# Problem A. ACM and ICPC

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Consider a $ACM$ trees with next features:

- An non-negative integer value is assigned to each vertice;
- Each vertice have no more than two children.

Lets call the ACM tree *ICPC*, if it have next additional features:

- Value of each vertice is sum of values of its children;
- Value of leaf vertice is not greater than 1.

Given an ACM tree. We can do following operarions:

- add 1 to value of some vertice;
- subtract 1 from value of some vertice;

Find out minimal number of operations needed to transform given ACM tree to the ICPC tree.

## Input

First line of the input contains one integer $n$ — number of vertice in the given ACM tree ($1 \le n \le 5000$). Second line contains two integers $a_i$ ($0 \le a_i \le 5000$); $i$-th of those integers denotes value of $i$-th vertice. $i$-th of next $n-1$ lines contains two integers $a$ and $b$ ($1 \le a, b \le n$) denoting that vertices $a$ and $b$ are connected by a edge. Root of the ACM tree is placed at the vertice with number 1.

## Output

Print minimum number of operations, needed to transform given ACM tree into ICPC tree.

## Examples

| standard input | standard output |
|---|---|
| 2<br>1 0<br>1 2 | 1 |
| 5<br>5 1 3 0 1<br>1 2<br>1 3<br>3 4<br>3 5 | 4 |

# Problem B. Beautiful Cities

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 2 seconds |
| Memory limit: | 512 mebibytes |

Little Petya likes travelling a lot. There are $N$ cities in his country Berland. These cities are located on a single line. Petya numbered them from 1 to $N$ in the increasing order of their beauty. Petya lives in the city 1 and he wants to get to the city $N$. He doesn't want to spoil the impression about the trip, so he decided to visit cities in the increasing order of their numbers (remember, that he numbered them in the increasing order of their beauty).

To move between the cities Petya chose the only aircompany in his country "Berland Airlines". The cost of flight from the city $i$ to the city $j$ equals $c_i \cdot |x_i - x_j| + t_j$ , where $x_i$ is the coordinate of the city $i$, $x_j$ is the coordinate of the city $j$, $c_i$ is the cost of a single unit of fuel in the city $i$ and $t_j$ is the tax for entering the city $j$. Petya expects this trip to be the best trip in his life, so he decided to spend as much money as possible on it. Help him to determine the maximum amount of money he can spend. Note that it is not necessary to visit all cities.

## Input

The first line of input contains one integer number $N$ ($1 \le N \le 10^5$), which is the number of cities in Berland. The $i$-th of the next $N$ lines contains 3 numbers: $x_i$ ($-10^6 \le xi \le 10^6$), $c_i$ ($1 \le ci \le 10^6$) and $t_i$ ($1 \le t_i \le 10^6$). All coordinates of cities are distinct.

## Output

Print the required largest amount of money that Petya can spend to get from the city 1 to the city $N$. It is guaranteed that the answer will not exceed $10^{12}$.

## Examples

| standard input | standard output |
|---|---|
| 4<br>5 10 2<br>0 1 10<br>15 3 14<br>17 2 3 | 123 |
| 1<br>709 50 8 | 0 |

## Note

Here are two possible optimal routes for the first test case: $1 \to 4$, $1 \to 3 \to 4$.

# Problem C. Christmas Sale

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Little Petya likes rectangles a lot. Petya gave his mother a list of rectangles he wants to get for Christmas. Each rectangle has width $w$ and height $h$. Petya's mother wants to make Petya happy and buy all the rectangles from his list. She went to the store and found out that the price of a rectangle is equal to its area. Luckily, the store has Christmas sale, so you can buy rectangles in sets. The cost of one set is calculated as the width of the widest rectangle multiplied by the height of the highest rectangle of this set. Note that you can't rotate the rectangles (thus swap their width and height). Help Petya's mother buy all the rectangles from the Petya's list spending a minimum amount of money on this.

## Input

First line contains an integer $N$ ($1 \le N \le 10^5$), which is the number of rectangles in the Petya's list. Each of the next $N$ lines contains 2 positive integers that do not exceed $10^6$, which represent width and height of the rectangle.

## Output

Output the smallest amount of money that Petya's mom can spend to buy all the rectangles from the Petya's list.

