tasks, writing a grader is trivial.
  - e.g. **Travelling Salesman Problem**: Given N points in a 2D plane, find the shortest route to visit each city exactly once and return to the origin city, a NP-hard problem.
  - While finding the shortest route is hard, calculating the length of a given route is very easy.
- Having a quick local grader is crucial, so you can run random solution in your machine for 100000 times and submit the best one.
  - You can make it write the output in a file whenever you have a score increase.
  - Learn about File I/O, at least know how to use freopen, or input redirection with linux command

I1921 - Broken Line

Problem

- Given **N** points on a 2D plane.
- Construct a walk with minimum segments s.t.
  - It passes through all points.
  - It consists of vertical & horizontal lines only.
- Optimal solution: **N+3** segments.



I1921 - Broken Line

Visualizer is provide in this task -> Use it to understand each test case

In the attachments of this task, there is a script that allows you to visualize input and output files.

To visualize an input file, use the following command:

```
python vis.py [input file]
```

You can also visualize your solution for some input, using the following command. Due to technical limitations, the provided visualizer shows only **the first** 1000 **segments** of the output file.

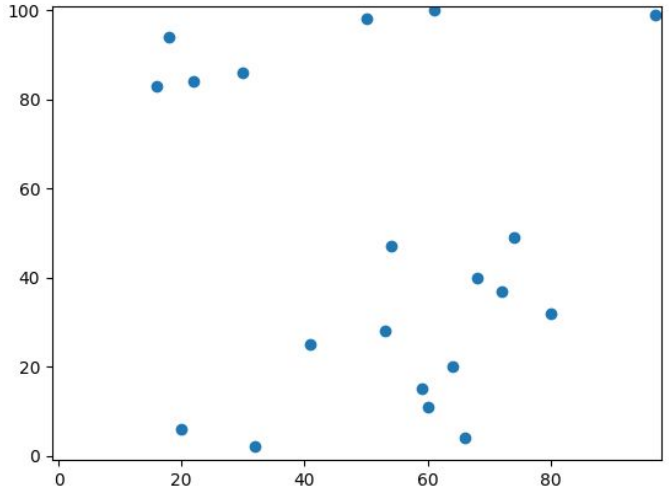
```
python vis.py [input file] --solution [output file]
```

Example:

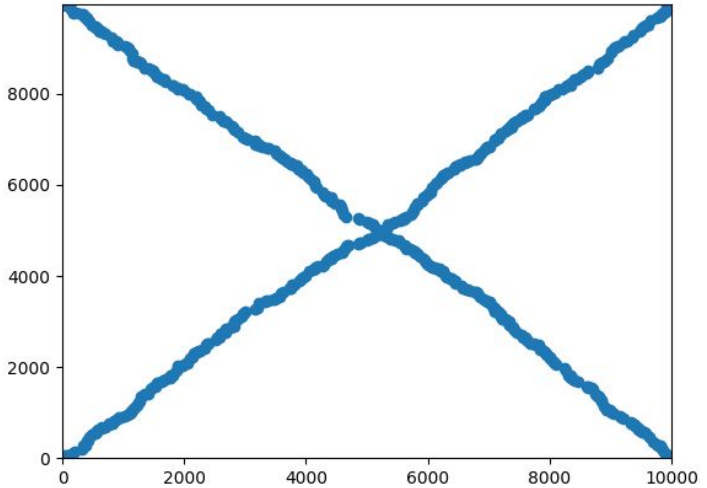
```
python vis.py examples/00.in --solution examples/00.out
```

I1921 - Broken Line

Case 1:

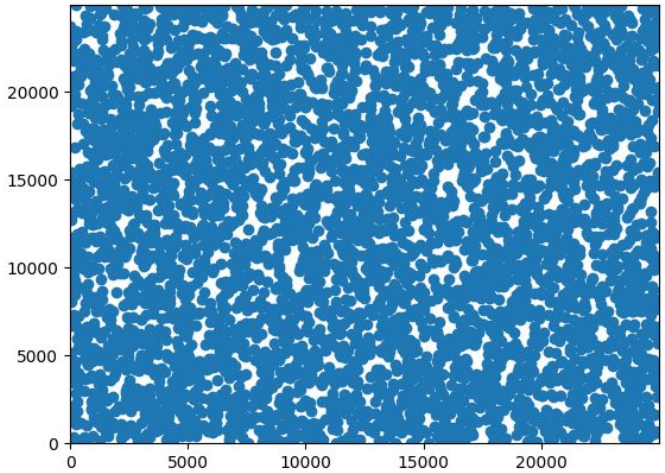


Case 2:

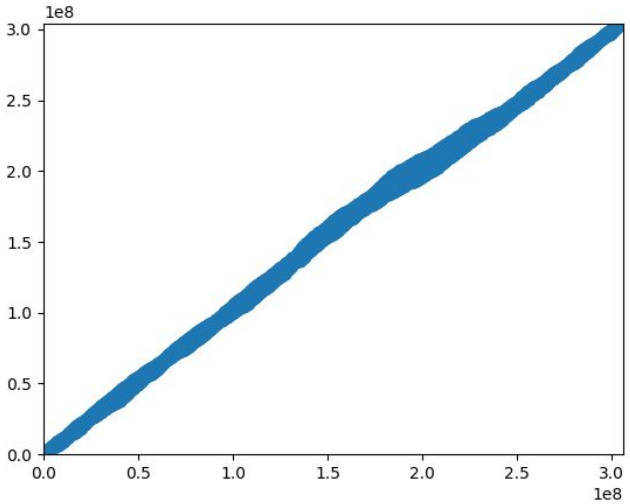


I1921 - Broken Line

Case 3:

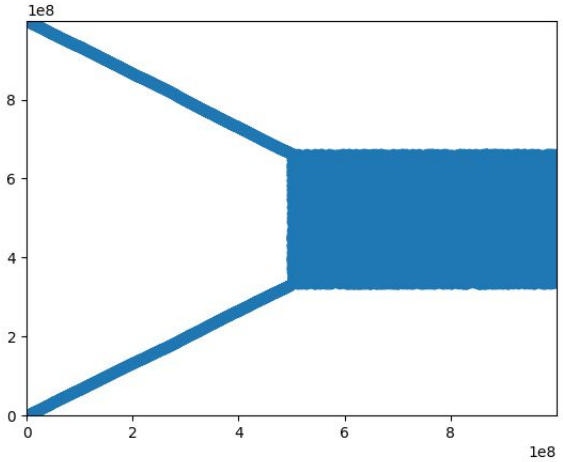


Case 4:

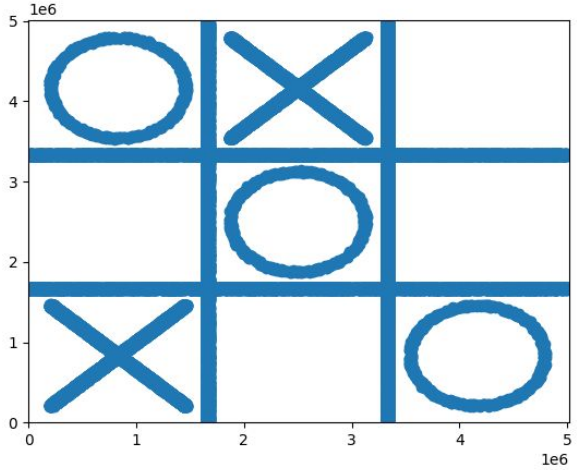


I1921 - Broken Line

Case 5:

