HW4

Rules & Instruction

- Compiler of programming problem.
  - C : `gcc -DONLINE_JUDGE -O2 -w -fmax-errors=3 -std=c11 main.c -lm -o main`
  - C++: `g++ -DONLINE_JUDGE -O2 -w -fmax-errors=3 -std=c++17 main.cpp -lm -o main`
  - Execution: `./main`
- Can I use theorems that haven't been mentioned in class?
  - Programming: No restriction. (無限制)
  - Handwritten: Please include its proof. (請寫上證明。)
- Can I refer to resources from the Internet or other sources that are not from textbooks or lecture slides?
  - Yes, but you must specify the references (the Internet URL you consulted with or the name and the page number of books). (可以，但必須附上參考來源，如網址或書名和頁數)
    - Handwritten: specify the references next to your solution. (手寫題請將參考附註於你的答案旁，你可以使用短網址工具)
    - Programming: specify the references at the top of your code in comments. (程式題請將參考以註解的方式附註於程式碼的最上方)
  - Although you can use external resources, it doesn't mean you can just paste the resources as your solution. **You have to answer by yourself**, and remember to specify the references; otherwise we'll view it as cheating. (雖然參考外部資源是允許的，但請用自己的話去作答，並切記要註記參考來源，否則會以作弊/抄襲處理)
- Rules of cheating.
  - Programming: We use tools to monitor code similarity. (我們用機器抓抄襲)
  - Handwritten: Though we forbid plagiarism, we still encourage you to discuss with each other. Remember that after discussion you must answer problems **in your own words**. (禁止抄襲，鼓勵大家有問題可以互相討論，**但請用自己的話去作答**。)
  - **Copycats will be scored 0.** You also need to have a cup of coffee with Professor Yeh. (**抄襲者與被抄襲者當次作業零分**，另外會跟教授喝杯咖啡☕。)
    - People who committed twice or more will be punished. (情節嚴重者，如累犯...，以校規處置)
- Rules of delay.
  NO LATE SUBMISSION IS ALLOWED.
  That is, zero toleration of late submission.

