Amazing knap sack problem

<https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050eda/00000000001198c3>

Problem

Duda the rock monster lives in the enchanted forest and has collected **N** *energy stones* for lunch. Since he has a small mouth, he eats energy stones one at a time. Some stones are tougher than others! The i-th stone takes him **Si** seconds to eat.

Duda eats energy stones to get *energy*. Different stones give him different amounts of energy. Furthermore, the stones lose energy over time. The i-th stone initially contains **Ei** units of energy and will lose **Li** units of energy each second. When Duda starts to eat a stone, he will receive all the energy the stone contains immediately (no matter how much time it takes to actually finish eating the stone). The stone's energy stops decreasing once it hits zero.

What is the largest amount of energy Duda could receive from eating his stones?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing the integer **N**, the number of energy stones Duda has. Then, there are **N** more lines, the i-th of which contains the three integers **Si**, **Ei** and **Li**, as described above.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the maximum amount of energy Duda could receive from eating stones.

Limits

Time limit: 30 seconds per test set.  
Memory limit: 1GB.  
1 ≤ **T** ≤ 100.  
1 ≤ **N** ≤ 100.  
1 ≤ **Si** ≤ 100.  
1 ≤ **Ei** ≤ 105.  
0 ≤ **Li** ≤ 105.

Test set 1 (Visible)

All stones take the same amount of time to eat. That is: **Si** = **Sj** for all i and j.

Test set 2 (Hidden)

There are no additional constraints beyond the general Limits.

Sample

|  |  |
| --- | --- |
| Input | Output |
| 3  4  20 10 1  5 30 5  100 30 1  5 80 60  3  10 4 1000  10 3 1000  10 8 1000  2  12 300 50  5 200 0 | Case #1: 105  Case #2: 8  Case #3: 500 |

**int M=1e9+7;**

**const int sz=1e4+1;**

**const int OO= 0x3f3f3f3f;**

**struct all{**

**ll seconds,energy,loss;**

**};**

**int n;**

**vector<all>v;**

**bool cmp(all f,all s)**

**{**

**return f.loss\*f.seconds>s.seconds\*s.loss;**

**}**

**ll dp[101][101\*101];**

**ll solve(int i,int seconds)**

**{**

**if (i==n){return 0;}**

**ll &ret=dp[i][seconds];**

**if (~ret){return ret;}**

**ret=-OO;**

**ret=max(ret,solve(i+1,seconds));**

**ret=max(ret,solve(i+1,seconds+v[i].seconds)+max((ll)0,v[i].energy-(v[i].loss\*seconds)));**

**return ret;**

**}**

**int main()**

**{**

**//myf.open("file.txt");**

**//freopen("task.in", "r", stdin);**

**//freopen("output.txt", "w", stdout);**

**ios\_base::sync\_with\_stdio(0),cin.tie(0),cout.tie(0);**

**int tt;**

**cin>>tt;**

**for(int t=1;t<=tt;t++)**

**{**

**v.clear();**

**dpp(dp,-1);**

**cin>>n;**

**for (int i=0;i<n;i++)**

**{**

**ll x,y,z;**

**cin>>x>>y>>z;**

**v.push\_back({x,y,z});**

**}**

**sort(v.begin(),v.end(),cmp);**

**ll ans=solve(0,0);**

**cout<<"Case #"<<t<<": "<<ans<<endl;**

**}**

**return 0;**

**}**

Amazing dynamic programming problem

<https://codeforces.com/contest/414/problem/B>

B. Mashmokh and ACM

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

*Mashmokh's boss, Bimokh, didn't like Mashmokh. So he fired him. Mashmokh decided to go to university and participate in ACM instead of finding a new job. He wants to become a member of Bamokh's team. In order to join he was given some programming tasks and one week to solve them. Mashmokh is not a very experienced programmer. Actually he is not a programmer at all. So he wasn't able to solve them. That's why he asked you to help him with these tasks. One of these tasks is the following.*

A sequence of *l* integers *b*1, *b*2, ..., *bl* (1 ≤ *b*1 ≤ *b*2 ≤ ... ≤ *bl* ≤ *n*) is called *good* if each number divides (without a remainder) by the next number in the sequence. More formally https://codeforces.com/predownloaded/99/fd/99fdd12414bd8efbc325fce766912a87af082646.png for all *i* (1 ≤ *i* ≤ *l* - 1).

Given *n* and *k* find the number of good sequences of length *k*. As the answer can be rather large print it modulo 1000000007 (109 + 7).

**Input**

The first line of input contains two space-separated integers *n*, *k* (1 ≤ *n*, *k* ≤ 2000).

**Output**

Output a single integer — the number of good sequences of length *k* modulo 1000000007 (109 + 7).

**Examples**

**input**

**Copy**

3 2

**output**

**Copy**

5

**input**

**Copy**

6 4

**output**

**Copy**

39

**input**

**Copy**

2 1

**output**

**Copy**

2

**Note**

In the first sample the good sequences are: [1, 1], [2, 2], [3, 3], [1, 2], [1, 3].

**int M=1e9+7;**

**const int sz=1e4+1;**

**const int OO= 0x3f3f3f3f;**

**ll dp[2010][2010];**

**ll n,k;**

**ll solve(int i,int l)**

**{**

**if (i>n){return 0;}**

**if (l==k){return 1;}**

**ll &ret=dp[i][l];**

**if (~ret){return ret;}**

**ret=0;**

**for (int j=i;j<=n;j+=i)**

**{**

**ret=add(ret,solve(j,l+1),M);**

**}**

**return ret%M;**

**}**

**int main()**

**{**

**//myf.open("file.txt");**

**//freopen("task.in", "r", stdin);**

**//freopen("output.txt", "w", stdout);**

**ios\_base::sync\_with\_stdio(0),cin.tie(0),cout.tie(0);**

**cin>>n>>k;**

**dpp(dp,-1);**

**cout<<solve(1,0);**