Digital Pakistan Speed Programing Competition Online Qualifier Round

Instructions

- Do not open the booklet unless you are explicitly told to do so. You can only read these instructions below.
- If you have any question regarding the problems, seek a clarification from the judges using DOMJudge.
- Before submitting a run, make sure that it is executable via command line. For Java, it must be executable via "javac" and for GNU C++ via "g++". Java programmers need to remove any "package" statements and source code's file name must be the same as of main class. C++ programmers need to remove any getch() / system("pause") like statements.
- Do not attach input files while submitting a run, only submit/attach source code files, i.e., *.java or *.cpp or *.py.
- Language supported: C/C++, Java and Python3
- Source code file name should not contain white space or special characters.
- You must take input from Console i.e.: Standard Input Stream (stdin in C, cin in C++, System.in in Java, stdin in Python)
- You must print your output to Console i.e.: Standard Output Stream (stdout in C, cout in C++, System.out in Java)
- Please, don't create/open any file for input or output.
- Please strictly meet the output format requirements as described in problem statements, because your program will be auto judged by computer. Your output will be compared with judge's output byte-by-byte and not tolerate even a difference of single byte. So, be aware! **Pay special attention to spaces, commas, dots, newlines, decimal places, case sensitivity etc.**
- All your programs must meet the time constraint specified.
- The decision of judges will be absolutely final.

Problem 05: The Last Light of Makran: Adversarial Restoration

Time limit: 1 second

Gwadar is the last city in Makran to regain electricity after a massive blackout. Power is transmitted from a remote source through a grid of substations and transmission lines. You are tasked with ensuring that even in the worst case of sabotage, at least D megawatts (MW) of electricity can still reach Gwadar. The power grid has N substations numbered 0 to N - 1. Power originates at substation Turbat (Node 0) and is consumed at substation N - 1.

The grid has M transmission lines. Each line connects two substations u and v and can carry up to c MW. Additionally, a line can be upgraded at a cost of p units to increase its capacity by x MW. You have a budget of B units and may upgrade any subset of the M lines within this budget. However, a Saboteur lurks in the hills and will destroy at most K transmission lines after you commit your upgrades.

Your goal is to decide whether it is possible to upgrade some lines (within budget B) such that, regardless of which K lines are destroyed, the minimum possible amount of power that can still be transmitted from substation 0 to N - 1 is at least D MW.

Input:

The first line contains a single integer T ($1 \le T \le 50$) — the number of test cases.

Each test case consists of:

- One line with three integers N, M, and K ($2 \le N \le 100$, $0 \le M \le 500$, $0 \le K \le M$) , the number of substations, transmission lines, and sabotage limit.
- One line with two integers D and B ($0 \le D \le 10^9$, $0 \le B \le 10^9$), the required minimum power after sabotage, and the total budget.
- M subsequent lines, each describing a transmission line with five integers:
 u v c x p ($0 \le u, v < N$, $1 \le c, x, p \le 10^9$) , a line between substations u and v, with base capacity c, upgrade increase x, and upgrade cost p.

Output:

For each test case it should print one line:

1 <flow> if the minimum flow (after the Saboteur's optimal destruction) is at least D, or

0 <flow> otherwise, where <flow> is the minimum flow that can be guaranteed after sabotage.

Notes:

- You may choose not to upgrade any lines (i.e., L = 0).
- The Saboteur destroys any K lines of their choice after you upgrade.
- Your selection must guarantee that the minimum possible flow from substation 0 to substation N - 1, even after worst-case sabotage, is at least D.

Sample Input	Sample Output

2
5 3 1
3 3
0 1 2 2 2
1 2 2 1 1
0 2 1 1 1
7 2 2
4 5
0 1 3 2 2
1 2 2 2 2
2 3 2 2 2
0 2 1 2 1
1 3 1 1 1
	1 5
	0 2