WOBURN CHALLENGE

2015-16 Online Round 1

Friday, October 16th, 2015

Senior Division Problems

Problem S1: Woburn Workload

15 Points / Time Limit: 2.00s / Memory Limit: 16M *Submit online: wcipeg.com/problem/wc151s1*

Welcome to the first Woburn Challenge in over a decade! We hope you find these contests fun and challenging.

Let's start by getting to know Woburn a little more. The motto of Woburn Collegiate Institute is *Studium Eruditionis Crescat*, or *Let the Zeal for Learning Flourish*. Clearly, Woburn values academic excellency, so it's no surprise that students often receive a hefty workload. There are also many extracurricular opportunities available to students, including Woburn Music, drama, and of course, the Programming Enrichment Group. Members of PEG meet Wednesdays and Fridays after school to study advanced computer science concepts, and are often assigned a fair amount of programming homework. Not only has PEG made students superb at programming, they've made students *love* programming. In fact, PEG students are often so hooked onto their programming homework that they forget about actual homework and become swamped with upcoming deadlines!

Being the natural problem-solvers they are, PEG students have realized that programming is once again the solution. They've decided to write a program to help them handle the workload of Woburn's courses, and they've asked you for help! Specifically, there are N ($1 \le N \le 100$) homework assignments that they would like to feed to your program. Each assignment is labeled with a unique integer from 1 to N, and will be given to your program in order. Each assignment i is due M_i ($1 \le M_i \le 10^6$) minutes from now. Any given assignment can only be worked on up until the time at which it's due. The assignments will be given to your program in strictly increasing order of due date (in other words, $M_1 < M_2 < \ldots < M_N$).

Furthermore, each assignment i takes T_i ($1 \le T_i \le 10^6$) minutes to fully complete. The amount of marks you get on an assignment is obviously proportional to the amount of time you spend working on it. Since spending all T_i minutes is guaranteed to be your best work, if you spend S_i minutes working on assignment i, where $S_i \le T_i$ (spending extra time on an assignment does no good), you'll have to incur $T_i - S_i$ penalty marks.You can only work on one homework assignment at a time, but you can switch from one assignment to another at any point in time. Time is tight, so you might not be able to fully complete every assignment. The goal of the program is to help the user allocate their precious time amongst the assignments. Your program must answer the question: what's the minimum total number of penalty marks that can be incurred across all of the assignments?

Input Format

The first line of input will contain a single integer N, representing the number of assignments to follow. N lines will follow, with the i-th of these lines containing two space-separated integers M_i and T_i, respectively representing the number of minutes from now at which assignment i is due and the number of minutes it takes to complete assignment i.

Output Format

Output a single integer representing the minimum total penalty marks that can be incurred across all of the assignments, if your program schedules the assignments optimally.

Sample Input

```
4
40 40
80 60
120 30
130 80
```

Sample Output

```
80
```

Explanation

There are four assignments, respectively due 40, 80, 120, and 130 minutes from now and respectively requiring 40, 60, 30, and 80 minutes to complete. One way to optimally tackle the assignments is as follows:

- Immediate start working on assignment 1. You'll be done just in the nick of time and incur no penalty marks.
- Right after you hand in assignment 1 (at 40 minutes in), start working on assignment 2. Unfortunately, you'll only have 40 minutes to work on the assignment, and so you'll have to incur 60 − 40 = 20 penalty marks.
- Right after you hand in assignment 2 (at 80 minutes in), start working on assignment 4. You'll be able to get 50 minutes of work on it before the due date (at 130 minutes in). You'll have to incur 80 − 50 = 30 penalty marks.
- Unfortunately, there was no time at all to tackle assignment 3, so you'll have to accept a zero and incur the full 30 penalty marks.

In total, there are 20 + 30 + 30 = 80 penalty marks incurred. There may be other ways to tackle the assignments, but it turns out that they will all incur 80 penalty marks or more.