## Examples

| standard input | standard output |
|---|---|
| 4<br>100 1<br>15 15<br>20 5<br>1 100 | 500 |
| 5<br>1 10<br>2 20<br>3 30<br>4 40<br>10 1 | 170 |

# Problem D. Determine Robot's Score

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 seconds |
| Memory limit: | 512 mebibytes |

You have entered a robot in a Robot Challenge. A course is set up in a 100m by 100m space. Certain points are identified within the space as targets. They are ordered – there is a target 1, a target 2, etc. Your robot must start at (0,0). From there, it should go to target 1, stop for 1 second, go to target 2, stop for 1 second, and so on. It must finally end up at, and stop for a second on, (100,100).

Each target except (0,0) and (100,100) has a time penalty for missing it. So, if your robot went straight from target 1 to target 3, skipping target 2, it would incur target 2's penalty. Note that once it hits target 3, it cannot go back to target 2. It must hit the targets in order. Since your robot must stop for 1 second on each target point, it is not in danger of hitting a target accidentally too soon. For example, if target point 3 lies directly between target points 1 and 2, your robot can go straight from 1 to 2, right over 3, without stopping. Since it didn't stop, the judges will not mistakenly think that it hit target 3 too soon, so they won't assess target 2's penalty. Your final score is the amount of time (in seconds) your robot takes to reach $(100, 100)$, completing the course, plus all penalties. Smaller scores are better.

Your robot is very maneuverable, but a bit slow. It moves at 1 m/s, but can turn very quickly. During the 1 second it stops on a target point, it can easily turn to face the next target point. Thus, it can always move in a straight line between target points.

Because your robot is a bit slow, it might be advantageous to skip some targets, and incur their penalty, rather than actually maneuvering to them. Given a description of a course, determine your robot's best (lowest) possible score.

## Input

There will be several test cases. Each test case will begin with a line with one integer, $N$ $(1 \le N \le 1000)$ which is the number of targets on the course. Each of the next $N$ lines will describe a target with three integers, $X$, $Y$ and $P$, where $(X, Y)$ is a location on the course $(1 \le X, Y \le 99$, $X$ and $Y$ in meters) and $P$ is the penalty incurred if the robot misses that target $(1 \le P \le 100)$. The targets will be given in order — the first line after $N$ is target 1, the next is target 2, and so on. All the targets on a given course will be unique — there will be at most one target point at any location on the course. End of input will be marked by a line with a single 0.

## Output

For each test case, output a single decimal number, indicating the smallest possible score for that course. Your answer must have error $10^{-3}$ or less. Print each answer on its own line, and do not print any blank lines between answers.

## Example

| standard input | standard output |
|---|---|
| 1<br>50 50 20<br>3<br>30 30 90<br>60 60 80<br>10 90 100<br>3<br>30 30 90<br>60 60 80<br>10 90 10<br>0 | 143.421<br>237.716<br>154.421 |

# Problem E. Euclid and Odysseus

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 3 seconds |
| Memory limit: | 512 mebibytes |

Famous Greek mathematician Euclid proposed Odysseus to solve the following problem.

Given is a segment $A$ of integer length $L$. Consider sequences $T$ of right triangles with integer side lengths built in the following way. First, we try to build a triangle $T_1$ where $A$ is the hypotenuse. In case of success, we choose one of the two legs of $T_1$, call it $B$ and try to build a right triangle $T_2$ where $B$ is the hypotenuse. We continue in such a way while it is possible to build the next triangle.

As Cunning is Odysseus' second name, he asked you to help him with this problem.

## Input

Input contains one integer $L$, the length of the starting segment ($1 \le L \le 10^6$).

## Output

Print one integer: the maximum possible number of right triangles with integer side lengths in the sequence $T$ if we start with the given length $L$ as the hypotenuse.

## Examples

| standard input | standard output |
|---|---|
| 2 | 0 |
| 5 | 1 |
| 13 | 2 |

# Problem F. Find The Optimal Code

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Binary code is a mapping of characters of some alphabet to the set of finite length bit sequences. For example, standard ASCII code is a fixed length code, where each character is encoded using 8 bits.

