[SOT_SUM25] Combinatorics & Graph Theory

Vo Ngoc Tram Anh

May 5, 2025

Contents

Contents **1**

1 Basic Combinatorics **2**
1.1 Inclusion-Exclusion Principle 2
1.2 Mathematical induction & recurrence 2
1.3 Pigeonhole principle & Ramsey theory 2
1.4 Counting rules & Stirling number of type 1 & type 2 2
1.5 Permutation & Combination 2
1.5.1 Consecutive 2 Dice Rolls 2
1.5.2 Simultaneous 2 Dice Rolls 4
1.5.3 Consecutive `n` Dice Rolls 6
1.5.4 Prime and Composite 6
1.5.5 Even and Odd 7

2 Basic Graph Theory **9**
2.1 Graph representation 9
2.2 Search Algorithm 9
2.2.1 Basic DFS & BFS 9
2.2.2 DFS & BFS for grid 9
2.2.3 Important algorithms 9

3 CSES Problem List **10**
3.1 Graph Algorithms 10
3.1.1 Counting Rooms 10
3.1.2 Labyrinth 12
3.1.3 Building Roads 13
3.1.4 Message Routes 14
3.1.5 Building Teams 15
3.1.6 Round Trip 15
3.1.7 Monsters 15

1 Basic Combinatorics

1.1 Inclusion-Exclusion Principle

1.2 Mathematical induction & recurrence

1.3 Pigeonhole principle & Ramsey theory

1.4 Counting rules & Stirling number of type 1 & type 2

1.5 Permutation & Combination

1.5.1 Consecutive 2 Dice Rolls

https://github.com/vntanh1406/Graph_SUM2025/blob/main/Combinatorics/Consecutive2DiceRolls.cpp

Let the sample space be denoted by Ω.

We consider the experiment of rolling two distinguishable six-sided dice in sequence. Each die has 6 possible outcomes, and since the rolls are independent and ordered, the sample space consists of all ordered pairs (i, j) where $i, j \in \{1, 2, 3, 4, 5, 6\}$.

Therefore, the total number of outcomes is: $|\Omega| = 6 \cdot 6 = 36$

a)

- Let A_1 be the event that both dice show the same number of dots. This corresponds to the set of outcomes: $A_1 = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}$

  So, $P(A_1) = \frac{6}{36} = \frac{1}{6}$

- Let A_2 be the event that both dice show different numbers of dots. Since there are 6 outcomes with the same number, there are: $36 - 6 = 30$ outcomes with different numbers

  So: $P(A_2) = \frac{30}{36} = \frac{5}{6}$

b)

Even numbers on a die: $\{2, 4, 6\}$

Odd numbers on a die: $\{1, 3, 5\}$

- Let B_1 be the event that both dice have the same parity:
  - Both even: $3 \cdot 3 = 9$ outcomes
  - Both odd: $3 \cdot 3 = 9$ outcomes

  Total favorable outcomes: $9 + 9 = 18$. So: $P(B_1) = \frac{18}{36} = \frac{1}{2}$

- Let B_2 be the event that the two numbers have different parity:
  - First even, second odd: $3 \cdot 3 = 9$ outcomes
  - First odd, second even: $3 \cdot 3 = 9$ outcomes

  Total favorable outcomes: $9 + 9 = 18$. So: $P(B_2) = \frac{18}{36} = \frac{1}{2}$

c)

Prime numbers on a die: $\{2, 3, 5\}$

Composite numbers on a die: $\{4, 6\}$

- Let C_1 be the event that the numbers on both dice are prime numbers:

  Both prime: $3 \cdot 3 = 9$ outcomes. So: $P(C_1) = \frac{9}{36} = \frac{1}{4}$

- Let C_2 be the event that the numbers on both dice are composite numbers:

  Both composite: $2 \cdot 2 = 4$ outcomes. So: $P(C_2) = \frac{4}{36} = \frac{1}{9}$

- Let C_3 be the event that at least one of the two dice shows a prime number.

  The number of outcomes where neither die shows a prime number, or both dice show non-prime numbers: $3 \cdot 3 = 9$ *(non-prime: {1, 4, 6})*

  Therefore, the number of favorable outcomes: $36 - 9 = 27$. So: $P(C_3) = \frac{27}{36} = \frac{3}{4}$

- Let C_4 be the event that at least one of the two dice shows a composite number.