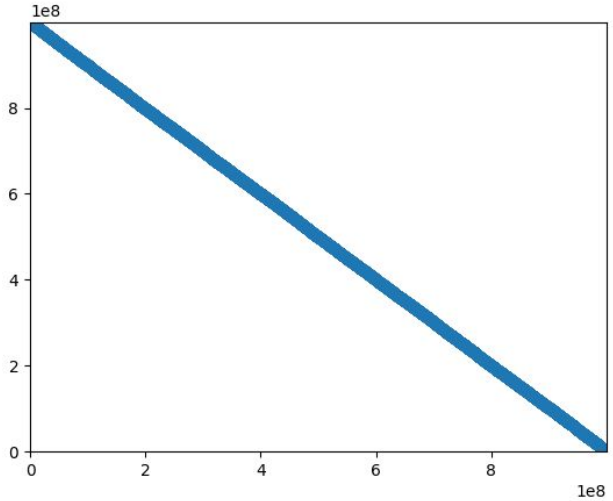


Case 6:

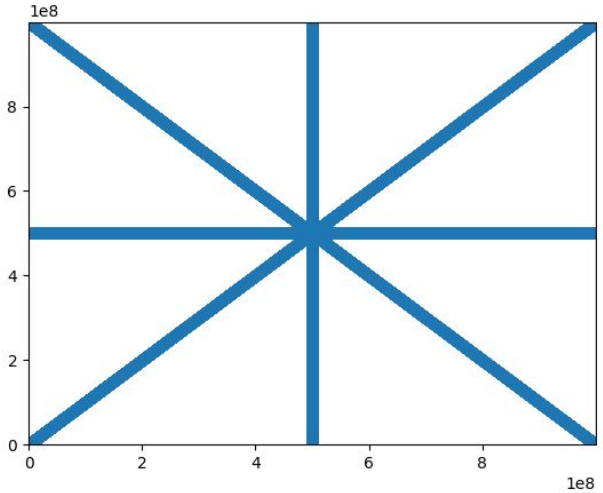


I1921 - Broken Line

Case 7:

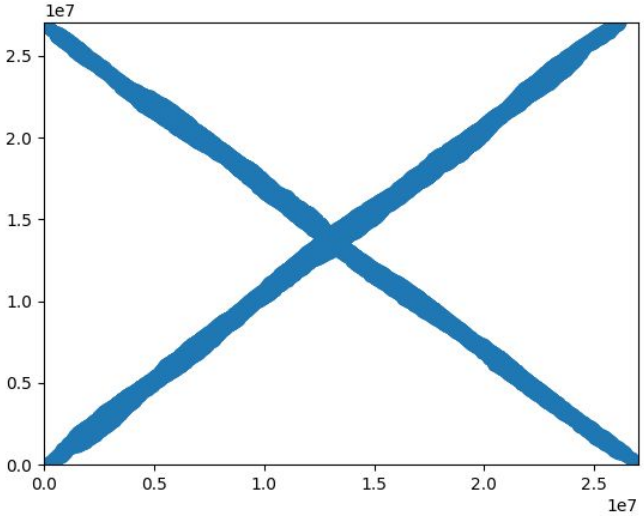


Case 8:

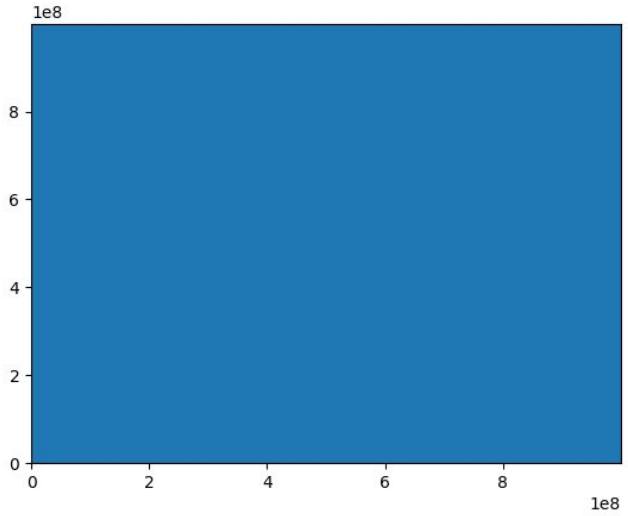


I1921 - Broken Line

Case 9:



Case 10:



I1921 - Broken Line

- **Main Idea:**
  There is a simple solution using **2 segments** for each point.

- If we order the points in a nice-enough way, then some points can be covered using **1 segment** only (extending the previous segment + an extra segment).

I1921 - Broken Line

- **Main Idea:**
  There is a simple solution using **2 segments** for each point.

- If we order the points in a nice-enough way, then some points can be covered using **1 segment** only (extending the previous segment + an extra segment).

- Greedy? Greedy with random? Simulated annealing?

I1921 - Broken Line

Attempt 1 (Greedy):

- $O(n^2)$ greedy: If the next point can be visited by extending the last segment and adding a single extra segment, cover that point.

- If there are no such points, cover the first unvisited point using two segments.

I1921 - Broken Line

Attempt 1 (Greedy):

	1	2	3	4	5	6	7	8	9	10
n	20	600	5000	50000	72018	91891	100000	100000	100000	100000
Full	23	603	5003	50003	72021	91894	100003	100003	100003	100003
Res	26	656	5003	50074	72507	91905	100879	100018	100049	100012
Score	**8.00**	**3.78**	**10.00**	**5.83**	**3.84**	**8.17**	**3.16**	**7.75**	**6.04**	**8.50**

Total: 65.064

I1921 - Broken Line

Attempt 2 (Greedy with Random):

- Use the same greedy solution. However, randomize the points in advance.

I1921 - Broken Line

Attempt 2 (Greedy with Random): *(Each case ~ 1 min)*

	1	2	3	4	5	6	7	8	9	10
n	20	600	5000	50000	72018	91891	100000	100000	100000	100000
Full	23	603	5003	50003	72021	91894	100003	100003	100003	100003
Res	23	644	5003	50055	72489	91899	100866	100019	100047	100011
Score	**10.00**	**4.64**	**10.00**	**5.95**	**3.89**	**9.17**	**3.18**	**7.67**	**6.13**	**8.67**

Total: 69.287

I1921 - Broken Line

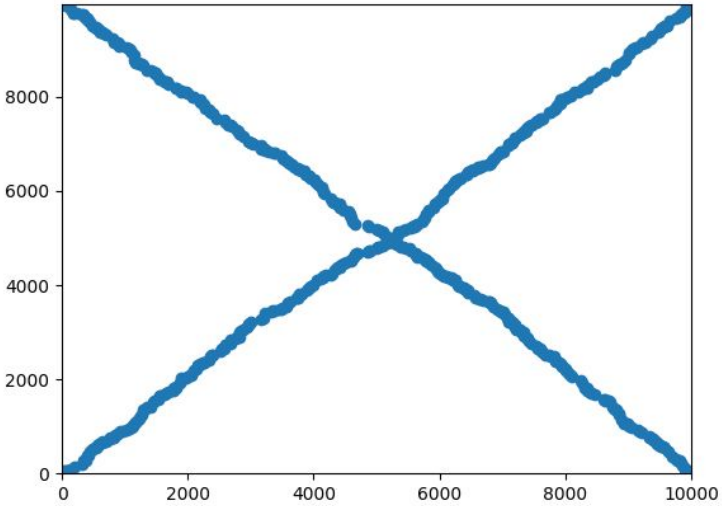
Attempt 2 (Greedy with Random): *(Each case ~ 1 min)*