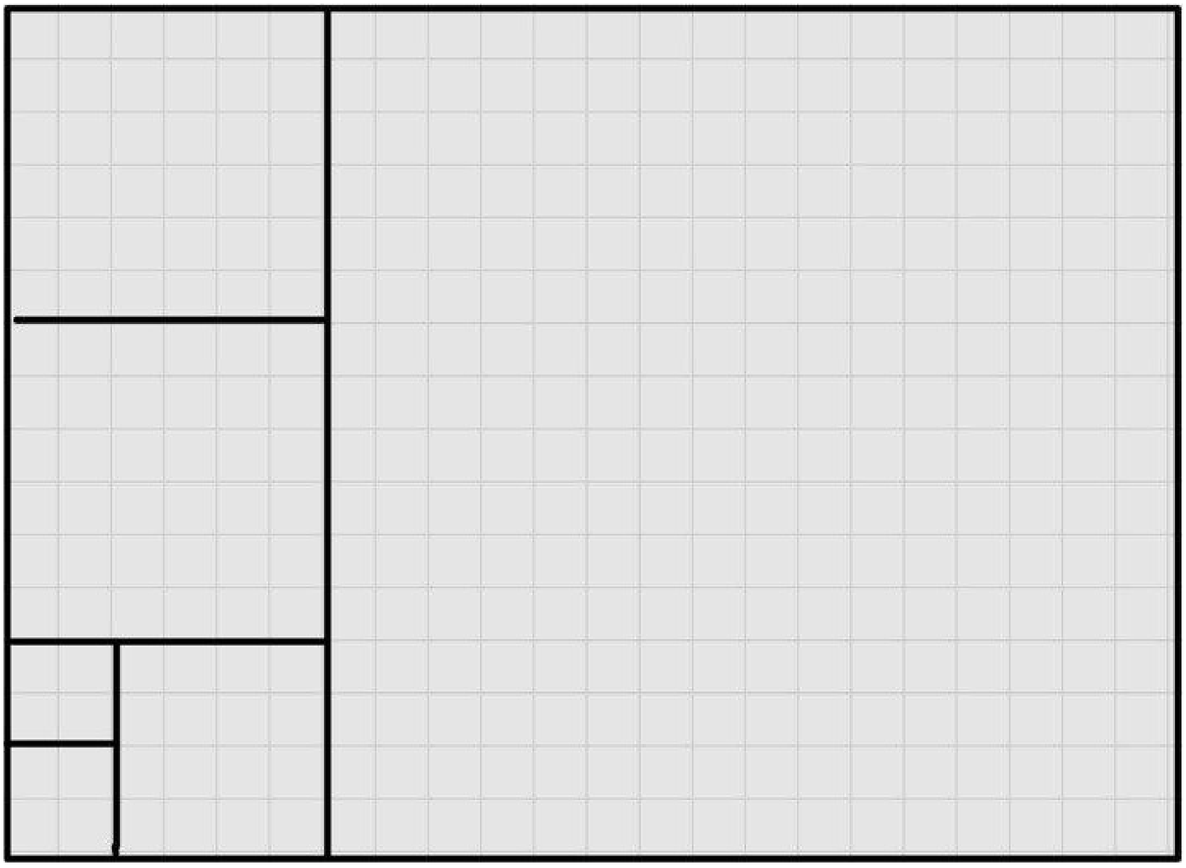
Non-Programming

1. Square

There is a rectangle with width W, and height H, now we wonder how to split the rectangle into the least amounts of squares. The way we split is that if the size of the rectangle is $i \times j$ assuming W.L.O.G $i < j$, we will split the rectangle into two parts, one will be size $i \times i$, the other will be size $(j - i) \times i$. After that, we will apply the way again until the initial rectangle is split into squares. (See examples following)

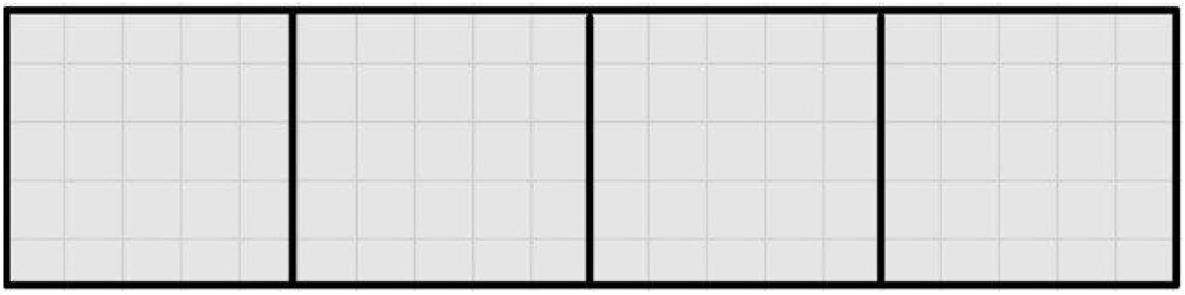
Example 1

> Width: 22
> Height: 16
> Used Squares: 6



Example 2

> Width: 20
> Hight: 5
> Used Squares: 4



(1). $(8\ points)$
Please determine the time complexity (Big-Theta) of the greedy way mentioned above. (Brief explanation is required.)

(2). $(15\ points)$
Please prove or disprove the greedy method. Proof requires completeness, disproof requires counter-example otherwise.

---

2. Lexicographical order

Given two different sequences of the same length, $a_1 a_2 \ldots a_k$ and $b_1 b_2 \ldots b_k$, the first one is smaller than the second one for the lexicographical order, if $a_i < b_i$ (for the order of alphabet), for the first i where a_i and b_i differ.
For example, "a" is smaller than "b", "beddc" is greater than "beczz".

To compare sequences of different lengths, the shorter sequence is usually padded at the end with enough blanks (a special symbol that is treated as smaller than any other alphabet).
For example, "aa" is smaller than "aaa".

