Homework 2 Part II: Programming Challenges

News:
Sept 30, 2022: Release

Code submission mechanism: Include your PSID and Submission ID from the on-line judge system in the written report submitted to blackboard. Your code must only read from stdin and write to stdout. **Save your code in your computer.**

We will be compiling and running your submitted code. We may use test cases that are not given to you.

For your convenience, the starter codes can be found in the sandbox environment: sandbox . You need a Github account to access; it's free of charge. We will be using the same environment and compiler flags to compile your code

Problem 1 (10pt): Backtracking

Online judge/submission link: OnlineJudge 1

Given a board with irregular shape, your task is to place chess pieces and make sure no more than 1 chess pieces is placed on the same row or the same column. Output the number of different ways of such placement.

The first line of the input consists of two integers n and k. n means the size of board is n×n and k is the number of chess pieces to be placed. The next n lines describe the shape of the chess board: '#' describes the board region and '.' is the blank region (cannot place chess piece here).

Starter code: pro1.cpp

Example 1

```
Input:
2 1
.#
#.

Output:
2
```

Example 2

```
Input:
3 3
#.#
.##
.##
Output:
2
```

Example 3

```
Input:
4 4
...#
..#.
.#..
#...
Output:
1
```

Problem 2 (10pt): Divide and Conquer

Online judge/submission link: OnlineJudge 2

Given an array A, write a program to find the max(A[j]-A[i]) where i < j. If max(A[j]-A[i])<0, output 0. The input will start with an integer n, which indicates the length of the given array. The next line will be the array.

Starter code: pro2.cpp

Example 1

```
Input:
2
1 5
Output:
4
```

Example 2

```
Input:
4
1 5 8 2
Output:
7
```

Example 3

```
Input:
6
8 7 4 3 2 1
Output:
0
```

Problem 3 (10pt): Dynamic Programming

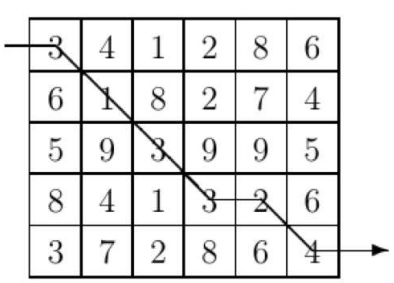
Online judge/submission link: OnlineJudge 4

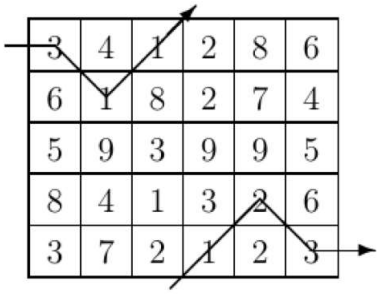
Given a `m * n` matrix, your task is to compute a path from any element on the 1 st column to any element on the last column at minimal cost. The path consists of several steps, each step is moving from column `j` to column `j+1` in an adjacent (horizontal or diagonal) row. The first row and the last row are considered as adjacent rows. The cost of a path is the sum of visited integers.

The two slightly different matrices are shown in figure below. The minimum cost path is illustrated in the figure and the 2 nd path takes advantage of the adjacency property of the first and last row.

You'll be given two integers `m` and `n` that indicate the number of rows and columns at the first line. The next `m` lines will be the matrix and each line represents one row of the given matrix. You're required to output the minimum cost.

Notes: : The test cases have the properties: number of rows is between 1 and 10; number of columns is between 1 and 100; number in matrix can be positive or negative; All path weights can be represented by 30-bit signed integer.





Example 1

```
Input:
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 8 6 4
Output:
16
Explanation:
the path: 1 2 3 4 4 5
```

Example 2

```
Input:
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 1 2 3
Output:
11
Explanation:
the path: 1 2 1 5 4 5
```