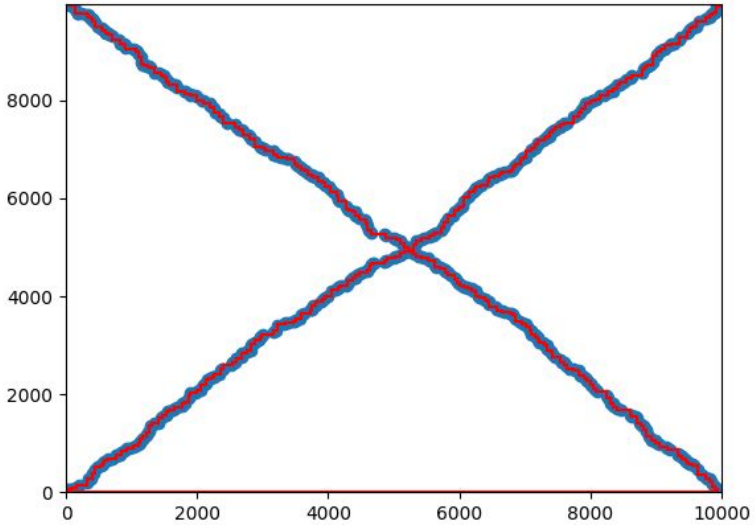
	1	2	3	4	5	6	7	8	9	10
n	20	600	5000	50000	72018	91891	100000	100000	100000	100000
Full	23	603	5003	50003	72021	91894	100003	100003	100003	100003
Res	23	644	5003	50055	72489	91899	100866	100019	100047	100011
Score	**10.00**	**4.64**	**10.00**	**5.95**	**3.89**	**9.17**	**3.18**	**7.67**	**6.13**	**8.67**

Total: 69.287

I1921 - Broken Line

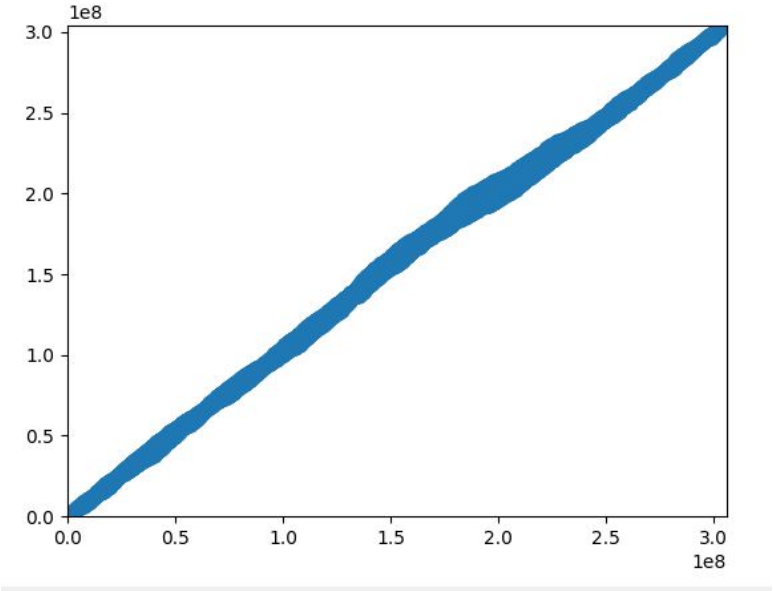
Case 2:

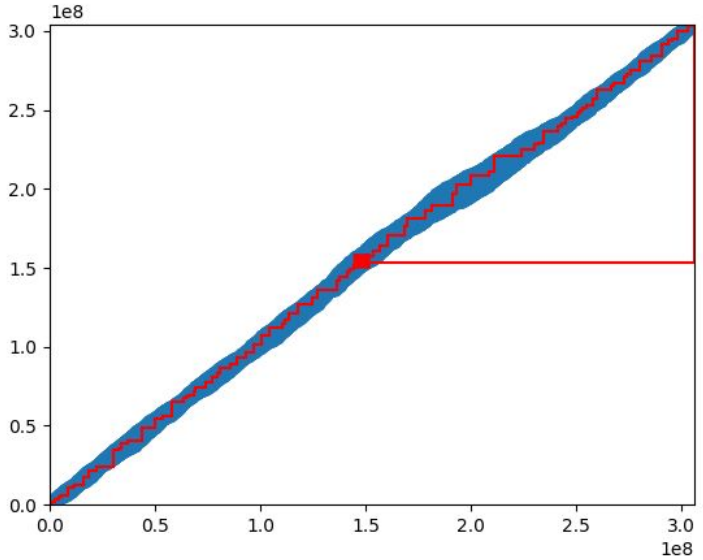




I1921 - Broken Line

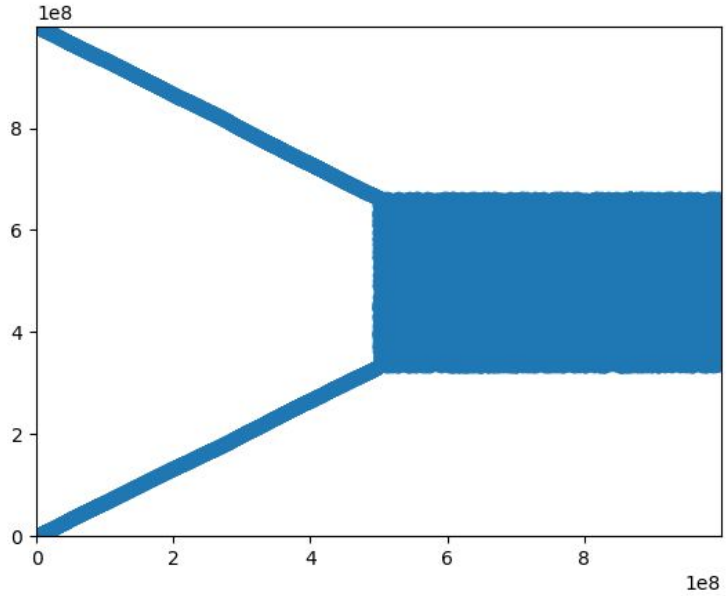
Case 4:

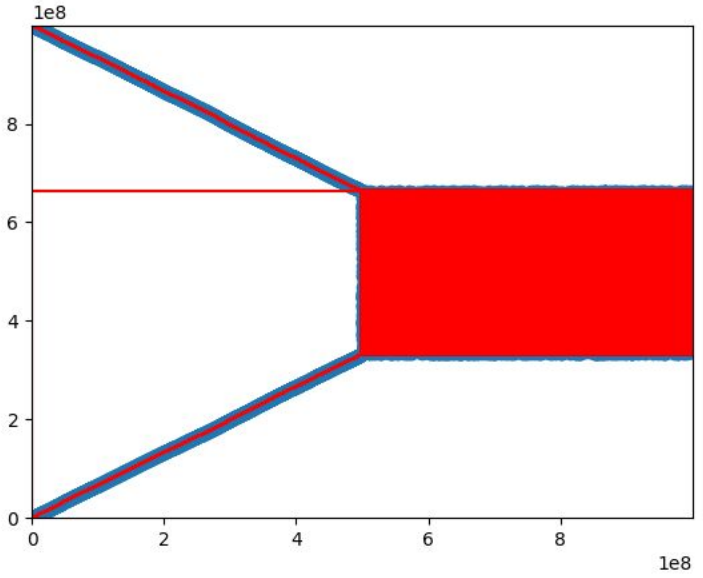




I1921 - Broken Line

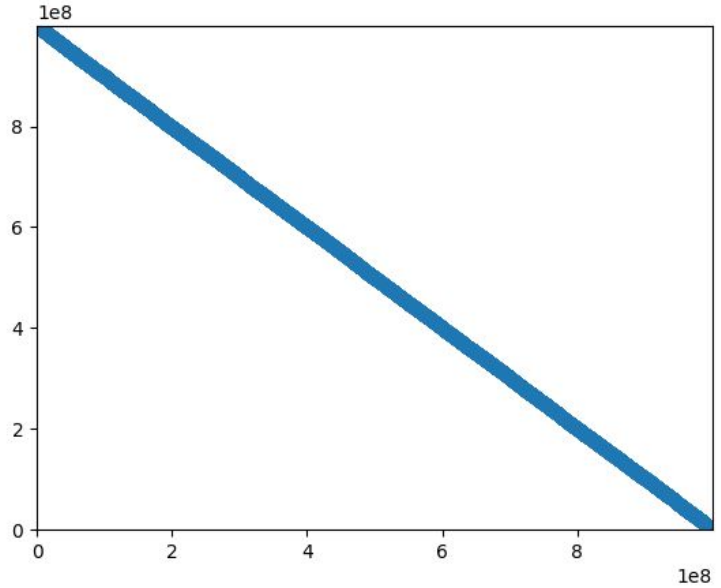
Case 5:

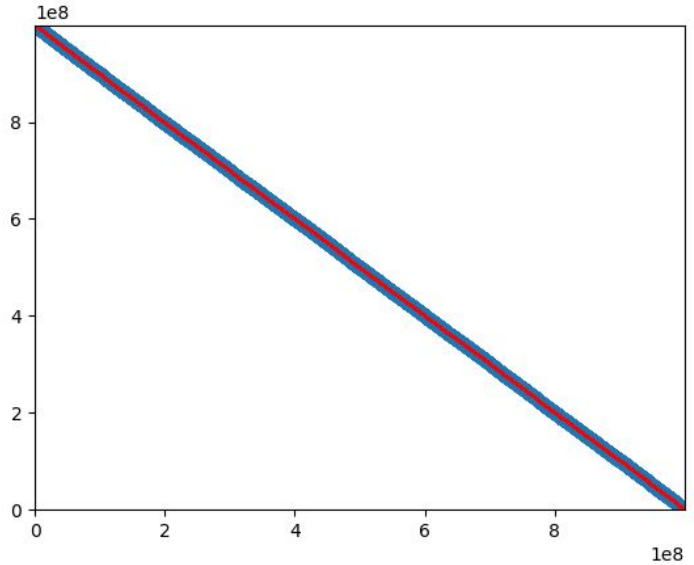




I1921 - Broken Line

Case 7:





I1921 - Broken Line

- Insights?

- Observe that most input files consists of **lines**.
- If x and y coordinates are both increasing, then we can easily cover the points with N segments.
- Run a greedy solution:
  - Sort the points by x-coordinate.
  - Find the longest increasing / decreasing subsequence of y-coordinate.
  - Cover the points by lines.
  - Remove the points covered.

I1921 - Broken Line

Chain Solution:

	1	2	3	4	5	6	7	8	9	10
n	20	600	5000	50000	72018	91891	100000	100000	100000	100000
Full	23	603	5003	50003	72021	91894	100003	100003	100003	100003
Res	23	603	5116	50063	72278	92325	100001	100438	100127	100544
Score	**9.00**	**10.00**	**4.20**	**5.90**	**4.89**	**4.21**	**10.00**	**4.28**	**5.66**	**3.90**

Total: 62.04

I1921 - Broken Line

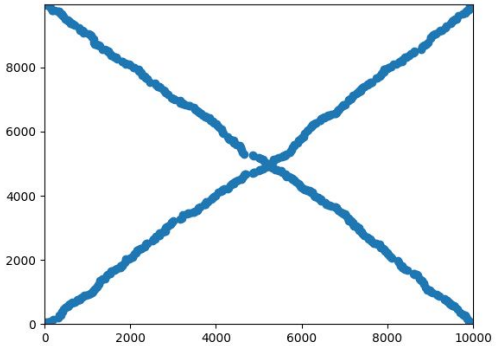
Cumulative:

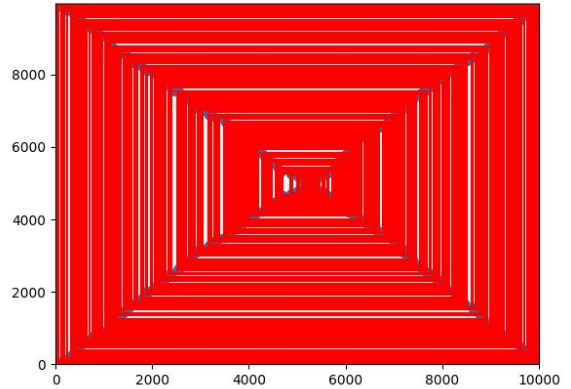
	1	2	3	4	5	6	7	8	9	10
n	20	600	5000	50000	72018	91891	100000	100000	100000	100000
Full	23	603	5003	50003	72021	91894	100003	100003	100003	100003
Res	23	603	5003	50055	72278	91899	100001	100018	100047	100011
Score	**10.00**	**10.00**	**10.00**	**5.95**	**4.89**	**9.17**	**10.00**	**7.75**	**6.13**	**8.67**

Total: 82.56

I1921 - Broken Line

- Alternatively, for test cases with diverse points, we can choose the top-left, top-right, bottom-left and bottom-right corner, and cover them using 4 segments.
  → Do it **recursively**





I1921 - Broken Line

Spiral Solution:

	1	2	3	4	5	6	7	8	9	10
n	20	600	5000	50000	72018	91891	100000	100000	100000	100000
Full	23	603	5003	50003	72021	91894	100003	100003	100003	100003
Res	23	865	5012	50194	96024	91931	199999	109168	100112	100005
Score	**10.00**	**2.27**	**8.00**	**5.06**	**2.35**	**6.43**	**1.00**	**2.83**	**5.72**	**9.67**

Total: 53.34

I1921 - Broken Line

Adapted From IOI 2019 Solutions

Full Solution:

- Full solution involves combining the **chain** and **spiral** solution:
  - Let us reconsider the spiral. We waste additional segments only if the bounding box contains less than 4 points (for example there's a point that is both the left-most and the top-most one).
  - We can remove these points and put them in one of two chains (one going from the top-left corner to the bottom-right corner) and one going from the top-right corner to the bottom-left one).

I1921 - Broken Line

Adapted From IOI 2019 Solutions

Full Solution:

- The algorithm is as follows:
  - if a bounding box has 4 points in it, add them to the **spiral**, and remove these points,
  - otherwise, find a point that lies on two sides of the bounding box, add it to the proper **chain** and remove it.
- Then we have three objects (spiral and two chains) and we can use up to 2 segments to connect them to themselves and the origin. This solution gets about 95 points. (N + 6 segments)

I1921 - Broken Line

Adapted From IOI 2019 Solutions

Full Solution:

- N + 3 solution
- To come up with an even better solution, we need to add some small tricks, including considering all possible orders in which we can connect these objects and directions in which we can traverse them. These optimizations lead a solution with N + 3 segments, which gets 100 points.

Common Techniques

- Manually solve by hand (with the aid of draft paper or drawings / visualizer)
- Searching / exhaustion algorithm (to find any solution satisfying the constraints)
  - Usually scores 1 point per case
- Special algorithms designed for cases with special patterns
  - Might score 10 points in some cases
- Heuristic Searching (Hill-climbing / Simulated Annealing)

Hill Climbing

- Let's denote each answer as a **State.**
- A **neighbour state**, is some state that you can get to by applying some small transformation (e.g. swap two elements in a sequence), the answer should not be changed too much.
- Start with a random state
  - Each time, enumerate a number of neighbouring states.
  - Calculate the scores of the neighbouring states and find the best one.
  - If that neighbouring state is better than the current state, accept the neighbouring state.
  - Otherwise, the algorithm ends. You may start all over with another random state.