  The number of outcomes where neither die shows a composite number, or both dice show non-composite numbers: $4 \cdot 4 = 16$ *(non-composite: {1, 2, 3, 5})*

  Therefore, the number of favorable outcomes: $36 - 16 = 20$. So: $P(C_4) = \frac{20}{36} = \frac{5}{9}$

d)
Let D be the event that one of the two numbers is a divisor or a multiple of the other.
Let each outcome be represented as an ordered pair (a, b), where $a, b \in \{1, 2, 3, 4, 5, 6\}$.
We are interested in counting the number of outcomes where: $a \mid b \quad \text{or} \quad b \mid a$
The valid pairs:

- When $a = 1$: $b = 1, 2, 3, 4, 5, 6$ (6 outcomes)
- When $a = 2$: $b = 1, 2, 4, 6$ (4 outcomes)
- When $a = 3$: $b = 1, 3, 6$ (3 outcomes)
- When $a = 4$: $b = 1, 2, 4$ (3 outcomes)
- When $a = 5$: $b = 1, 5$ (2 outcomes)
- When $a = 6$: $b = 1, 2, 3, 6$ (4 outcomes)

Total favorable outcomes: $6 + 4 + 3 + 3 + 2 + 4 = 22$. So: $P(D) = \frac{22}{36} = \frac{11}{18}$

e)
Let E_n be the event that the sum of the two dice is equal to n.
For $n \in \{2, 3, \ldots, 12\}$, we define the function: $f(n) = \min\{n - 1, 6\} - \max\{n - 6, 1\} + 1$
This function counts the number of integer pairs $(a, b) \in \{1, 2, 3, 4, 5, 6\}^2$ such that $a + b = n$.

- The smallest possible sum is $1 + 1 = 2$, and the largest is $6 + 6 = 12$, so $n \in \{2, 3, \ldots, 12\}$.
- For a fixed n, valid pairs (a, b) must satisfy: $a \in [\max(1, n - 6), \min(6, n - 1)]$, and then $b = n - a$.
- Therefore, the number of such values of a is: $f(n) = \min(n - 1, 6) - \max(n - 6, 1) + 1$
- This formula works because:
  - $\min(n - 1, 6)$ gives the largest possible value of a such that $b = n - a \geq 1$
  - $\max(n - 6, 1)$ gives the smallest possible value of a such that $b = n - a \leq 6$
  - The total number of integers a in that interval is: upper bound $-$ lower bound $+1$

So: $P(E_n) = \frac{f(n)}{36} \cdot \mathbf{1}_{\{2 \leq n \leq 12\}}$

1.5.2 Simultaneous 2 Dice Rolls

https://github.com/vntanh1406/Graph_SUM2025/blob/main/Combinatorics/Consecutive2DiceRolls.cpp

Let the sample space be denoted by Ω.

We consider the experiment of rolling two indistinguishable six-sided dice simultaneously. Each die has 6 possible outcomes, and since the dice are indistinguishable and rolled at the same time, the sample space consists of all unordered pairs (i, j) where $i, j \in \{1, 2, 3, 4, 5, 6\}$ and $i \leq j$.

Therefore, the total number of outcomes is: $|\Omega| = 6 + \binom{6}{2} = 21$

a)

- Let A_1 be the event that both dice show the same number of dots. This corresponds to the set of outcomes: $A_1 = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}$

  So, $P(A_1) = \frac{6}{21} = \frac{2}{7}$

- Let A_2 be the event that both dice show different numbers of dots. Since there are 6 outcomes with the same number, there are: $21 - 6 = 15$ outcomes with different numbers

  So: $P(A_2) = \frac{15}{21} = \frac{5}{7}$

b)

Even numbers on a die: $\{2, 4, 6\}$

Odd numbers on a die: $\{1, 3, 5\}$

- Let B_1 be the event that both dice have the same parity:
  - Both even: $3 + \binom{3}{2} = 6$ outcomes
  - Both odd: $3 + \binom{3}{2} = 6$ outcomes

  Total favorable outcomes: $6 + 6 = 12$. So: $P(B_1) = \frac{12}{21} = \frac{4}{7}$

- Let B_2 be the event that the two numbers have different parity:

  Total favorable outcomes: $3 \cdot 3 = 9$. So: $P(B_2) = \frac{9}{21} = \frac{3}{7}$

c)

Prime numbers on a die: $\{2, 3, 5\}$

Composite numbers on a die: $\{4, 6\}$

- Let C_1 be the event that the numbers on both dice are prime numbers:

  Both prime: $3 + \binom{3}{2} = 6$ outcomes. So: $P(C_1) = \frac{6}{21} = \frac{2}{7}$

- Let C_2 be the event that the numbers on both dice are composite numbers:

  Both composite: $2 + \binom{2}{2} = 3$ outcomes. So: $P(C_2) = \frac{3}{21} = \frac{1}{7}$

- Let C_3 be the event that at least one of the two dice shows a prime number.

