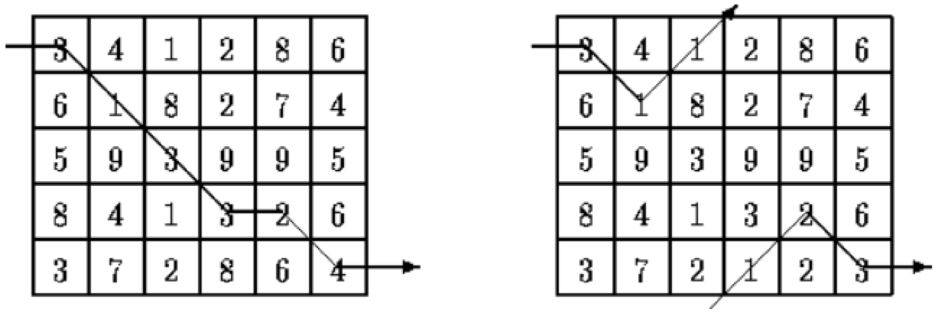
CS222 Homework 4

Dynamic Programming
Deadline: 2018-10-23 Tuesday 24:00

Exercises for Algorithm Design and Analysis by Li Jiang, 2018 Autumn Semester

Hint: There are some programming problems in this assignment. You'd better complete it using java or c++. And the source files are needed which should be named like "Q1/2/3 .c/.cpp/.java"

1. Given an $m * n$ matrix of integers, you are to write a program that computes a path of minimal weight. A path starts anywhere in column 1 (the first column) and consists of a sequence of steps terminating in column n (the last column). A step consists of traveling from column i to column $i+1$ in an adjacent (horizontal or diagonal) row. The first and last rows (rows 1 and m) of a matrix are considered adjacent, i.e., the matrix "wraps" so that it represents a horizontal cylinder. The weight of a path is the sum of the integers in each of the n cells of the matrix that are visited. For example, two slightly different $5 * 6$ matrices are shown below (the only difference is the numbers in the bottom row). The minimal path is illustrated for each matrix. Note that the path for the matrix on the right takes advantage of the adjacency property of the first and last rows. Please give an algorithm to solve this problem.



2. There is one very popular song called Jin Ge Jin Qu. It is a mix of 37 songs, and is extremely long (11 minutes and 18 seconds). Why is it popular? Suppose you have only 15 seconds left (until your time is up), then you should select another song as soon as possible, because the KTV will not crudely stop a song before it ends (people will get frustrated if it does so!). If you select a 2-minute song, you actually get 105 extra seconds! ....and if you select Jin Ge Jin Qu, you'll get 663 extra seconds!!!

   Now that you still have some time, but you'd like to make a plan now. You should stick to the following rules:

   (1) Don't sing a song more than once (including Jin Ge Jin Qu).

   (2) For each song of length t, either sing it for exactly t seconds, or don't sing it at all.

   (3) When a song is finished, always immediately start a new song.

   Your goal is simple: sing as many songs as possible, and leave KTV as late as possible (since we have rule 3, this also maximizes the total lengths of all songs we sing) when there are ties. Please give an algorithm to solve this problem.

3. **(Programming Problem)** We have received an order from Pizoor Communications Inc. for a special communication system. The system consists of several devices. For each device, we are free to choose from several manufacturers. Same devices from two manufacturers differ in their maximum bandwidths and prices.

   By overall bandwidth (B) we mean the minimum of the bandwidths of the chosen devices in the communication system and the total price (P) is the sum of the prices of all chosen devices. Our goal is to choose a manufacturer for each device to maximize B/P.

Input

The first line of the input file contains a single integer t ($1 <= t <= 10$), the number of test cases, followed by the input data for each test case. Each test case starts with a line containing a single integer n ($1 <= n <= 100$), the number of devices in the communication system, followed by n lines in the following format: the i-th line ($1 <= i <= n$) starts with mi ($1 <= mi <= 100$), the number of manufacturers for the i-th device, followed by mi pairs of positive integers in the same line, each indicating the bandwidth and the price of the device respectively, corresponding to a manufacturer.