Hill Climbing

Pseudocode

```
for i = 1..rounds:
    state = random state (or state from other solutions)
    while true:
        for j = 1..neighbour_no:
            new_state = state.random_transition()
            if new_state.score() > best_score:
                best_score = new_state.score()
                best_state = new_state
        if best_score > state.score() then
            state = best_state
        else: break
```

Hill Climbing

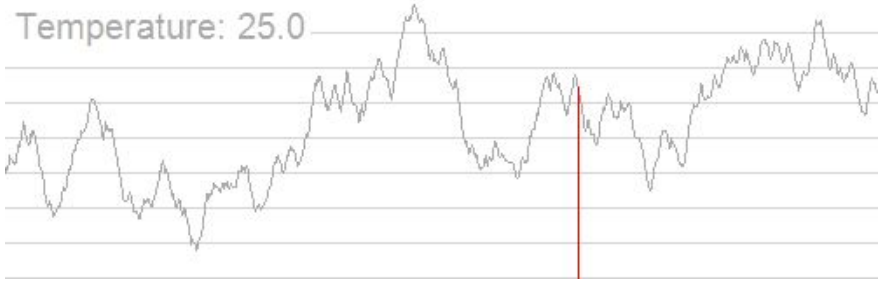
Drawback:

- It often gets stuck in a local minimum, instead of reaching a global minimum.
- Also, it could not handle plateau (nearby state return indistinguishable score), hill climbing do not know which direction to go

- It depends largely on the initial state chosen, so mostly rely on luck

Simulated Annealing

- Simulated Annealing helps fix the issue by allowing transition to a worse neighbour, sometimes, in order to reach the global maximum.
- It depends on,
  - **"Acceptance Probability Function":** a function that determines whether a step should be taken or not
  - **Temperature value**: a value that determines how big of a "bad" step to take
- Progressing through the iteration, the temperature gets lower and lower



(from wikipedia)

Simulated Annealing

Pseudocode

```
state = random state or empty state
for i = 1..rounds:
    new_state = state.random_transition()
    if new_state.score() > state.score():
        state = new_state
    else:
        temperature = f(i / rounds) # percentage done
        probability = e^((new_state.score() - state.score()) / temperature)
        if rand(0, 1) < probability:
            state = new_state
    if state.score() > best_state.score():
        best_state = state
```

Assumption, we want to maximize score and score ranges from **0** and **1**
Example function:
temperature = 0.2 x 0.999^i

Simulated Annealing

Pseudocode

```
state = random state or empty state
for i = 1..rounds:
    new_state = state.random_transition()
    if new_state.score() > state.score():
        state = new_state
    else:
        temperature = f(i / rounds) # percentage done
        probability = e^((new_state.score() - state.score()) / temperature)
        if rand(0, 1) < probability:
            state = new_state
    if state.score() > best_state.score():
        best_state = state
```

If temperature == 0.1,

Old score = 0.5, New score = 0.45
Difference = -0.05
Probability = e^-0.5 = **60.6%**

Old score = 0.5, New score = 0.35
Difference = -0.15
Probability = e^-1.5 = **22.3%**

Different functions in Simulated Annealing

- Reference: [Tutorial] Simulated Annealing in Competitive Programming
- **Neighbour Function:**
  - Apply a small transition (it is ok even to transite to invalid state, just make it have -inf score)
  - The **diameter** of the search space graph better be **small**: it should be able to transit from one state to another in small amount of moves.

Different functions in Simulated Annealing

- **Temperature Function**
  - **At 0**: only move to better states, **at inf**: will move to any state
  - You would like it to go from high temperature to low temperature:
    - At first, more states should be explore to make you have better chance at reaching the global maximum
    - At the end, you want to take better state to hopefully attain the global maximum
  - 3 common reduction rules:
    - **t = t * a**
    - **t = t - a**
    - **t = t / (1 + at)**

The most commonly used is the first one, set $\mathbf{T_0}$ as some high value (like $\mathbf{10^5}$), and set a as **0.999** (close to **1**)

Different functions in Simulated Annealing

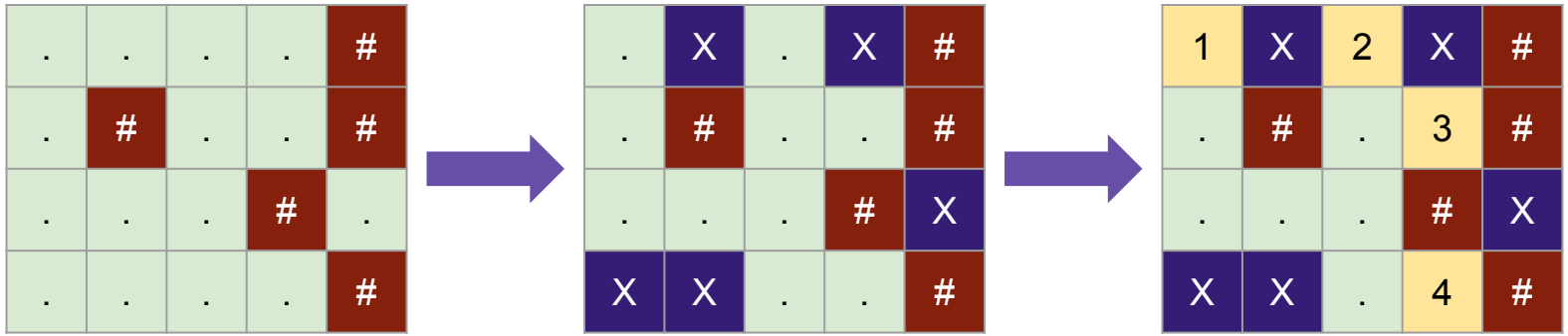
- **Acceptance Probability Function**
  - Determine the probability of going to a new state, given
    - Score of old state
    - Score of new state
    - Temperature
  - Most commonly use function is the following:

```
if (new_score > old_score): return 1.0
else return exp((new_score - old_score) / temperature)
```

I1711 Nowruz

Problem:

- Given an **n** × **m** grid with some obstacle cells
- Build a maze that has as many「堀頭路」(dead end) as possible
  - 「堀頭路」(dead end): cell that has exactly one free neighbour



- An infamous task since the score distributions have low correlations with students' performance. (and many more reasons...)

I1711 Nowruz

- [Codeforces Blog] Personal view to output-only problem in IOI
- Some criticisms of the problem includes:
  1. There are **no ways to logically reason** about if a solution is good or not without trial-and-error and submitting → not testing a contestant skill but testing if they are lucky enough with their solution
  2. An important condition about **input data**, which was **not mentioned in the problem description**: the input file contains mostly empty space, with the empty space being in one connected component

I1711 Nowruz

- While how valid these criticisms are up for discussion, at least they give these takeaways:

- Open the Input file! They are given to you, so you should read it.
  - Watch for patterns!
- Try to come up with as many ideas as possible, don't hesitate to use trial-and-error or heuristics.
  - some ideas may work better than you think
  - Could be weak inputs, weak scoring function or inherent limitation of the problem
  - Assess which idea is the most cost-effective (cost = coding time)

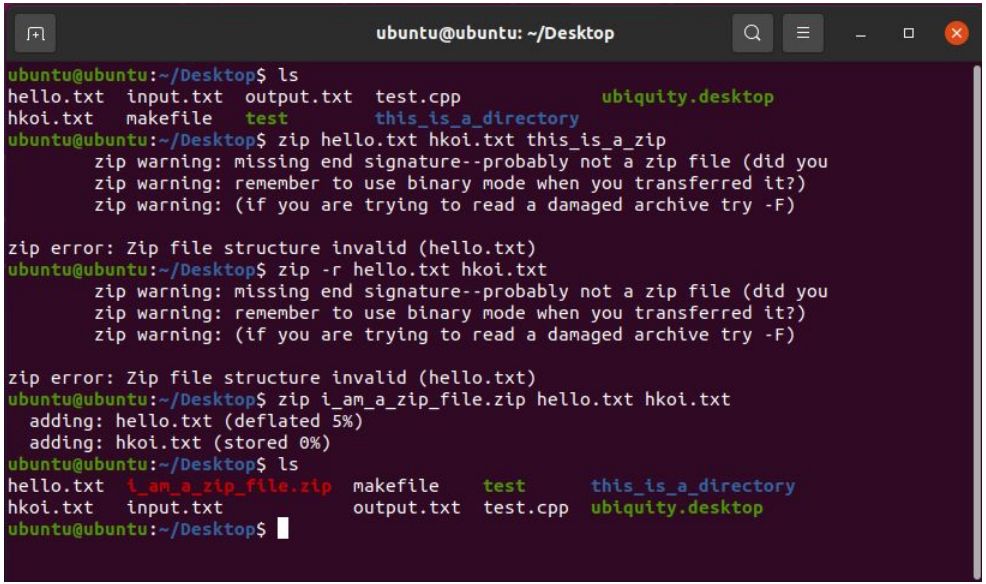
Sidenote: Archive utility - `zip` / `unzip`