  The number of outcomes where neither die shows a prime number, or both dice show non-prime numbers: $3 + \binom{3}{2} = 6$ *(non-prime: {1, 4, 6})*

  Therefore, the number of favorable outcomes: $21 - 6 = 15$. So: $P(C_3) = \frac{15}{21} = \frac{5}{7}$

- Let C_4 be the event that at least one of the two dice shows a composite number.

  The number of outcomes where neither die shows a composite number, or both dice show non-composite numbers: $4 + \binom{4}{2} = 10$ *(non-composite: {1, 2, 3, 5})*

  Therefore, the number of favorable outcomes: $21 - 10 = 11$. So: $P(C_4) = \frac{11}{21}$

d)
Let D be the event that one of the two numbers is a divisor or a multiple of the other.
Let each outcome be represented as an unordered pair (a, b), where $a, b \in \{1, 2, 3, 4, 5, 6\}$.
We are interested in counting the number of outcomes where: $a \mid b \quad \text{or} \quad b \mid a$
The valid pairs:

- When $a = 1$: $b = 1, 2, 3, 4, 5, 6$ (6 outcomes)
- When $a = 2$: $b = 2, 4, 6$ (3 outcomes)
- When $a = 3$: $b = 3, 6$ (2 outcomes)
- When $a = 4$: $b = 4$ (1 outcome)
- When $a = 5$: $b = 5$ (1 outcome)
- When $a = 6$: $b = 6$ (1 outcome)

Total favorable outcomes: $6 + 3 + 2 + 1 + 1 + 1 = 14$. So: $P(D) = \frac{14}{21} = \frac{2}{3}$

e)
Let E_n be the event that the sum of the two dice is equal to n.
For $n \in \{2, 3, \ldots, 12\}$, we define the function: $f(n) = \left\lfloor \frac{\min\{n-1,6\} - \max\{n-6,1\}}{2} \right\rfloor + 1$
This function counts the number of unordered integer pairs $(a, b) \in \{1, 2, 3, 4, 5, 6\}^2$ such that $a + b = n$ and $a \leq b$.

- The smallest possible sum is $1+1 = 2$, and the largest is $6+6 = 12$, so $n \in \{2, 3, \ldots, 12\}$.
- For a fixed n, valid unordered pairs (a, b) must satisfy:

$$a \in \left[\max(1, n-6), \left\lfloor \frac{n}{2} \right\rfloor\right], \quad b = n - a, \quad \text{and } a \leq b$$

- Therefore, the number of such values of a is given by:

$$f(n) = \left\lfloor \frac{\min(n-1, 6) - \max(n-6, 1)}{2} \right\rfloor + 1$$

- This formula works because:
  - $\min(n-1, 6)$ gives the largest possible value of a such that $b = n - a \in [1, 6]$
  - $\max(n-6, 1)$ gives the smallest possible value of a such that $b = n - a \in [1, 6]$
  - We divide the range by 2 and take floor to count only unordered pairs (i.e., $a \leq b$)
  - Adding 1 accounts for inclusive bounds

So: $P(E_n) = \frac{f(n)}{21} \cdot \mathbf{1}_{\{2 \leq n \leq 12\}}$.

1.5.3 Consecutive n Dice Rolls

https://github.com/vntanh1406/Graph_SUM2025/blob/main/Combinatorics/ConsecutiveNDiceRolls.ipynb

Let the sample space be denoted by Ω.

We consider the experiment of rolling `n` distinguishable six-sided dice in sequence. Each die has 6 possible outcomes, and since the rolls are independent and ordered, the sample space consists of all ordered pairs (i, j) where $i, j \in \{1, 2, 3, 4, 5, 6\}$.