Problem S2: Wildcat Wildcards

20 Points / Time Limit: 2.00s / Memory Limit: 16M *Submit online: wcipeg.com/problem/wc151s2*

The mascot of Woburn is the Wildcat. The Woburn Wildcats are soon going to be playing in the finals of the Scarborough Halo 2 Association (SHA-2) and the rest of the school would like to make signs to cheer them on. The Student Activity Council (SAC) has purchased heaps of giant cards for the sole purpose of creating signs that spell out the word "`WILDCATS`". Specifically, we can construct one such sign by combining 8 cards with the letters `W`, `I`, `L`, `D`, `C`, `A`, `T`, and `S` on them, respectively.

There are two types of cards – letter cards and wildcards. Printed on each letter card is what you would expect… a single letter that is part of the word "`WILDCATS`". On the other hand, there are three types of wildcards – the `@` card, the `#` card, and the `*` card. A wildcard marked with a `@` may be filled in with any vowel of your choice (in this case, either an `A` or an `I`). Similar, each wildcard marked with a `#` may be filled in with any consonant (in this case, one of `W`, `L`, `D`, `C`, `T`, or `S`). Finally, each wildcard marked with a `*` may be filled in with any letter at all!

After counting inventory, the SAC was able to come up with 11 integers, the number of cards they have marked with `W`, `I`, `L`, `D`, `C`, `A`, `T`, `S`, `@`, `#`, and `*`, in that order. There are at most 10^8 (100,000,000) of each type of card. The Woburn SAC would like you to write a program to help them maximize the number of "`WILDCATS`" signs that can be made, given that you choose to fill in the wildcards optimally.

In test cases worth 50% of the points, there will be at most 1000 of each type of card.
In a subset of those test cases worth 10% of the points, there will be no wildcards (the last 3 integers of the input will each be 0).

Input Format

The input will consist of 11 lines, each containing a single nonnegative integer less than or equal to 10^8. The first 8 integers will represent the number of cards for each letter of "`WILDCATS`", in the order they occur in the word.
The last 3 integers will represent the number of `@`, `#`, and `*` wildcards, in that order.

Output Format

Output a single integer – the maximum number of "`WILDCATS`" signs that can be constructed if the wildcards are filled in optimally.

Sample Input

```
6
7
10
10
10
10
5
10
10
3
2
5
```

Sample Output

```
9
```

Explanation

There are 6 `W`'s, 7 `I`'s, 5 `A`'s, and 10 of each remaining letter card. We fill in 1/3 of our `@` wildcards as `I` and the remaining 2/3 as `A`. We fill both of our `#` wildcards as `W`. Finally, we fill in one `*` wildcard as `W`, one as `I`, and two as `A`. This will be enough to create 9 signs with 6 cards to spare (namely, one each of `L`, `D`, `C`, `T`, `S`, and a `*` wildcard).

Problem S3: Contest Sites

30 Points / Time Limit: 2.00s / Memory Limit: 16M *Submit online: wcipeg.com/problem/wc151s3*

It's no surprise that the Programming Enrichment Group at Woburn participates in a wide number of programming contests. However, not all contests are held at the same place (worry not – the Woburn Challenge finals will be held at a single location). In fact if too many people register for a contest, the contest organizers may even demand that the same contest be held in two entirely different towns! This will make it a lot harder for our young Woburnites to compete in the contest, but certainly won't stop them from trying.

There are N ($2 \le N \le 10^5$) towns uniquely numbered with integers from 1 to N, and M ($1 \le M \le 10^5$) one-way roads running amongst them. Specifically, the i-th road (for i = 1..M) allows one to travel from a town A_i to a different town B_i ($1 \le A_i$, $B_i \le N$; $A \ne B$) and has a distance of D_i ($1 \le D_i \le 100$) kilometres. No two roads run between the same pair of towns in the same direction. A very large programming contest is soon to be held across two locations, with the main contest site located in town 1 and the secondary contest site located in town 2. A non-zero number of Woburnites will be competing in this contest, with C_i ($0 \le C_i \le 10^6$) of them living in town i (for each i = 1..N). Each competitor must select one of the two contest sites and travel to it using a sequence of roads. This sequence is possibly empty, if they're fortunate enough to live in the same town as their chosen contest site.