Adapted from Introduction to Linux

- In Linux, we can use `zip` to compress multiple files / directories to a single archive, or `unzip` to expand the archive.
- Usage:
  - `zip [output_zip_filename] [file1] [file2] [file3]…`
  - `zip -r [output_zip_filename] [directory1] [file1] [file2]…`
    - When zipping a directory, the `-r` argument must be used
  - `unzip a.zip`
    - This unzip the contents of an archive file

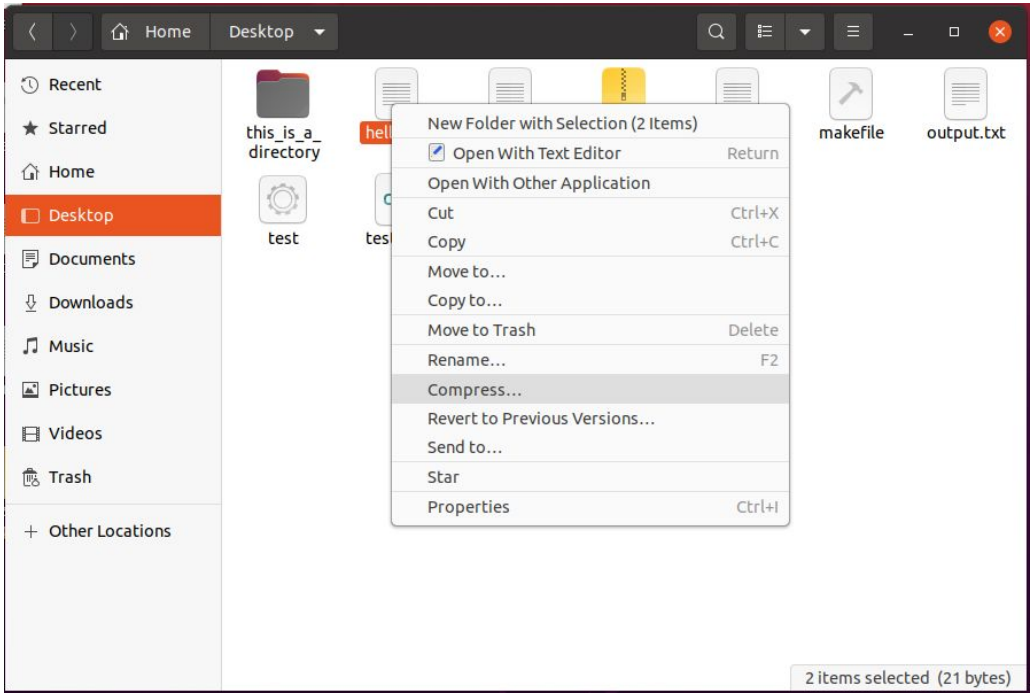
Sidenote: Archive utility - `zip` / `unzip`