Therefore, the total number of outcomes is: $|\Omega| = 6^n$

a) Let A be the event that all dice show the same number of dots.
This corresponds to the set of outcomes where $x_1 = x_2 = \cdots = x_n$.
There are exactly 6 such outcomes (all 1's, all 2's, ..., all 6's), so: $P(A) = \frac{6}{6^n}$

b) Let B be the event that all dice show different numbers of dots.
This is only possible when $1 \leq n \leq 6$ *(since there are only 6 distinct values from 1 to 6).*

- If $n > 6$ or $n < 2$, then clearly: $P(B) = 0$
- For $2 \leq n \leq 6$: The number of favorable outcomes as the number of one-to-one mappings from n dice to 6 values, i.e., number of permutations: $P(6, n) = 6 \cdot 5 \cdot 4 \cdots (6 - n + 1)$. So: $P(B) = \frac{P(6,n)}{6^n} = \frac{6!}{(6-n)! \cdot 6^n}$

c) Let C be the event that all dice have the same parity.

- All even: 3^n outcomes
- All odd: 3^n outcomes

Total favorable outcomes: $2 \cdot 3^n$. So: $P(C) = \frac{2 \cdot 3^n}{6^n} = \frac{1}{2^{n-1}}$

1.5.4 Prime and Composite

https://github.com/vntanh1406/Graph_SUM2025/blob/main/Combinatorics/PrimeAndComposite.ipynb

Let $A_n = \{1, 2, \ldots, n\} \subset N^*$ be the set of the first n positive integers.
Let $m \in N^*$, and suppose we randomly select m distinct elements from A_n.
Let $T = \binom{n}{m}$ be the total number of ways to choose m distinct elements from A_n.
Suppose $k \in \{0, 1, \ldots, m\}$

a)
Let:

- $d_e = \left\lfloor \dfrac{n}{2} \right\rfloor$ be the number of even numbers in A_n,
- $d_o = n - d_e$ be the number of odd numbers in A_n,

We have the following probabilities:

- $P_{\text{All even}} = \frac{\binom{d_e}{m}}{T}$
- $P_{\text{All odd}} = \frac{\binom{d_o}{m}}{T}$

- $P_{\text{At least one even}} = 1 - P_{\text{All odd}} = 1 - \frac{\binom{d_o}{m}}{T}$

- $P_{\text{At least one odd}} = 1 - P_{\text{All even}} = 1 - \frac{\binom{d_e}{m}}{T}$

- $P_{\text{Exactly k even}} = \frac{\binom{d_e}{k} \cdot \binom{d_o}{m-k}}{T}$

- $P_{\text{Exactly k odd}} = \frac{\binom{d_o}{k} \cdot \binom{d_e}{m-k}}{T}$

- $P_{\text{At least k even}} = \frac{1}{T} \sum_{i=k}^{m} \binom{d_e}{i} \cdot \binom{d_o}{m-i}$

- $P_{\text{At least k odd}} = \frac{1}{T} \sum_{i=k}^{m} \binom{d_o}{i} \cdot \binom{d_e}{m-i}$

b)
Let:

- $p = \pi(n)$: the number of prime numbers less than or equal to n

- $c = n - 1 - \pi(n)$: the number of composite numbers in A_n

We have the following probabilities:

- $P_{\text{All prime}} = \frac{\binom{p}{m}}{T}$

- $P_{\text{All composite}} = \frac{\binom{c}{m}}{T}$

- $P_{\text{At least one prime}} = \frac{1}{T} \sum_{i=1}^{m} \binom{p}{i} \cdot \binom{n-p}{m-i} = 1 - \frac{\binom{c}{m} + \binom{c}{m-1}}{T}$

- $P_{\text{At least one composite}} = \frac{1}{T} \sum_{i=1}^{m} \binom{c}{i} \cdot \binom{n-c}{m-i} = 1 - \frac{\binom{p}{m} + \binom{p}{m-1}}{T}$

- $P_{\text{Exactly k prime}} = \frac{\binom{p}{k} \cdot \binom{n-p}{m-k}}{T}$

- $P_{\text{Exactly k composite}} = \frac{\binom{c}{k} \cdot \binom{n-c}{m-k}}{T}$

- $P_{\text{At least k prime}} = \frac{1}{T} \sum_{i=k}^{m} \binom{p}{i} \cdot \binom{n-p}{m-i}$

- $P_{\text{At least k composite}} = \frac{1}{T} \sum_{i=k}^{m} \binom{c}{i} \cdot \binom{n-c}{m-i}$

1.5.5 Even and Odd

https://github.com/vntanh1406/Graph_SUM2025/blob/main/Combinatorics/EvenAndOdd.ipynb

