A. Requirements

Code (90%)

You can write your code in Java, Python, C, or C++. The *time limit* may vary among different languages, depending on the performance of the language. Your code must be a complete excutable program instead of only a function. We guarantee test data strictly compliance with the requirements in the description, and you do not need to deal with cases where the input data is invalid.

Libraries in this assignment:

- For C/C++, you can only include standard library.
- For Java, you can only `import java.util.*`
- For Python, you can only import standard library. In other words, you cannot import libraries such as `numpy`.

We provide an example problem to illustrate the information above better.

Report (10%)

You also need to write a report in `pdf` type to explain the following:

- What are the possible solutions for the problem?
- How do you solve this problem?
- Why is your solution better than others?

Please note that the **maximum** number of pages allowed for your report is **5 pages**.

Remember that the report is to illustrate your thinking process. Keep in mind that your report is supposed to show your ideas and thinking process. We expect clear and precise textual descriptions in your report, and we do not recommend that you over-format your report.

B. Example Problem: A + B Problem

Description

Given 2 integers A and B, compute and print A + B

Input

Two integers in one line: A, and B

Output

One integer: A + B

Sample Input 1

```
1 2
```

Sample Output 1

```
3
```

Problem Scale & Subtasks

For 100% of the test cases, $0 \leq A, B \leq 10^6$

Solutions

Java

```java
import java.util.*;

public class Example {
    public static void main(String[] args) {
        int a, b;
        Scanner scanner = new Scanner(System.in);
        a = scanner.nextInt();
        b = scanner.nextInt();
        scanner.close();
        System.out.println(a + b);
    }
}
```

Python

```python
AB = input().split()
A, B = int(AB[0]), int(AB[1])
print(A + B)
```

C

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  int A, B;
  scanf("%d%d", &A, &B);
  printf("%d\n", A + B);
  return 0;
}
```

C++

```cpp
#include <iostream>

int main(int argc, char *argv[])
{
  int A, B;
  std::cin>> A >> B;
  std::cout<< A + B << std::endl;
  return 0;
}
```

C. Submission

After finishing this assignment, you are required to submit your code to the Online Judge System (OJ), and upload your `.zip` package of your code files and report to BlackBoard.

C.1 Online Judge

Once you have completed one problem, you can submit your code on the page on the Online Judge platform (oj.cuhk.edu.cn, campus only) to gain marks for the code part. You can submit your solution of one problem for **no more than 80 times**.

After you have submitted your program, OJ will test your program on all test cases and give you a grade. The grade of your latest submission will be regarded as the final grade of the corresponding problem. Each problem is tested on multiple test cases of different difficulty. You will get a part of the score even if your algorithm is not the best.

Note: The program running time may vary on different machines. Please refer to the result of the online judge system. OJ will show the time and memory limits for different languages on the corresponding problem page.

If you have other questions about the online judge system, please refer to OJ wiki (campus network only). If this cannot help you, feel free to contact us.

C.2 BlackBoard

You are required to upload your **source codes and report** to the BlackBoard platform. You need to name your files according to the following rules and compress them into `A4_<Student ID>.zip` :

```
A4_<Student ID>.zip
|-- A4_P1_<Student ID>.java/py/c/cpp
|-- A4_P2_<Student ID>.java/py/c/cpp
|-- A4_Report_<Student ID>.pdf
```

For Java users, **you don't need to consider the consistency of class name and file name**.

For example, suppose your ID is 123456789, and your problem 1 is written in `Python`, problem 2 is written in `Java` then the following contents should be included in your submitted `A4_123456789.zip`:

```
A4_123456789.zip
|-- A4_P1_123456789.py
|-- A4_P2_123456789.java
|-- A4_Report_123456789.pdf
```

C.3 Late Submissions

Submissions after Dec 13 2023 23:59:00(UTC+8) would be considered as LATE.

The LATE submission page will open after deadline on OJ.

Submisson time = max{latest submisson time for every problem, BlackBoard submisson time}

There will be penalties for late submission:

- 0–24 hours after deadline: final score = your score×0.8
- 24–72 hours after deadline: final score = your score×0.5
- 72+ hours after deadline: final score = your score×0

FAQs

Q: I cannot access to Online Judge.

A: First, please ensure that you are using the campus network. If you are not on campus, please use the university VPN. Second, please delete cookies and refresh browser or use other browser. If you still cannot access to Online Judge, try to visit it via the IP address `10.26.200.13`.

Q: My program passes samples on my computer, but not get AC on OJ.

A: Refer to OJ Wiki Q&A

Authors

If you have questions for the problems below, please contact:

- Problem 1. Shu Wang: shuwang3@link.cuhk.edu.cn
- Problem 2 & 3. Ziyi Zhao: ziyizhao2@link.cuhk.edu.cn

CSC3100 Data Structures Fall 2023

Programming Assignment 4

Due: Dec 13 2023 23:59:00

Assignment Link: http://oj.cuhk.edu.cn/contest/csc310023falla4

Access Code: `9v7Dxqet`

1 Divine Ingenuity (40% of this assignment)

1.1 Description

If you have ever played Genshin Impact, you must remember "Divine Ingenuity: Collector's Chapter" event. In this event, players can create custom *domains* by arranging components, including props and traps, between the given starting point and exit point.

Paimon does not want to design a difficult *domain*; she pursues the ultimate "automatic map". In the given domain with a size of $m \times n$, she only placed *Airflow* and *Spikes*. Specifically, *Spikes* will eliminate the player (represented by 'x'), while the *Airflow* will blow the player to the next position according to the wind direction (up, left, down, right represented by 'w', 'a', 's', 'd', respectively).

The starting point and exit point are denoted by 'i' and 'j', respectively. Ideally, in Paimon's *domain*, the player selects a direction and advances one position initially; afterward, the *Airflow* propels the player to the endpoint without falling into the *Spikes*. The player will achieve *automatic clearance* in such a domain.

However, Paimon, in her slight oversight, failed to create a domain that allows players to achieve *automatic clearance*. Please assist Paimon by making the minimum adjustments to her design to achieve *automatic clearance*.

Given that the positions of the starting point and exit point are fixed, you can only adjust components at other locations. You have the option to remove existing component at any position; then, place a new direction of *Airflow*, or position a *Spikes*.

1.2 Input

The first line of input contains two integers m and n, representing the size of the domain. m lines follow, each containing n characters. The characters must be one of 'w', 'a', 's', 'd', 'x', 'i' and 'j'. It is guaranteed that there is only one 'i' and 'j' on the map, and they are not adjacent.

1.3 Output

Output a single integer, representing the minimum number of changes needed.

Sample Input 1

```
3 3
dsi
ssd
jdd
```

Sample Output 1

```
1
```

You can make one modification to transform the domain as Fig. 2, allowing automatic clearance by choosing to move left initially.

→	↓	i
↓	↓	→
j	→	→

Figure 1: Original Domain in Sample 1

→	↓	i
↓	↓	→
j	←	→

Figure 2: Modified Domain in Sample 1

Sample Input 2

```
4 4
jxsx
xdxa
dxax
xwxi
```

Sample Output 2

```
4
```

You can make 4 modifications to transform the domain as Fig. 4, and choose to move upwards initially.

j	x	↓	x
x	→	x	←
↓	x	←	x
x	↑	x	i

Figure 3: Original Domain in Sample 2

j	←	←	x
x	→	↑	←
↓	x	←	↑
x	↑	x	i

Figure 4: Modified Domain in Sample 2

You can find More Sample in the attached file on BB.

Problem Scale & Subtasks

Test Case No.	Constraints
1-4	$3 \leq m, n \leq 20$
5-8	$3 \leq m, n \leq 500$
9-10	$3 \leq m, n \leq 2000$

Hint

Hint1: The player cannot move outside the domain, and initially, they can choose any direction to move from the starting point.

Hint2: Consider the scenario where the cost is 0 when the player moves with the wind and 1 otherwise. What can the original problem be transformed into?

2 Edge Changing (50% of this assignment)

2.1 Description

Give you a graph with n vertices and m edges. No two edges connect the same two vertices. For vertex ID from 1 to n, we do the following operation: If any two neighbors of a vertex have a $k\times$ relationship in terms of their IDs, we add a new edge between them. In other words, for any vertex $i = 1$ to n, if $u = kv$ or $v = ku$, we add an edge (u, v), where $u, v \in Neighbor(i)$. Besides, if there is already an edge between u and v, no operation is taken.

After the operation, we want you to output the BFS order starting from vertex s. Please traverse all neighbors in ascending order of their IDs when expanding a vertex.

2.2 Input

The first line of input contains four integers n, m, k and s.

m lines follow, each containing two integers a, b, indicating that there is an edge between a and b.

There is no self-loop and repeated edges.

2.3 Output

Output the smallest BFS order of the new graph.

Sample Input 1