Adapted from Introduction to Linux



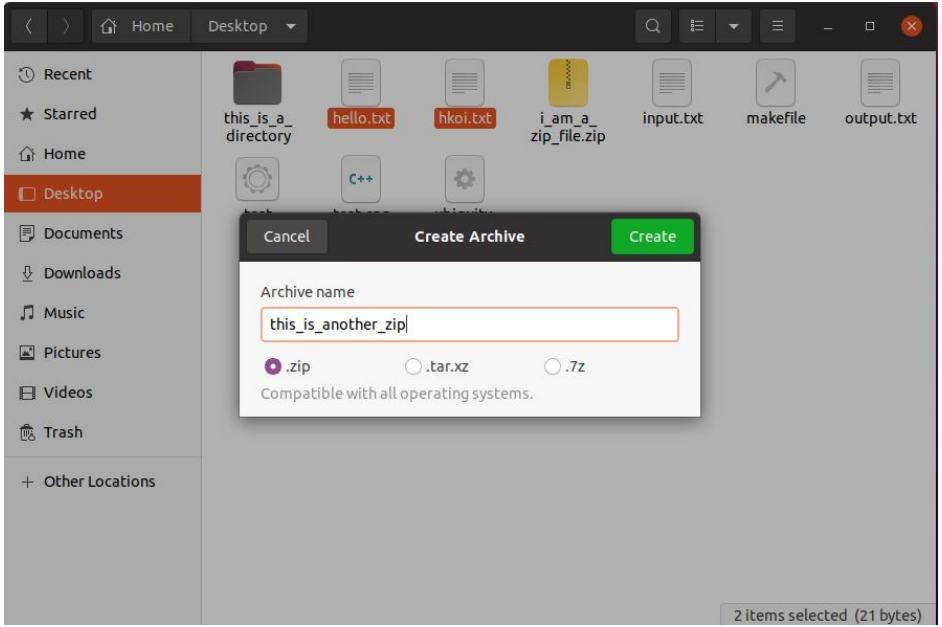
Sidenote: Using GUI to compress

Adapted from Introduction to Linux



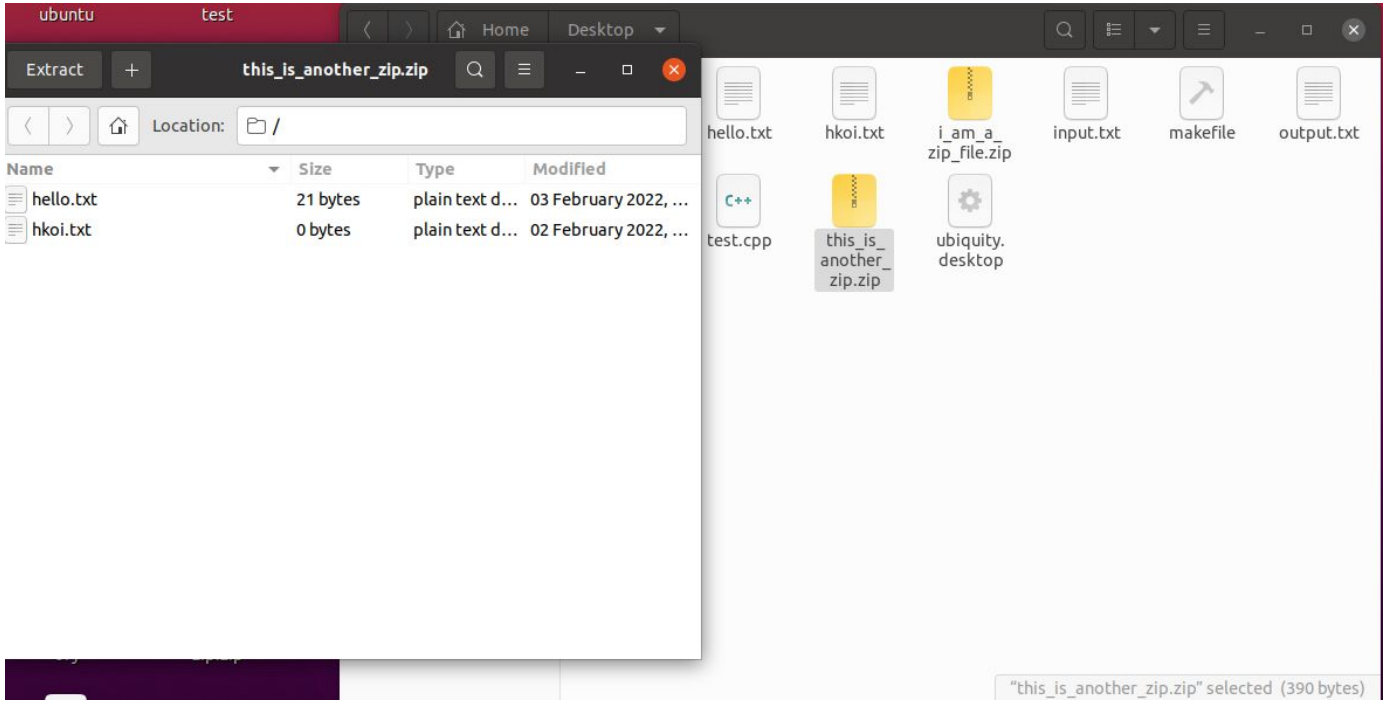
Sidenote: Using GUI to compress

Adapted from Introduction to Linux



Sidenote: Using GUI to compress

Adapted from Introduction to Linux



Practice Tasks

- [IOI 2010] Maze
- [IOI 2012] Pebbling Odometer
- [IOI 2017] Nowruz
- [IOI 2019] Packing*
- [IOI 2019] Line ★
- [IOI 2025] Duplicated Binary Strings*#
- [IOI 2025] Triple Peaks#
- [IOI 2011] Parrots ★
- [IOI 2019] Transfer*
- [IOI 2020] Squares*
- [IOI 2020] Stations
- [IOI 2022] Magic Cards*
- [IOI 2024] Treasure*
- [IOI 2024] Message ★
- [IOI 2025] Magic Trick*
- [IOI 2025] Migrations

*appeared in practice section

hybrid of normal subtasks + oo subtask

- **Output-only**
- **Communication**

Practice Tasks

- [TFT 2017] Constellation
- [TFT 2018] Exam Anti-Cheat
- [TFT 2020] Metros of Runeterra
- [TFT 2022] Paint the Ceiling
- [TFT 2014] Lost Sequence
- [TFT 2024] Tree Speculation

- **Output-only**
- **Communication**

Practice Tasks

- M1944 Prime Scrabble
- M2244 Dynamic Training
- M2334 A Way Out ★
- M2454 The Great Awakening
- M2532 Coprime Cycle
- M1743 Tree Recovery II
- M2332 Collaborative Sudoku ★
- M2352 Prisoners' Gamble ★
- M2433 Mind Reading

- **Output-only**
- **Communication**

Conclusion

- Just like constructive task, non-batch task is another type of problems
  - NOT LIMITED by any algorithms, topics
  - therefore, no standard rules to deal with them
  - again, “practice makes perfect”
  - as long as you solve / take a look at more non-batch tasks,
  - more techniques / experiences you can accumulate
- From the history of Team Formation Test,
  - non-batch tasks often appear :)
  - good luck :)

Sidenote: Additional preparation for TFT

Try to be familiar with Linux command line interface

- TFT / IOI / NOI usually provide Linux machines
- might not have VSCode / Dev C++ / other IDEs

Things to learn

- How to compile and run a program
- How to redirect the stdin/stdout/stderr streams

If you're a Windows user and want to try Linux CLI for competitive programming, consider using Windows Subsystem for Linux

Minicomp

- Try to work on some special tasks!
  - Teams of 2, work collaboratively!
  - Duration: Around 1 hour (depends on lesson time)

- Coins (50%)
  - Communication Task
- Packing (50%)
  - Output-only Task

- Team with highest score will get prizes

Running C++ locally in CityU computer

- CodeBlocks
- Search CodeBlocks in Google > Download Binary releases
- Codeblocks-25.03mingw-setup.exe > Sourceforge.net > Download Installer
- Run Installer

- DevC++
- Desktop > Work Desk > Dev-C++ 5.11
- Run Installer

Minicomp Solution Session

- Coins
  - Source: IOI 2017 Practice T3
  - Communication Task
- Packing
  - Source: IOI 2019 Practice T3
  - Output-Only Task
- We will go over some interesting insights gain from the question, not aiming to explain the whole solutions in details.

M26B1
Coins

The Problem

- Given a 8x8 grid (numbered 0-63) of 0 and 1.
- Alice is given the grid and a specific cell C. She can flip at least 1 cell and at most K cells on the grid (i.e. 0->1 or 1->0).
- Bob is given the grid after flipping. He need to deduce the cell number C.

- Subtasks constraints on value of C and bound of K
- Full solution: 0 <= C < 64, K = 1

Solution: K = 1, C < 2

- Easiest way is to just make bit 0 decide whether C is 0 or 1.
- Alice look at bit 0. If bit 0 is correct already, we flip bit 1. Else we flip bit 0.

- Bob can just look at bit 0 to decode.

Solution: K = 64

- Alice can just flip everything to make cell C is different than others.
- e.g. cell C is 1 and everything is 0

- Bob can identify cell C by finding the coin 1.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Solution: K = 8

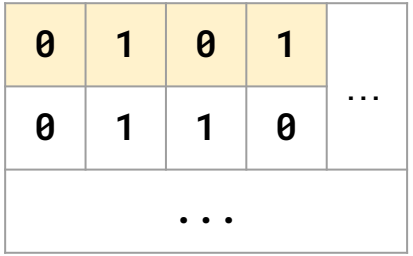
- Alice can represent the position of C with binary encoding by fixing the first 6 bit (0-5)

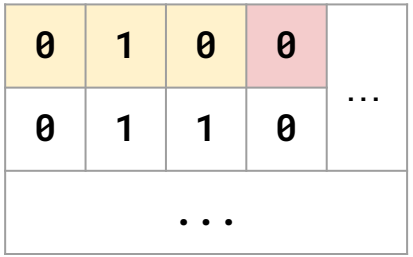
- Bob can look only at first 6 bit to decode the position.
- e.g. 011101 -> cell 29

0	1	1	1	0	1	0	0
0	1	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	1
0	0	1	0	1	1	1	0

Solution: K = 1, C < 3

- We need to send 2 bits of information (C = 0, 1, 2) with 1 flip. Let's use existing information on the board.
- A property of XOR is useful
  - Let's say we have A and B < 2^k
  - We can always find a C < 2^k and A xor C = B
- We can represent C by xor sum of index of first 4 bit
  - e.g. the current first 4 bits is 0101
  - 1 xor 3 = 2
  - If C is 1, we know that 2 xor 1 = 3
  - Alice can flip bit 3 to make to xor sum correctly represent 1.
  - Bob see first 4 bits is 0100 and know the answer is 1.





Solution: K = 1

- We can see that the previous subtask solution actually can expand to C < 64.
  - Using 4 bits of the board can communicate 2 bits of information through 1 flip
  - Using 64 bits of the bound can communication 8 bits of information through 1 flip
- i.e. represent C by calculating this on the board:

```
for i in 0..63: if (board[i] == 1) C ^= i
```

- Alice can always flip a single cell in the board to make C correct.