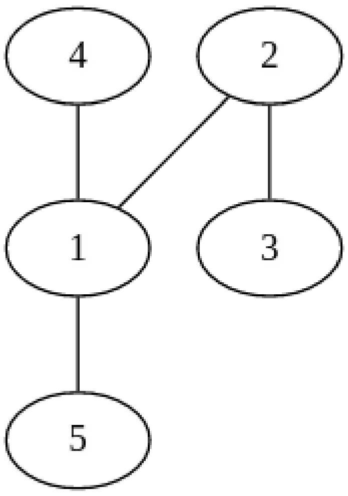
(1). $(5\ points)$
Fill in the blanks using $>$ or $<$.

card _________ cast
ntu _________ ntust
ntu _________ ntnu

When applied to sequences of numbers, lexicographical order is to compare the numerical value of elements. For example, the permutations of $\{1, 22, 18\}$ in lexicographical order are
$1, 18, 22 < 1, 22, 18 < 18, 1, 22 < 18, 22, 1 < 22, 1, 18 < 22, 18, 1$.

Now, given an undirected acyclic graph, each node has a number on it. You can choose any node to start traveling the whole graph, but you must travel it by Depth-First Search (Depth-First Traversal).

In the following graph, both $2, 1, 4, 5, 3$ and $3, 2, 1, 4, 5$ are valid DFS traversal, and $4, 1, 2, 5, 3$ is not.



Similarly, we can compare two traversals by lexicographical order, for example, $2, 1, 4, 5, 3$ is smaller than $3, 2, 1, 4, 5$.

(2). $(8\ points)$
In the above graph, to make the minimum lexicographical order DFS traversal, which node should you pick to start with? **Explain why this choice is correct.**

(3.) $(8\ points)$
A node v has one or more adjacent nodes, denoted by $N(v)$. Assuming the complexity of sorting the adjacent nodes of node v is $O(N(v)\log N(v))$, please prove that $\sum_{i=1}^{n} N(i)\log N(i)$ (the complexity of sorting the adjacent nodes of each node if there are n nodes) $\in O(n\log n)$.

(4.) $(16\ points)$
Design a greedy algorithm to find the minimum lexicographical order DFS traversal in the undirected acyclic graph that contains n nodes numbered from 1 to n.
Prove the correctness of your strategy (why your greedy choice is right?) and describe the time complexity of your algorithm. To get a full score, you have to find a $o(n^2)$ solution.
Make sure to clearly describe your answer, and you can use the answer in (1). (2). (3). to help you complete the explaining.

---

3. Arithmetic

> Sometime, even you come up with the correct algorithm by greedy, it's still hard to prove the correctness of it.

There are n numbers denoted by $a_1, a_2, \ldots, a_n$, we can sum up two numbers x, y and get $x + y$ with cost $x + y$. Not until all the numbers are summed up, in other words, there is only one number left, we should choose two numbers and sum them up.

e.g. If there are $1, 2, 3$, there are three ways to sum them up.

#	Step 1	Step 2	Total Cost
1	1 + 2 = 3	3 + 3 = 6	3 + 6 = 9
2	1 + 3 = 4	2 + 4 = 6	4 + 6 = 10
3	2 + 3 = 5	1 + 5 = 6	5 + 6 = 11

Consequently, the minimum cost of summing $1, 2, 3$ up is 9.

(1).

> For time complexity analysis, you can directly use the conclusion you learned from the course "Data Structure".

(a). $(5\ points)$
Please find out a strategy to sum n numbers. **Better time complexity solution could get a higher score.**

(b). $(10\ points)$
Please analyze the time complexity of your strategy(using Big-Theta).

(2). $(25\ points)$
Please prove the solution you write down in the previous problem is correct. Show your work as much as possible, any effort may be given a partial score.

Hint: Maybe you can imitate the way Professor proved the correctness of Huffman Code.

Programming

pA: Board Game

> Time Limit: 1s
> Memory Limit: 134218KB

Description

> We have a similar problem in the handwritten part, which might be helpful to solve this programming problem.