```
5 5 2 3
1 2
1 3
2 3
3 4
4 5
```

Sample Output 1

```
3 1 2 4 5
```

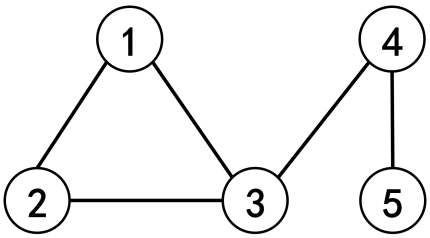


Figure 5: Original Graph

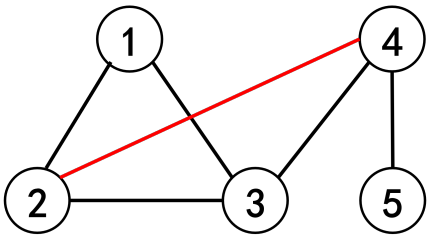


Figure 6: Modified Graph

As shown in Fig. 6, due to the $2\times$ relationship between neighbors 2 and 4 of vertex 3, We add a new edge $(2, 4)$ to the original graph.

You can find More Sample in the attached file on BB.

Problem Scale & Subtasks

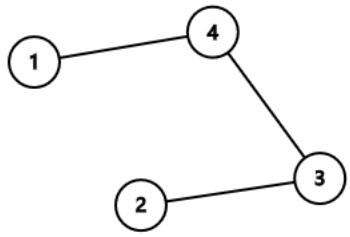
For 100% of the test cases, $1 \leq n, m \leq 10^5$, $2 \leq k \leq 10^5$.

Test Case No.	Constraints
1-4	$n, m \leq 100$
5-7	$n, m \leq 1000$
8-10	$n, m \leq 100000$

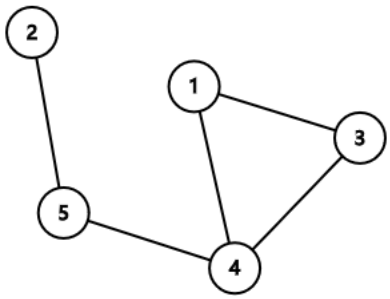
Hint

Hint1: For C/C++ and Java users, an `int` type stores integers range from -2,147,483,648 to 2,147,483,647. It may be too small for this problem. You need other data types, such as `long long` for C/C++ and `long` for Java. They store integers ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Use `scanf("%lld",&n)` for C, `cin>>n` for C++ and `n = scanner.nextLong()` for Java to get the input n. And the other operations for `long` and `long long` are quite same as `int`.

Hint2: Note that we deal with the vertices in increasing order. For example, in the following graph, when $k = 2$, we will add an edge between 2 and 4 when dealing with 3. Then 1 and 2 become the neighbors of 4, and we will add an edge between 1 and 2.



However, in the following graph, we add an edge between 2 and 4 when dealing with 5. Then 1 and 2 become the neighbors of 4. But we have already visited vertex 4 (before visiting 5), so we will not add an edge between 1 and 2.



3 Obstacle (optional)

This question is optional and will not be included in the grade.

3.1 Description

In the piggy kingdom, there are n cities and m roads connecting them. A road connects two different cities. For some reason, there are obstacles on some of the roads. The city 1 and city n are two important cities of the kingdom, and many people travel between these two cities. However, because of the obstacles, it may need more time to go from 1 to n. Now, the pig king wants to remove the obstacles. Since it takes a lot of time and money to remove the obstacles, the king decides to **only remove two of them**. Suppose the distance between city 1 and city n decreased by D after removing the obstacles. The king wants to know the maximum value of D. Your task is to find this value.

3.2 Input

The first line of input contains two integers n and m.

m lines follow, each containing four integers a, b, c, d, indicating that there is a road between a and b with length c. If d is 0, there is no obstacle on the road. If d is 1, there is an obstacle on the road. There might be more than one road between two cities.

It is guaranteed that one can travel from city 1 to city n via the roads without obstacles.

3.3 Output

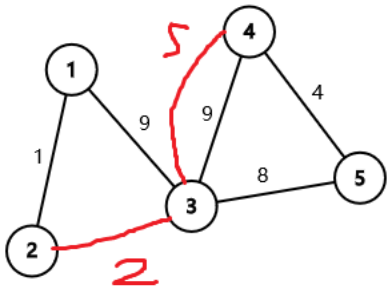
Output a single integer, representing the maximum value of D.

Sample Input 1

```
5 7
1 2 1 0
2 3 2 1
1 3 9 0
5 3 8 0
4 3 5 1
4 3 9 0
4 5 4 0
```

Sample Output 1

```
6
```



The original distance is 17, and the distance after removing the two obstacles is 11.

You can find More Sample in the attached file on BB.

Problem Scale & Subtasks

For 100% of the test cases, $1 \leq n, m, c \leq 10^5$.

Test Case No.	Constraints
1-2	$n, m \leq 10$
3-5	$n, m \leq 100$
6-7	$n, m \leq 1000$
8-10	$n, m \leq 100000$

Hint

Hint1: For C/C++ and Java users, an `int` type stores integers range from -2,147,483,648 to 2,147,483,647. It may be too small for this problem. You need other data types, such as `long long` for C/C++ and `long` for Java. They store integers ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Use `scanf("%lld",&n)` for C, `cin>>n` for C++ and `n = scanner.nextLong()` for Java to get the input n. And the other operations for `long` and `long long` are quite same as `int`.

Hint2: Do not get stuck on the original graph. Try to build a new graph to solve the problem.