Let $a, b \in Z$ with $a < b$, and let $n, k \in N^*$ such that $n \geq 2$ and $k \leq n$.
Define the set $A = \{a, a+1, a+2, \ldots, b\} \subset Z$ with total size $N = |A| = b - a + 1$.
Let:

- $d_o = \left\lfloor \frac{b-a}{2} \right\rfloor + 1$ be the number of odd numbers in A_n

- $d_e = N - d_o$

- T be the total number of possible selections.

a)

- **Distinct** ($T = \binom{N}{2}$):
  - Probability that both numbers have the same parity: $P_{\text{same parity}} = \frac{\binom{d_e}{2}+\binom{d_o}{2}}{T}$
  - Probability that the two numbers have different parity: $P_{\text{different parity}} = \frac{d_e \cdot d_o}{T}$
- **With replacement** ($T = N^2$):
  - Probability that both numbers have the same parity: $P_{\text{same parity}} = \frac{{d_e}^2+{d_o}^2}{T}$
  - Probability that the two numbers have different parity: $P_{\text{different parity}} = \frac{2 \cdot d_e \cdot d_o}{T}$

b)
Suppose $d_e, d_o \geq n$

- **Distinct** ($T = \binom{N}{n}$):
  - $P_{\text{all even}} = \dfrac{\binom{d_e}{n}}{T}$
  - $P_{\text{all odd}} = \dfrac{\binom{d_o}{n}}{T}$
  - $P_{\text{same parity}} = \dfrac{\binom{d_e}{n} + \binom{d_o}{n}}{T}$
  - $P_{\text{exactly } k \text{ even}} = \dfrac{\binom{d_e}{k} \cdot \binom{d_o}{n-k}}{T}$
  - $P_{\text{exactly } k \text{ odd}} = \dfrac{\binom{d_o}{k} \cdot \binom{d_e}{n-k}}{T}$
  - $P_{\text{at least } k \text{ even}} = \dfrac{1}{T} \sum_{i=k}^{n} \binom{d_e}{i} \cdot \binom{d_o}{n-i}$
  - $P_{\text{at least } k \text{ odd}} = \dfrac{1}{T} \sum_{i=k}^{n} \binom{d_o}{i} \cdot \binom{d_e}{n-i}$
- **With replacement** ($T = N^n$):
  - $P_{\text{all even}} = \dfrac{{d_e}^n}{T}$
  - $P_{\text{all odd}} = \dfrac{{d_o}^n}{T}$
  - $P_{\text{same parity}} = \dfrac{{d_e}^n + {d_o}^n}{T}$
  - $P_{\text{exactly } k \text{ even}} = \dfrac{\binom{n}{k} \cdot {d_e}^k \cdot {d_o}^{n-k}}{T}$
  - $P_{\text{exactly } k \text{ odd}} = \dfrac{\binom{n}{k} \cdot {d_o}^k \cdot {d_e}^{n-k}}{T}$
  - $P_{\text{at least } k \text{ even}} = \dfrac{1}{T} \sum_{i=k}^{n} \binom{n}{i} \cdot {d_e}^i \cdot {d_o}^{n-i}$
  - $P_{\text{at least } k \text{ odd}} = \dfrac{1}{T} \sum_{i=k}^{n} \binom{n}{i} \cdot {d_o}^i \cdot {d_e}^{n-i}$

2 Basic Graph Theory

Link to C++ Sources: https://github.com/vntanh1406/Graph_SUM2025/tree/main/BasicGraphTheory

2.1 Graph representation

- Adjacency Matrix to Edge List : https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/AdjacencyMatrixToEdgeList.cpp
- Adjacency Matrix to Adjacency List: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/AdjacencyMatrixToAdjacencyList.cpp
- Edge List to Adjacency Matrix: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/EdgeListToAdjacencyMatrix.cpp
- Edge List to Adjacency List: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/EdgeListToAdjacencyList.cpp
- Adjacency List to Adjacency Matrix: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/AdjacencyListToAdjacencyMatrix.cpp
- Adjacency List To Edge List: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/AdjacencyListToEdgeList.cpp

2.2 Search Algorithm

2.2.1 Basic DFS & BFS

- Basic Depth First Search: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/BasicDFS.cpp
- Basic Breadth First Search: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/BasicBFS.cpp
- Counting Connected Components: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/CountingConnectedComponents.cpp
- Find Path From `s` to `e`: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/FindPath_Basic.cpp