Variable length codes are often used to compress texts taking into account the frequencies of occurence of different characters. Characters that occur more often get shorter codes, while characters occuring less often — longer ones.

To ensure unique decoding of variable length codes so called *prefix codes* are usually used. In a prefix code no code sequence is a proper prefix of another sequence. Prefix code can be easily decoded scanning the encoded sequence from left to right, since no code is the prefix of another, one always knows where the code for the current character ends and the new character starts.

Among prefix codes, the optimal code is known, so called Huffman code. It provides the shortest possible length of the text among all prefix codes that separatly encode each character with an integer number of bits.

However, as many other codes, Huffman code does not preserve character order. That is, Huffman codes for lexicographically ordered characters are not necessarily lexicographicaly ordered.

In this problem you are asked to develop a prefix code that would be optimal for the given text among all order-preserving prefix codes. Code is called order-preserving if for any two characters the code sequence for the character that goes earlier in the alphabet is lexicographically smaller.

Since text itself is not essential for finding the code, only the number of occurences of each character is important, only this data is given.

## Input

The first line of the input file contains $n$ — the number of characters in the alphabet ($2 \le n \le 2000$). The next line contains $n$ integer numbers — the number of occurences of the characters in the text for which the code must be developed (numbers are positive and do not exceed $10^9$). Characters are described in the alphabetical order.

## Output

Output $n$ bit sequences, one on a line — the optimal order-preserving prefix code for the described text.

## Example

| standard input | standard output |
|---|---|
| 5<br>1 8 2 3 1 | 00<br>01<br>10<br>110<br>111 |

# Problem G. Guerilla by Lawrence of Arabia

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 2 seconds |
| Memory limit: | 512 mebibytes |

T. E. Lawrence was a controversial figure during World War I. He was a British officer who served in the Arabian theater and led a group of Arab nationals in guerilla strikes against the Ottoman Empire. His primary targets were the railroads. A highly fictionalized version of his exploits was presented in the blockbuster movie, "Lawrence of Arabia".

You are to write a program to help Lawrence figure out how to best use his limited resources. You have some information from British Intelligence. First, the rail line is completely linear — there are no branches, no spurs. Next, British Intelligence has assigned a Strategic Importance to each depot — an integer from 1 to 5. A depot is of no use on its own, it only has value if it is connected to other depots. The Strategic Value of the entire railroad is calculated by adding up the products of the Strategic Values for every pair of depots that are connected, directly or indirectly, by the rail line. Consider this railroad:

4=====5=====1=====2

Its Strategic Value is $4 \cdot 5 + 4 \cdot 1 + 4 \cdot 2 + 5 \cdot 1 + 5 \cdot 2 + 1 \cdot 2 = 49$.

Now, suppose that Lawrence only has enough resources for one attack. He cannot attack the depots themselves — they are too well defended. He must attack the rail line between depots, in the middle of the desert. Consider what would happen if Lawrence attacked this rail line right in the middle:

4=====5==X==1=====2

The Strategic Value of the remaining railroad is $4 \cdot 5 + 1 \cdot 2 = 22$. But, suppose Lawrence attacks between the 4 and 5 depots:

4==X==5=====1=====2

The Strategic Value of the remaining railroad is $5 \cdot 1 + 5 \cdot 2 + 1 \cdot 2 = 17$. This is Lawrence's best option.

Given a description of a railroad and the number of attacks that Lawrence can perform, figure out the smallest Strategic Value that he can achieve for that railroad.

## Input

There will be several data sets. Each data set will begin with a line with two integers, $n$ and $m$. $n$ is the number of depots on the railroad ($1 \le n \le 1000$), and $m$ is the number of attacks Lawrence has resources for ($0 \le m < n$). On the next line will be $n$ integers, each from 1 to 5, indicating the Strategic Value of each depot in order. End of input will be marked by a line with $n = 0$ and $m = 0$, which should not be processed.

## Output

For each data set, output a single integer, indicating the smallest Strategic Value for the railroad that Lawrence can achieve with his attacks. Output each integer in its own line.

## Example

| standard input | standard output |
|---|---|
| 4 1<br>4 5 1 2<br>4 2<br>4 5 1 2<br>0 0 | 17<br>2 |