# Overview

| **Author(s)** | **Max Rabkin** | **Keegan Carruthers-Smith** | **Julian Kenwood** | **Julian Kenwood** |
|---|---|---|---|---|
| **Problem** | **copyrite** | **bus** | **primcomp** | **minitris** |
| Source | copyrite.c<br>copyrite.cpp | bus.c<br>bus.cpp | primcomp.c<br>primcomp.cpp | minitris.c<br>minitris.cpp |
| Input file | copyrite.in | stdin | stdin | stdin |
| Output file | copyrite.out | stdout | stdout | stdout |
| Time limit | 1 second | 1 second | 1 second | 1 second |
| Memory limit | 64MiB | 64MiB | 64MiB | 64MiB |
| Number of tests | 10 | 10 | 10 | 10 |
| Points per test | 5 | 10 | 10 | 10 |
| Detailed feedback | No | Yes | No | No |
| **Total points** | **100** | **100** | **100** | **100** |

The maximum total score is 400 points.

# Detecting Plagiarism

Max Rabkin

## Introduction

You all know the legend of Bruce, the hero who had legendary abilities in both computer science and mathematics — according to some, from as young as six months. A recent discovery has cast doubt on the legend, however: it seems that some of Bruce's discoveries were in fact the work of a little known contemporary, Carl Hultquist. Hultquist suspected this from early on, but was dismissed as paranoid and delusional during his life.

In an attempt to prove Bruce's guilt, the young Hultquist placed secret "watermarks" in his programs and writings (he changed the text of the watermark over time). Unfortunately, the processing power needed to search through Bruce's enormous collection of work was not available at the time.

With modern processors and clever algorithms, it may be possible to return some dignity to Hultquist's memory by proving him correct.

## Task

You will be given a text, and Carl's watermark for the period in which it was written. You need to determine whether the text contains the watermark, and if it does, at what position.

Carl's watermarking system was case-insensitive and treated all punctuation the same, so we have preprocessed the input so that all the letters (in the text and watermark) are lowercase and all punctuation marks are replaced with an underscore (_).

## Example

Suppose the text was "MVEMJSUNP is probably not a real word". Carl's watermark at the time was "NP is P", As you can see, the watermark starts at position 7 in the text: `mvemsunp_is_probably_not_a_real_word`

## Input (copyrite.in)

The first line of input file contains Carl's watermark. The next line contains the text to be searched. Both the watermark and the text are made up of lowercase letters and underscores.

### Sample input

```
np_is_p
mvemjsunp_is_probably_not_a_real_word
```

## Output (copyrite.out)

The output consists of a single line, containing a single integer: `-1` is the watermark does not appear in the text; otherwise, its position (the first position is `1`, not `0`).

### Sample output

```
7
```

## Constraints

- $1 \leq$ length of watermark $\leq 100$
- $1 \leq$ length of text $\leq 1\,000$

## Time limit

1 second.

## Scoring

# Bus Routes

Keegan Carruthers-Smith

## Introduction

The government wants to create more bus routes in South Africa. Every bus route has buses travelling in both directions on it. Unfortunately, it is not always possible to get from a city to every other city using only buses. So the government wants to create more bus routes, such that it is possible to travel between any two cities just using buses.

## Task

Creating a bus route costs money, so your task is to work out the minimum cost needed such that there is a path between any two cities using only buses.

## Example

In the example there already exists bus routes between the cities $4 \leftrightarrow 2$ and between cities $3 \leftrightarrow 5$. If we build the bus routes $2 \leftrightarrow 3$ and $1 \leftrightarrow 3$, we can reach any city using the layout. This has a cost of 150, which is not minimal. However if we build the bus routes $2 \leftrightarrow 1$ (of cost 2) and $3 \leftrightarrow 1$, we can reach any city. This has a cost of 52, which is minimal.

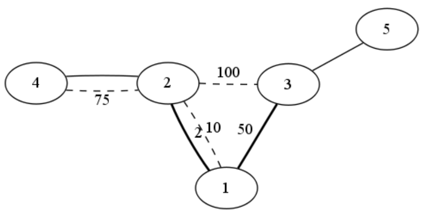


Figure 1: This represents the sample input. Non-bold solid lines represent bus routes that already exist. Bold solid lines represent the bus routes picked to minimize the cost of the task. Dashed lines represent other bus routes which were not picked.

## Input (stdin)

The first line of the input contains 3 space-separated integers, $N, C$ and $M$. There are $N$ cities, numbered 1 to $N$. The next $C$ lines each contain two space-separated integers $a$ and $b$. This represents an already existing bidirectional bus route between $a$ and $b$. The next $M$ lines each contain 3 space-separated $a, b$ and $c$. This represents a bus route you can build between $a$ and $b$ with cost $c$. No two cities are directly connected by both a potential bus route and an existing bus route. No two cities are directly connected by more than one potential or existing bus route.