2.2.2 DFS & BFS for grid

- Counting Connected Components and Checking Path Existence: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/BFS_DFS_OnGrid.cpp
- Find The Shortest Path: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/FindShortestPath.cpp

2.2.3 Important algorithms

- Topological Sort: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/TopologicalSort.cpp
- Detect Cycles in Undirected Graph: https://github.com/vntanh1406/Graph_SUM2025/blob/main/BasicGraphTheory/UndirectedGraphCycle.cpp

3 CSES Problem List

3.1 Graph Algorithms

3.1.1 Counting Rooms

Problem Description

- https://cses.fi/problemset/task/1192
- The task is to determine the number of rooms in a building.
- A room is defined as a maximal connected area of floor tiles (denoted by `'.'`) in a 2D map.
- We can move up, down, left, and right between adjacent floor tiles. Wall tiles are represented by `'#'` and cannot be walked through.
- **Input:**
  - The first line contains two integers n and m ($1 \leq n, m \leq 1000$), denoting the height and width of the map.
  - The next n lines each contain a string of m characters representing the map.
- **Output:** The number of distinct rooms.
- **Example**

Input	Output

`5 8`
`########`
`#..#...#`
`####.#.#`
`#..#...#`
`########`
	3

Algorithm Explanation

- https://cses.fi/paste/6fa005ccb9388ed1c2d9a9/
- Use DFS to find the number of connected components in a grid where we treat floor tiles `'.'` as nodes in a graph. (2 floor tiles are connected if they are adjacent: up/down/left-/right).
- Implementation:
  - `n, m`: the dimensions of the grid (height and width).
  - `grid`: a vector of strings representing the map, where each character is either `'.'` (floor) or `'#'` (wall).
  - `visited[n][m]`: a 2D boolean array used to track visited floor tiles.
  - `dx[4], dy[4]`: Two arrays representing the relative movement in four directions:
    * Up: $(-1, 0)$
    * Down: $(+1, 0)$

        - $*$ Left: $(0, -1)$
        - $*$ Right: $(0, +1)$
    - **void dfs(int x, int y)**
        - $*$ Given a starting floor tile at position (x, y):
            1. Mark `visited[x][y] = true`.
            2. For each of the 4 directions:
                - New coordinates $(nx, ny) = (x + dx[i], y + dy[i])$.
                - If (nx, ny) is within bounds, not visited, and is a floor tile, recursively call `dfs(nx, ny)`.
    - **Main loop:**
        - $*$ Iterate over all grid cells.
        - $*$ For each unvisited floor tile:
            - Call DFS from that tile to explore its connected room.
            - Increment the room counter.

- **Time complexity:** The algorithm visits each cell at most once, and each DFS runs in time proportional to the number of floor tiles in a room. Hence, the overall complexity is $\mathcal{O}(n \cdot m)$.

- **Step-by-Step Example** (using the sample input): We visualize the grid and track DFS visits:

  Initial Grid:

#	#	#	#	#	#	#	#
#	.	.	#	.	.	.	#
#	#	#	#	.	#	.	#
#	.	.	#	.	.	.	#
#	#	#	#	#	#	#	#

—

Room 1: Start DFS at (1,1)

DFS visits: (1,1) → (1,2)

→ Room 1 completed. Total rooms = 1

Room 2: Start DFS at (1,4)

DFS visits: (1,4) → (2,4) → (3,4) → (3,5) → (3,6) → (2,6) → (1,6) → (1, 5)

→ Room 2 completed. Total rooms = 2

Room 3: Start DFS at (3,1)

DFS visits: (3,1) → (3,2)

→ Room 3 completed. Total rooms = 3

3.1.2 Labyrinth

Problem Description

- https://cses.fi/problemset/task/1193
- There is a map of a labyrinth, and we need to find the shortest path from a start point `A` to an endpoint `B`.
- Movement is allowed in four directions: up, down, left, right.
- The labyrinth is represented by a grid:
  - `'.'` denotes an empty tile (floor).
  - `'#'` denotes a wall.
  - `'A'` is the starting point.
  - `'B'` is the target.
- **Input:**
  - The first line contains two integers n and m ($1 \leq n, m \leq 1000$), the dimensions of the map.
  - The next n lines contain m characters each, describing the map.
- **Output:**
  - First print "YES" if a path exists, and "NO" otherwise.
  - If a path exists, print its length and then a string consisting of the steps: `L`, `R`, `U`, `D`.
- **Example**