Fuji likes to play board games with her friends, but during the pandemic, everyone has to stay at home and can't play board games together.
Eventually, Fuji decides to play a board game that supports only one player, and the following is the game:

This board game has a main map, and there are N villages on the map, and each of them has a number which is between 1 to N, moreover, all the numbers are distinct.

At first, the player can choose a village to start, then, when visiting a village **for the first time**, the player can get the number of that village.
You have to visit all the villages, but you can only travel the map via DFS. (as a board game master, it's reasonable that Fuji knows some algorithm)

The most special part of this board game is about scoring, the order of numbers which player got from villages is important!
Formally, if the order of numbers that the player got **has a bigger lexicographical order**, the higher score the player can get!

Given a map, help Fuji to conquer the board game with the highest scores!

Same as before, the Hint section contains explanations of sample input/output.

Input Format

The first line contains an integer N, indicating the number of villages on the map.
For the next $N-1$ lines, each line contains two integers u_i and v_i, indicating there is a road between the village with the number u_i and the village with v_i.

For all test data, it is guaranteed:

- $1 \leq N \leq 10^5$
- $1 \leq u_i, v_i \leq N$
- $u_i \neq v_i$

Subtask1 (50%)

- $1 \leq N \leq 2000$

Subtask2 (50%)

- No other restrictions.

Output Format

Output N numbers in the order from first to last you got from villages, which can get the highest score.

Sample Input

```
3
1 2
1 3
```

Sample Output

```
3 1 2
```

Sample Input

```
7
1 2
1 7
1 6
2 3
3 4
2 5
```

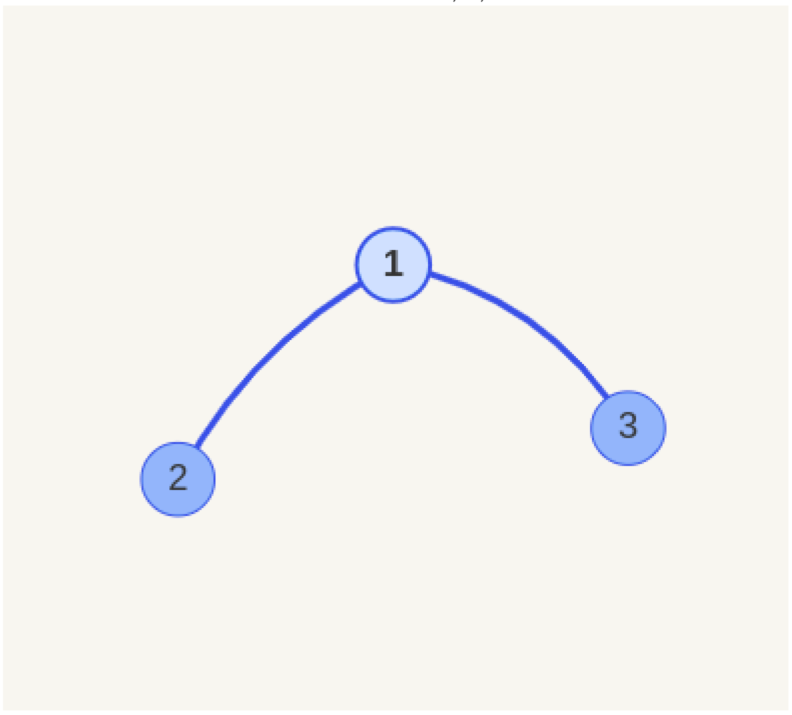
Sample Output

```
7 1 6 2 5 3 4
```

Hint

In the first sample, three villages with $1, 2, 3$ and two roads are given.
All possible routes to visit villages are:

- $1 \to 2 \to 1 \to 3$, and obtained numbers $1, 2, 3$
- $1 \to 3 \to 1 \to 2$, and obtained numbers $1, 3, 2$
- $2 \to 1 \to 3$, and obtained numbers $2, 1, 3$
- $3 \to 1 \to 2$, and obtained numbers $3, 1, 2$



pB: Fireworks

> Time Limit: 1s
> Memory Limit: 134218KB

Description

Bogay, one of the greatest wizards in NTNU, bought another Mastery Book, which is about using magic to launch fireworks!
He is doing this so well that it attracts lots of tourists to come to watch the beautiful fireworks.