There is a catch! Resources are limited at the second contest site, so the contest organizers have insisted that no more than K ($0 \le K \le 10^9$) of the competitors attend the secondary site (located in town 2). Given that the Woburnites collaborate to come up with the best travel strategy, you must help them determine the minimum total combined distance that they must travel in order to all attend the contest, such that no more than K of them travel to the secondary site.

In test cases worth 60% of the marks, $N \le 100$.
In a subset of those cases worth 30% of the marks, $C_i \le 1$ (for i = 1..N).

Input Format

Line 1 of input will contain three space-separated integers, the values of N, M, and K, respectively representing the number of towns, roads, and the maximum number of Woburnites that may attend the second site.
There will be N lines to follow. The i-th of these lines (for i = 1..N) will contain a single integer C_i, representing the number of Woburnites living in town i.
There will be M lines to follow. The i-th of these lines (for i = 1..M) will contain three space-separated integers, the values of A_i, B_i, and D_i representing the i-th one-way road.

Output Format

The output should consist of a single integer. If it is possible for the Woburnites to travel to all contest sites with no more than K of them attending the secondary site, then output the minimum total combined distance that they must travel to do so. Output `-1` if it is impossible for all of them to reach a contest site while satisfying the condition. Please note that the answer may not necessarily fit in a 32-bit integer.

Sample Input

```
4 5 5
2
1
5
7
1 2 1
3 2 1
3 4 1
4 1 1
4 3 1
```

Sample Output

```
13
```

Explanation

There are 4 towns and 5 roads between them. At most 5 people may attend the second contest site. Each road happens to have a distance of 1. An optimal travel strategy is as follows.

- We let competitors from towns 1 and 2 attend the contest sites at their respective towns (incurring an additional 0 kilometres to our total distance).
- We let all 7 of our competitors from town 4 travel to the main contest site (incurring a total of 7 kilometres).
- We let 4/5 of the competitors from town 3 travel to the secondary contest site (incurring a total of 4 kilometres).
- We let the remaining competitor from town 3 must travel to the main site through town 4 (incurring an additional 2 kilometres to our total distance).

The total distance traveled across all of the competitors is 0 + 7 + 4 + 2 = 13 kilometres.

Problem S4: Chocolate Milk

35 Points / Time Limit: 2.00s / Memory Limit: 16M *Submit online: wcipeg.com/problem/wc151s4*

After the finals of the 2015-16 Woburn Challenge, young programmers from all over the region will be invited to Woburn for a top-secret after-party (not really though)! The school has generously agreed to nourish them with copious amounts of chocolate milk.

To this end, the Student Activity Council (SAC) has acquired N ($2 \le N \le 200$) cisterns with infinite capacities, and positioned them at distinct heights off the ground. The cisterns are uniquely numbered from 1 to N in order of their heights from lowest to highest. For each cistern i (such that $2 \le i \le N$), there will be P_i ($0 \le P_i \le 10^7$) litres of chocolate milk pumped into it per second, directly from Scarborough's local chocolate milk farm. Additionally, there will be a pipe leading down from each cistern i to a different cistern C_i. According to the laws of gravity, chocolate milk can only flow from a higher cistern to a lower cistern, so it's guaranteed that $i > C_i$. The pipe coming out of i will be wide enough to allow at most F_i ($1 \le F_i \le 10^7$) litres of chocolate milk to flow through it per second. Chocolate milk flows through the system as one might expect – for every cistern, the amount of chocolate milk flowing out of it cannot exceed the total amount flowing into it (both directly from the farm and down from the other cisterns).