### Sample input

```
5 2 5
4 2
3 5
2 3 100
3 1 50
1 2 10
4 2 75
2 1 2
```

## Output (stdout)

### Sample output

```
52
```

## Constraints

- $1 \le N \le 1\,000$
- $0 \le C, M \le 10\,000$
- $1 \le a, b \le N$
- $1 \le c \le 1\,000$
- There is always a way to build bus routes such that there is a path between any two cities.

## Time limit

1 second.

## Detailed feedback

Detailed feedback is enabled for this problem.

## Scoring

100% per test case for a correct answer. 0% otherwise.

# Primitive Compression

Julian Kenwood

## Introduction

Carl has been working with large datasets for his job. The datasets are so large, in fact, that he has had to develop a new kind of compression. The hard drives that Carl uses are very efficient at storing certain strings of data. Help Carl get the most compression possible.

## Task

The dataset is represented as a string of $N$ lowercase characters. You will also be given $K$ patterns each of length $|K_i|$. These patterns represent the strings that may be efficiently stored on the hard drive. Each pattern has an associated cost which is the amount of space required to store it on the hard disk.

Your task is to find the minimum space required to store Carl's dataset. The hard drives that Carl uses are experimental. Any stored data must be properly broken down into the patterns given. Luckily, there is always at least one way to break up each dataset into these patterns.

## Example

Suppose Carl would like to store the string 'aaabbc' on a hard drive and that there are six patterns that the hard drive accepts. 'a', 'b' and 'c' are allowed with cost 10 each. 'aa' can be stored with cost 15, 'ab' can be stored with cost 12 and finally 'abbc' can be stored with cost 35.

Carl can break up his dataset a number of different ways. Using only the single letter patterns yields a cost of 60. We can also use patterns 'aa' and 'abbc' for a cost of 50. We can do even better by using 'aa', 'ab', 'b' and then 'c' to give us a total of 45. This is the best we can do.

## Input (stdin)

The first line of the input contains two space-separated integers, $N$ and $K$. The second line contains a single lowercase word: a string representing Carl's dataset. The following $K$ lines each contain a lowercase string and an integer. These are the patterns and their corresponding cost to store them on the hard drive.

### Sample input

```
6 6
aaabbc
a 10
b 10
c 10
aa 15
ab 10
abbc 35
```

## Output (stdout)

Your output should consist of a single integer: the minimum cost required to store the dataset on the hard drive.

### Sample output

```
45
```

## Constraints

- $1 \leq N \leq 10,000$
- $1 \leq K \leq 100,000$
- $1 \leq |K_i| \leq 10$

Additionally, in 60% of the test cases:

- $1 \leq K \leq 25$

Additionally, in 30% of the test cases:

- $1 \leq N \leq 10$
- $1 \leq K \leq 5$

## Time limit

1 second.

## Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

# Minimal Triangles

Julian Kenwood

## Introduction

Bruce is doing research into various new technologies. While looking at other programs he has found some interesting open source software. This software relates to the field of computer vision, the field that allows computers to make sense of images. There is just one major problem: the software is **SLOW!** Instead of solving the problem himself, Bruce would like you to help.

## Task

Bruce has tracked the source of the software's slowness down to a single routine. This code takes in a list of $N$ points in the 2D plane and finds the triangle with smallest perimeter. The vertices of this triangle must be chosen from the list of points that were given. Degenerate triangles, triangles with zero area, are permitted. This part of the code is used extensively so it is important that it is made to run as fast possible.

Bruce would like you to write a program demonstrating the algorithm that you will use to solve the problem. Your program will be tested against data prepared by Bruce to determine if your code is fast enough to replace the existing code in the project.

Warning: The test data in this problem ranges from easy to very hard. If you get stuck on this problem it is advisable to skip it until you are satisfied with your other solutions.

## Example

Suppose you have a list of five points you want to run through the routine. Theseare: (1, 2), (3, 5), (10, 2), (0, 3) and (10, 10). There are ten triangles that you could make from the points.

We could make a triangle from (1, 2), (3, 5) and (10, 2). This gives us a triangle of perimeter 20.221 units. However if we choose (1, 2), (3, 5) and (0, 3) we get a much smaller answer of 8.625. This is clearly the smallest triangle by perimeter and is the answer.

## Input (stdin)

The first line of the input contains a single integer: $N$, the number of points. The following $N$ lines each contain two space-separated integers, the x and y coordinates of each point. All coordinates will fit into a signed 32-bit integer.

### Sample input

```
5
1 2
3 5
10 2
0 3
10 10
```

## Output (stdout)

The output consists of a single decimal number, the perimeter of the smallest triangle rounded off to three decimal places.

### Sample output

```
8.625
```

## Constraints

- $1 \leq N \leq 100,000$

Additionally, in 30% of the test cases:

- $1 \leq N \leq 100$

## Time limit

1 second.

## Scoring

A correct solution will score 100% while an incorrect solution will score 0%.