A flock of tourists will come at the beginning of $l_i - th$ day and leave at the end of $r_i - th$ day.
Bogay is a nice wizard and wants to let everyone can watch fireworks **at least twice**.
However, to launch fireworks is dead tired, Bogay can only launch it at most once in a single day.

Now, given the schedule of the coming N flocks of tourists, help Bogay to plan so that he can launch the minimum times of fireworks and let all the tourists can watch fireworks twice.

All of the tourists will stay at least two days, so a solution is always exist.

Input Format

The first line contains an integer N, indicating there are N flocks of tourists coming.
For the next N lines, each line contains two integers l_i and r_i, indicating the $i - th$ flock of tourist will come from at $l_i - th$ day and leave at $r_i - th$ day.

For all test data, it is guaranteed:

- $1 \leq N \leq 10^5$
- $1 \leq l_i < r_i \leq 10^9$

Subtask1 (25%)

- $1 \leq N \leq 100$
- $1 \leq l_i < r_i \leq 10$

Subtask2 (75%)

- No other restrictions.

Output Format

Output An integer, indicating the minimum number of launching that Bogay has to do.

Sample Input

```
5
2 3
1 3
4 9
2 8
10 12
```

Sample Output

```
6
```

Sample Input

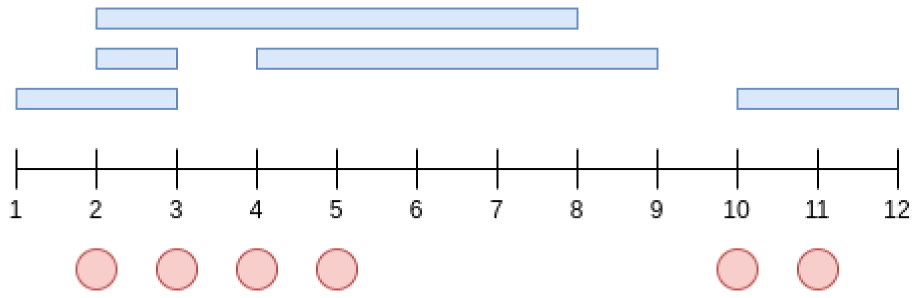
```
5
1 3
3 7
5 7
6 8
7 10
```

Sample Output

```
5
```

Hint

In the sample, at least 6 times of launching to let every group watch at least twice.



Because the input files are large, please add

```
std::ios_base::sync_with_stdio(false);
std::cin.tie(nullptr);
```

to the beginning of the main function if you are using std::cin.

Bonus: Simple Problem

> Time Limit: 1s
> Memory Limit: 268436KB

Description

We've learned **Inversion** and **Lexicographical order** at *HW2* and *HW4*.
This simple problem is: in all permutation of $\{1, 2, \ldots, N\}$, which permutation is the smallest for the lexicographical order, among the permutations that have **exactly** M inversions.

Print the permutation.

Input Format

A single line that contains two space-separated integers N, M.

For all test data, it is guaranteed:

- $1 \leq N \leq 10^5$
- $0 \leq M \leq \frac{N(N-1)}{2}$

Subtask1 (100%)

- No other restrictions.

Output Format

Please print that permutation, the elements should be separated by a single space.

Sample Input

```
4 2
```

Sample Output

```
1 3 4 2
```

Hint

The permutations of $\{1, 2, 3, 4\}$ that have exactly M inversions are:

- $1, 3, 4, 2$
- $1, 4, 2, 3$
- $2, 1, 4, 3$
- $2, 3, 1, 4$
- $3, 1, 2, 4$

The smallest permutation is $1, 3, 4, 2$.