The traffic network in a country consists of N cities (labeled with integers 1 to N) and N-1 roads connecting the cities. There is a **unique** path between **each pair of different cities**.

Because of the many years of lazy maintenance the roads are pretty damaged and for each road two numbers A and B are known – the integer A represents the **current** time (in seconds) needed to travel along the road, and the integer B represents the **smallest possible time** (in seconds) needed to travel along this road if we repair **all the damage**.

We want to invest a certain amount of money into road repair. For a particular road, the result will be proportional to the amount of invested money. **For each euro** invested in some road, the time needed to travel along that road will be **reduced by one second** (the amount of money invested in some road has to be an **integer**). The travel time cannot be reduced beyond the smallest possible time B described above.

We are given a certain amount of money. We want to distribute this money along different roads in such a way that the **time** needed to travel from the **city 1** to the **most distant city** (after all the repairs) is **as small as possible**.

Write a program that will find this smallest time.

input data

The first line of input contains two integers N and K, $2 \leq N \leq 100\,000$, $0 \leq K \leq 1\,000\,000$, the number of cities and the total amount of money (in euros).

Each of the next N-1 lines contains four integers X, Y, A and B, $0 \leq B \leq A \leq 10\,000$. It means that there is a road between cities X and Y, with the numbers A and B representing the current time and the minimum time as described above.

output data

The first and only line of output should contain a single integer – the minimum time from the task description.

examples

input

```
3 200
1 2 200 100
2 3 450 250
```

output

```
450
```

input

```
5 11
1 2 10 5
1 3 3 2
1 4 9 6
3 5 7 3
```

output

```
6
```

input

```
11 12
1 2 7 5
1 3 20 15
2 4 10 8
2 5 5 3
2 6 6 2
4 7 3 0
4 8 7 2
5 9 8 4
5 10 9 8
5 11 6 5
```

output

```
17
```