Input	Output

`5 8`
`########`
`#.A#...#`
`#.##.#B#`
`#......#`
`########`
	`YES`
	`9`
	`LDDRRRRRU`

Algorithm Explanation:

- https://cses.fi/paste/261fd1d6d36a0c05c32743/
- **Initialize:**
  - `visited[i][j]` = false for all cells
  - `d[i][j]` = 0 (distance from 'A')
  - `parent[i][j]` = previous cell in path
- **BFS(start):**
  - Enqueue start cell, mark as visited
  - While queue not empty:

* Dequeue (x, y)
* For each direction (U, L, R, D): If neighbor (nx, ny) is valid and unvisited:
  - Mark visited, set parent, update distance
  - If cell is 'B': stop search

- **Trace Path:**
  - If distance to 'B' is 0: print "NO"
  - Else:
    * Backtrack from 'B' to 'A' using `parent`
    * Record directions (U, L, R, D)
    * Reverse the path and print "YES", distance, and path
- **Time Complexity:** $\mathcal{O}(n \cdot m)$

3.1.3 Building Roads

Problem Description

- https://cses.fi/problemset/task/1666
- Given n cities and m roads, determine the minimum number of new roads required to connect all cities, and specify which roads to build. Each existing road connects two different cities.
- **Input:**
  - The first line contains two integers n and m $(1 \leq n \leq 10^5, 1 \leq m \leq 2 \cdot 10^5)$ (the number of cities and existing roads).
  - The next m lines contain two integers a and b $(1 \leq a, b \leq n)$ (meaning there is a road between cities a and b).
- **Output:**
  - First, print an integer k (the minimum number of new roads needed).
  - Then print k lines, each containing two integers u and v, indicating a road to build between cities u and v.
  - Any valid solution is accepted.
  - **Example**

Input	Output

`4 2`
`1 2`
`3 4`
	`1`
	`2 3`

Algorithm Explanation

- https://cses.fi/paste/bdeee055189e59e5c2f5d8/
- Use DFS to find all connected components.
- For each new component found, save a representative city.

- To connect the components:
  - If there are k components, we need $k-1$ roads.
  - Connect representative cities linearly: $res[i]$ with $res[i+1]$.
- **Time Complexity:** $\mathcal{O}(n+m)$

3.1.4 Message Routes

Problem Description

- https://cses.fi/problemset/task/1667
- n computers and m connections.
- Each connection links two distinct computers directly.
- Check if there is a path exists from computer 1 to computer n, find the minimum number of computers on the route, and output one such route.
- **Input:**
  - The first line contains two integers n and m
  - Then follow m lines, each with two integers a and b: there is a connection between computers a and b.
- **Output:**
  - If there exists a route from computer 1 to computer n, first print k, then print k space-separated integers representing the computers along this path.
  - If no such route exists, print `IMPOSSIBLE`.
- **Example**

Input	Output
5 5 1 2 1 3 1 4 2 3 5 4	3 1 4 5

Algorithm Explanation

- https://cses.fi/paste/4e4274bd1cea72b6c2f69d/
- The problem is to find the shortest path from node `1` to node `n` in an undirected graph.
- Use BFS starting from node `1` to:
  - Mark visited nodes (`visited[i]`).
  - Record the parent of each node during traversal (`parent[i]`) to reconstruct the path.
  - Count the number of steps from the start node to each node (`d[i]`).

- After BFS:
  - If `visited[n]` is false, there is no path from `1` to `n`, so the output is `IMPOSSIBLE`.
  - Otherwise, we reconstruct the shortest path using the `parent[]` array starting from node `n` back to `1`.
  - Finally, we print the path length (`d[n] + 1`, because the path includes both endpoints), and the path in correct order.
- **Time Complexity:** $\mathcal{O}(n + m)$

3.1.5 Building Teams

Problem Description

- https://cses.fi/problemset/task/1668/

Algorithm Explanation

- https://cses.fi/paste/4ac88035ca891502c2f78e/

3.1.6 Round Trip

Problem Description

- https://cses.fi/problemset/task/1669

Algorithm Explanation

- https://cses.fi/paste/c716877ebfb8afaec307d9/

3.1.7 Monsters

Problem Description

- https://cses.fi/problemset/task/1194

Algorithm Explanation

- https://cses.fi/paste/1a35c0381d423f67c3084e/