4. Design of algorithms You want to build a wall towards Mexico in the Southern United States. You have a map of the area, which is partitioned into squares. Each square can hold a piece of wall; the task is to connect the eastern and western shores with wall. Two squares are adjacent if they share a side to the north, south, east, or west. (In other words, the wall does not connect diagonally.) Due to various reasons (rivers, water, swamp, mountains, need to relocate a town, etc.), the sqares are more or less expensive to build on.

In the image below, you can see squares overlaid on a map of Mexico. The integers give the cost. The letters mean `A` for Atlantic, `P` for Pacific, `U` for the inner United States, and `M for Mexico`. No wall can be built on either of those four types of square. The task is to connect any of the `P` to any of the `A` using adjacent pieces of wall for as little total cost as possible.

Input

A space separated sequence of squares, line by line, in the obvious fashion. A full stop `.` is used for the empty squares. Note that the examples below are small, in general you cannot assume that the costs are single digits.

To fix notation, let r denote the number of rows in the input (i.e., the number of lines), and l the number of columns (i.e., the length of the longest line).

Output

The cost of the cheapest wall you can build.

Don't take 'wall' too literally. For instance, the area south of Florida goes across water, so this is probably a job for the US coast guard instead, rather than a bricklayer. Think of the wall pieces as 'making a route impossible to use.'

Also, don't let your knowledge of correct geography get in the of solving the exercise. An optimal solution to the example above loses San Diego and a good part of Florida, which would be undesirable in reality, but is not important for this exercise.

Example

Input:
```
. U U U U U U
P 7 9 8 8 7 5 U . . . U U U
P 2 2 2 1 1 6 6 U U U 5 5 U A
P 1 2 3 2 2 2 2 4 5 5 4 2 5 A
. M M M 3 3 3 2 6 5 4 2 2 2 A
. . . . M M 2 2 2 2 3 7 2 2 A
. . . . . . . M 2 3 2 7 7 7 7 A
. . . . . . . . M 7 7 7 7 7 7 A
. . . . . . . . M 7 7 7 M M 7 A
. . . . . . . . . M 7 M
. . . . . . . . . . M
```

Output:
```
35
```

Input:
```
. U U U
P 1 1 3 A
P 3 1 1 A
. M M M
```

Output:
```
4
```

(a) (*3 pt.*) Describe an algorithm that solves the problem.

(b) (*2 pt.*) Because of shady deals with Russian entrepreneurs you found a cheaper way of constructing wall sections: Now each square costs the same. (Equivalently, you can pretend that all costs in the input are 1.) Describe an algorithm that solves this problem faster than your answer to 4a.

For both questions, explain your algorithm briefly and clearly, and state the running time of your algorithm in terms of the given parameters. You are strongly encouraged to make use of existing algorithms, models, or data structures from the book, but please be precise in your references (for example, use page numbers or full class names of constructions in the book). Each question can be perfectly answered on half a page of text. (Even less, in fact.) If you find yourself writing much more than one page, you're using the wrong level of detail.

Important: If you decide to view this problem as a graph problem, you *must* describe how the graph is constructed, and you *must* draw the *entire* graph corresponding to the smaller of the two examples.

You don't need to write code. (However, some people have an easier time expressing themselves clearly by writing code. In that case, go ahead.) You are evaluated on correctness and efficiency of your solutions, and clarity of explanation.