Output

Your program should produce a single line for each test case containing a single number which is the maximum possible B/P for the test case. Round the numbers in the output to 3 digits after decimal point.

Sample Input

1 3

3 100 25 150 35 80 25

2 120 80 155 40

2 100 100 120 110

Sample Output

0.649

4. **(Programming Problem)**Given a two-dimensional array of positive and negative integers, a sub-rectangle is any contiguous sub-array of size $1 * 1$ or greater located within the whole array. The sum of a rectangle is the sum of all the elements in that rectangle. In this problem the sub-rectangle with the largest sum is referred to as the maximal sub-rectangle. As an example, the maximal sub-rectangle of the array:

   0 -2 -7 0

   9 2 -6 2

   -4 1 -4 1

   -1 8 0 -2

   is in the lower left corner:

   9 2

   -4 1

   -1 8

   and has a sum of 15.

   Input

   The input consists of an $N * N$ array of integers. The input begins with a single positive integer N on a line by itself, indicating the size of the square two-dimensional array. This is followed by N^2 integers separated by whitespace (spaces and newlines). These are the N^2 integers of the array, presented in row-major order. That is, all numbers in the first row, left to right, then all numbers in the second row, left to right, etc. N may be as large as 100. The numbers in the array will be in the range [-127,127].

   Output

   Output the sum of the maximal sub-rectangle.

   Sample Input

   4

   0 -2 -7 0

   9 2 -6 2

   -4 1 -4 1

   -1 8 0 -2

   Sample Output

   15

5. **(Programming Problem)**A Bank plans to install a machine for cash withdrawal. The machine is able to deliver appropriate @ bills for a requested cash amount. The machine uses exactly N distinct bill denominations, say Dk, k=1,N, and for each denomination Dk the machine has a supply of nk bills. For example,

   N=3, n1=10, D1=100, n2=4, D2=50, n3=5, D3=10

   means the machine has a supply of 10 bills of @100 each, 4 bills of @50 each, and 5 bills of @10 each.

   Call cash the requested amount of cash the machine should deliver and write a program that computes the maximum amount of cash less than or equal to cash that can be effectively delivered according to the available bill supply of the machine.

   Notes: @ is the symbol of the currency delivered by the machine. For instance, @ may stand for dollar, euro, pound etc.

   Input

   The program input is from standard input. Each data set in the input stands for a particular transaction and has the format:

   cash N n1 D1 n2 D2 ... nN DN

   where $0 <= cash <= 100000$ is the amount of cash requested, $0 <= N <= 10$ is the number of bill denominations and $0 <= nk <= 1000$ is the number of available bills for the Dk denomination, $1 <= Dk <= 1000$, $k = 1, N$. White spaces can occur freely between the numbers in the input. The input data are correct.

   Output

   For each set of data the program prints the result to the standard output on a separate line as shown in the examples below.

   Sample Input

   ```
   735 3 4 125 6 5 3 350
   633 4 500 30 6 100 1 5 0 1
   735 0
   0 3 10 100 10 50 10 10
   ```

   Sample Output

   ```
   735
   630
   0
   0
   ```

   Hint: The first data set designates a transaction where the amount of cash requested is @735. The machine contains 3 bill denominations: 4 bills of @125, 6 bills of @5, and 3 bills of @350. The machine can deliver the exact amount of requested cash.

   In the second case the bill supply of the machine does not fit the exact amount of cash requested. The maximum cash that can be delivered is @630. Notice that there can be several possibilities to combine the bills in the machine for matching the delivered cash.

   In the third case the machine is empty and no cash is delivered. In the fourth case the amount of cash requested is @0 and, therefore, the machine delivers no cash.