M26B2
Packing

The Problem

- You are given N number. Each number is a positive integer between 1 to 10.
- You need to pack the numbers into multiple groups. Sum of number in each group <= 20. Minimize number of groups. You are given the target number of groups K.
- R = K / L, where L is the number of group your solution attains.
    - R >= 1 (i.e. L <= K), you will get 10 marks
    - 0.75 < R < 1, score = 1 + 9^(4(R - 0.75))
    - R <= 0.75, score = 8R / 3
- Bin Packing Problem: NP-hard
    - The number is quite small though (number 1-10, capacity = 20)

Always look at the Data

Filename	n	k	TotalVol	MinBox	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
packing1.in	20823	7138	142743	7138	520	874	1272	1598	1895	2263	2563	2895	3317	3626
packing2.in	32020	10990	219786	10990	776	1321	1893	2451	2937	3542	3991	4514	5051	5544
packing3.in	21953	7555	151089	7555	498	923	1277	1689	1999	2342	2801	3117	3502	3805
packing4.in	93894	28034	540528	27027	2702	2666	642	42	41974	2694	42413	47	41	673
packing5.in	46084	15810	316195	15810	1169	1967	2679	3471	4219	4968	5778	6554	7400	7879
packing6.in	44352	15295	305697	15285	1124	1816	2557	3319	3960	4822	5584	6185	7156	7829
packing7.in	97867	29242	563775	28189	2723	2721	700	46	43923	2722	44274	39	41	678
packing8.in	81029	24165	466101	23306	2333	2220	562	35	36609	2298	36321	28	32	591
packing9.in	96207	34550	685968	34299	2313	9454	2416	534	4787	18806	1190	18936	18936	18835
packing10.in	79528	23759	458067	22904	2180	2276	617	36	35652	2235	35853	27	37	615

- Case 1, 2, 3, 5: Must be exact fit -> Sum / 20 = k
  - Naive greedy may not do so well

Always look at the Data

Filename	n	k	TotalVol	MinBox	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
packing1.in	20823	7138	142743	7138	520	874	1272	1598	1895	2263	2563	2895	3317	3626
packing2.in	32020	10990	219786	10990	776	1321	1893	2451	2937	3542	3991	4514	5051	5544
packing3.in	21953	7555	151089	7555	498	923	1277	1689	1999	2342	2801	3117	3502	3805
packing4.in	93894	28034	540528	27027	2702	2666	642	42	41974	2694	42413	47	41	673
packing5.in	46084	15810	316195	15810	1169	1967	2679	3471	4219	4968	5778	6554	7400	7879
packing6.in	44352	15295	305697	15285	1124	1816	2557	3319	3960	4822	5584	6185	7156	7829
packing7.in	97867	29242	563775	28189	2723	2721	700	46	43923	2722	44274	39	41	678
packing8.in	81029	24165	466101	23306	2333	2220	562	35	36609	2298	36321	28	32	591
packing9.in	96207	34550	685968	34299	2313	9454	2416	534	4787	18806	1190	18936	18936	18835
packing10.in	79528	23759	458067	22904	2180	2276	617	36	35652	2235	35853	27	37	615

- Case 4, 7, 8, 10: A lot of 5 and 7
  - Special handle? 5 + 7 + 7 = 19? 5 + 5 + 5 + 5 = 20?
  - Greedy may already works quite well with the gap between k and MinBox.

Always look at the Data

Filename	n	k	TotalVol	MinBox	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
packing1.in	20823	7138	142743	7138	520	874	1272	1598	1895	2263	2563	2895	3317	3626
packing2.in	32020	10990	219786	10990	776	1321	1893	2451	2937	3542	3991	4514	5051	5544
packing3.in	21953	7555	151089	7555	498	923	1277	1689	1999	2342	2801	3117	3502	3805
packing4.in	93894	28034	540528	27027	2702	2666	642	42	41974	2694	42413	47	41	673
packing5.in	46084	15810	316195	15810	1169	1967	2679	3471	4219	4968	5778	6554	7400	7879
packing6.in	44352	15295	305697	15285	1124	1816	2557	3319	3960	4822	5584	6185	7156	7829
packing7.in	97867	29242	563775	28189	2723	2721	700	46	43923	2722	44274	39	41	678
packing8.in	81029	24165	466101	23306	2333	2220	562	35	36609	2298	36321	28	32	591
packing9.in	96207	34550	685968	34299	2313	9454	2416	534	4787	18806	1190	18936	18936	18835
packing10.in	79528	23759	458067	22904	2180	2276	617	36	35652	2235	35853	27	37	615

- Case 5, 6, 9
  - A lot of large number. Not enough fillers (1, 2). May be quite hard.

Greedy

- A naive greedy algorithm is First Fit Decreasing.
- Sort all numbers from large to small.
- Find the **first** bin that the number can still fit in. Put the number in it.
- If no bin can fit the number, open a new bin.
- Turns out this already give us 91 points, and align with our analysis.

Case	k	Boxes l	Score
1	7138	7244	8.914
2	10990	11154	8.909
3	7555	7665	8.934
4	28034	27817	10.000
5	15810	16039	8.939
6	15295	15515	8.945
7	29242	29024	10.000
8	24165	23982	10.000
9	34550	36216	7.007
10	23759	23579	10.000
Total Score			91.647

Greedy

- Sort all numbers from large to small.
- We can give some noise to this to see if it can be better.
  - Randomly swap some number in close proximity
  - Do this 100 times and save the best result
- Sadly, it only helps a little bit.
- We need something that is more disruptive.

Case	k	Boxes l	Score
1	7138	7242	8.933
2	10990	11152	8.921
3	7555	7663	8.951
4	28034	27817	10.000
5	15810	16036	8.951
6	15295	15513	8.954
7	29242	29024	10.000
8	24165	23982	10.000
9	34550	36215	7.008
10	23759	23579	10.000
Total Score			91.720

Some Thinking

- A problem is that the greedy solution, although give us a ok baseline, it can create a lot bad bucket e.g. size 18 19 but we need to get exact fit for some cases.
- We want to find a way to release these space for better use.

Local Search

- An idea to get a better solution is to:
  1. Choose some of the bucket with the most remaining space + some random full bucket.
  2. Try to repack these numbers to see if we can get better solution.
  Hill climbing (10 iteration)
- 91.7 -> 92.96! Very promising!

Case	k	Boxes l	Score
1	7138	7217	9.174
2	10990	11116	9.147
3	7555	7640	9.162
4	28034	27817	10.000
5	15810	15982	9.188
6	15295	15462	9.185
7	29242	29024	10.000
8	24165	23982	10.000
9	34550	36141	7.112
10	23759	23579	10.000
Total Score			92.968

Local Search

- Just do more iterations!
- Iteration 10 -> 200
- Score 92.96 -> 97.79

Case	k	Boxes l	Score
1	7138	7143	9.945
2	10990	10999	9.936
3	7555	7560	9.948
4	28034	27817	10.000
5	15810	15825	9.925
6	15295	15299	9.979
7	29242	29023	10.000
8	24165	23981	10.000
9	34550	35529	8.064
10	23759	23579	10.000
Total Score			97.797

Local Search

- Just do more iterations!
- Iteration 200 -> 400
- Score 97.79 -> 97.92

- Most cases are pretty saturate already, only case 9 is giving us significantly more marks.

Case	k	Boxes l	Score
1	7138	7142	9.956
2	10990	10997	9.950
3	7555	7560	9.948
4	28034	27817	10.000
5	15810	15824	9.930
6	15295	15296	9.995
7	29242	29022	10.000
8	24165	23982	10.000
9	34550	35479	8.150
10	23759	23579	10.000
Total Score			97.928

Further Ideas

- Tune the number of number of full bucket destroyed in one search
- Try some more random ideas
- Try to tune the hill climbing scoring function
  - Now we only accept if the bucket number decreases
  - What about if the bucket is packed a bit tighter?
    - e.g. more bucket that is full?
    - This may be a better state and we should hill climb on these as well.

Takeaways

- Don’t be afraid to attempt output-only tasks
  - Sometimes OO can be very easy to get marks from

- If you are stuck, throw a bunch of ideas at it and see which gets the best score
  - This might reveal some hidden insight of the problem