However, PEG has received enough funding this year to be able to upgrade K ($0 \le K < N$) of the $N - 1$ pipes. When a pipe is upgraded, it permanently becomes able to support an infinite amount of chocolate milk flowing through it.

Cistern 1 is located in Woburn's cafeteria, and the students will have permission to drink from it at will! Coding is a strenuous sport, so we know that everyone will be extremely thirsty. So the more chocolate milk flows into that cistern, the more they will collectively drink. And the more chocolate milk they consume, the more their coding skills will improve for next year's Challenge. Given the configuration of cisterns and pipes, and provided that we choose the optimal set of K pipes to upgrade, what's the maximum rate of incoming chocolate milk flow (in litres per second) that cistern 1 can attain?

In test cases worth 25% of the marks, $K \le 2$.
In a subset of those test cases worth 10% of the marks, $K = 0$.

Input Format

Line 1 of input will contain two space-separated integers N and K, respectively representing the number of cisterns and the number of upgrades allowed.
Line 2 to N of input will each contain information about a cistern. Specifically, line i of input (for i = 2..N) will contain three space-separated integers P_i, C_i, and F_i, respectively representing the rate at which chocolate milk is being pumped into cistern i (in litres per second), the cistern that the pipe from cistern i flows into, and the maximum rate at which chocolate milk (in litres per second) can flow through this pipe.

Output Format

Output a single integer, the maximum rate of incoming chocolate milk flow that cistern 1 can attain, in litres per second.

Sample Input

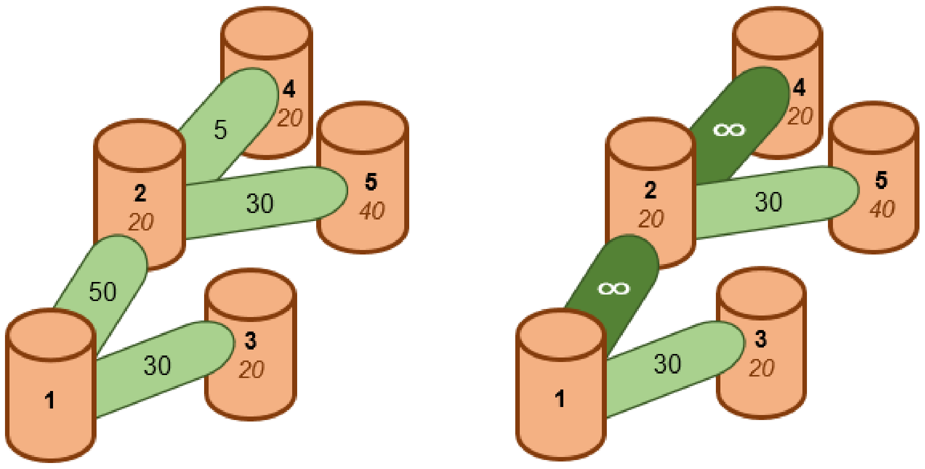
```
5 2
20 1 50
20 1 30
20 2 5
40 2 30
```

Sample Output

```
90
```

Explanation

There are 5 cisterns and we're allowed to upgrade 2 of the pipes. The network of cisterns and pipes is depicted in the following diagrams. The numbers on the pipes represent the maximum rates of chocolate milk flow, and the bottom numbers on the cisterns represent the rate at which chocolate milk is being pumped into them. The left diagram depicts the original configuration, and the right diagram depicts the system after pipes have been upgraded.



By upgrading the pipes flowing down from cisterns 2 and 4, cistern 1 in the cafeteria is able to receive 20 litres/second from cistern 3, 20 litres/second from cistern 2 directly, and 50 litres/second from cistern 2 indirectly (20 of which came from cistern 4 and 30 of which came from cistern 5). In total, this allows cistern 1 to receive 20 + 20 + 50 = 90 litres/second.