ShanghaiTech University

CS240 Algorithm Design and Analysis

Spring 2020

Course Project

Due: 23:59, July 3, 2020

1. This project requires you to solve four problems. For each problem, write a program with the input and output formats as specified in the problem.
2. Each problem has a time and memory limit, and programs exceeding either limit will not be accepted. You can write your programs in C, C++, Java or Python, though certain languages may lead to more efficient implementations.
3. You may not use third-party libraries in your solution, *e.g.* numpy in Python is not allowed. If you are not sure whether a library is third-party or not, you can try to submit your program, and it will cause a Runtime Error or Compile Error if it is.
4. Your program submission is scored by an Online Judge platform containing a number of test cases. Your score depends on the percentage of test cases your program passes.
5. The Online Judge website is `http://10.19.125.64`. You can also find the problem list at `http://10.19.125.64/contest/1/problems`.
6. Each student's account name is **Your student ID** and the initial password is also **Your student ID**. After logging into your account for the first time, please change your password.
7. To avoid abuse of the OJ, you are allowed at most one submission every five minutes. Violations of this rule may result in penalties to your score.
8. You must **NOT**
   (a) Read or use solutions written by others.
   (b) Allow other people to read or use your solutions.
   (c) Obtain test data by repeated submissions or other means.

Problem 1: Pay up

A group of companies made a number of transactions among themselves, and now it's come time to settle the bills. The companies want to pay each other so they all get the money they're owed at the end. However, when several companies owe each other money, it may be possible to reduce the number of transactions. For example, if A owes \$100 to B, B owes \$40 to C and C owes \$60 to A, then it's possible to settle all the bills by having A and C pay B \$40 and \$20, resp., thereby performing only two transactions instead of three.

Given the initial transactions between the companies, determine the minimum number of transactions needed by the companies to settle all their bills.

Hint: A "brute force" solution may be able to pass the Online Judge, as long as it's not "too" brute force.

Input:
The input consists of

1. A line containing two integer n, the number of companies, with $1 \leq n \leq 20$, and m, the number of transactions, with $0 \leq m \leq 1000$.
2. m lines, each with three integers i, j, t, where $0 \leq i, j, < n$, and $1 \leq t \leq 1000$, representing that company j owes company i an amount t.

Output:
A single integer, equal to the number of transactions needed to settle all the bills.

Time and memory limit:
1000ms, 256MB

Sample Input 1:
4 2
0 1 1
2 3 1

Sample Output 1:
2

Sample Input 2:
5 5
0 1 3
1 2 3
2 3 3
3 4 3
4 0 3

Sample Output 2:
0

Problem 2: 1D 2048

Have you ever played the game 2048? The game is played on a 2D board with numbered tiles. When two tiles with the same number touch, they merge into one. We will play 2048 in 1 dimension. The rules are as follows.

1. We start with n tiles (numbers) on the line.
2. We can merge two tiles if they are adjacent and equal, producing a single tile whose value is one higher. That is, if we have tiles $a_i = a_{i+1}$, they can be merged into a single tile with value $a_i + 1$.

The goal is to find an order to merge the tiles in order to have the minimum number of tiles at the end.

For example, if we start with tiles 3, 3, 4, 4, 4, 3, 3, we can merge the two pairs of 3 tiles, producing the sequence 4, 4, 4, 4, 4. Then we can merge two pairs of 4 tiles, to produce 5, 5, 4, and finally merge the 5 tiles to produce two tiles 6, 4, which is the minimum possible number of remaining tiles.

Input:
The first line contains an integer $n (1 \leq n \leq 500)$ equal to the number of tiles at the beginning.
The second line contains n integers $a_1, a_2, \ldots, a_n, (1 \leq a_i \leq 1000)$ representing the initial values of the tiles.

Output:
An integer – the minimum possible number of tiles at the end of the game.

Time and memory limit:
2000ms, 256MB

Sample Input:
7
3 3 4 4 4 3 3

Sample Output:
2

Problem 3: Knights

Suppose we are given an $n \times n$ chess board, along with $m \leq n^2$ squares of the board marked as obstacles. Our goal is to place as many knight pieces on the board as possible, such that no pair of knights attack each other, and no knight is placed on an obstacle. Given a knight placed at square (x, y), the knight attacks any other knights placed at squares $(x+1, y+2), (x+1, y-2), (x-1, y+2), (x-1, y-2), (x+2, y+1), (x+2, y-1), (x-2, y+1)$, or $(x-2, y-1)$. Compute the maximum number of knights that can be placed.

Hint: Recall that squares on the chess board are colored black and white. Use the fact that a knight placed on a square only attacks squares of the opposite color.

Input:
A line containing two integer n, m
Next m lines, each with two integers x, y, representing the coordinates of an obstacle.

Output:
An integer – the maximum number of knights that can be placed on the board so that they do not attack each other.

Time and memory limit:
1000ms, 256MB

Sample Input:
3 2
1 1
3 3

Sample Output:
5

Problem 4: Square coloring

Suppose you're given an $n \times n$ grid in which some squares are white and some are black. Your goal is to make all the squares white. To do this, you can select any rectangle of size $w \times h$ and make all these squares white. This operation has a cost of $\max(w, h)$. Compute the minimum cost to make all squares white.

Input:
The first line contains a single integer $n (1 \leq n \leq 50)$, equal to the size of the square grid.
Each of the next n lines contains a string of length n, consisting of characters '.' and '#'. The j'th character of the i'th line is '#' if the cell with coordinates (i, j) is black; otherwise it is white.

Output:
Print a single integer — the minimum total cost to make all cells white.

Time and memory limit:
1000ms, 256MB

Sample Input 1:
```
3
###
#.#
###
```

Sample Output 1:
3

Sample Input 2:
```
4
#...
....
....
#...
```

Sample Output 2:
2