C. Cloud Computing

time limit per test 3 seconds
memory limit per test 256 megabytes
input standard input
output standard output

Buber is a Berland technology company that specializes in waste of investor's money. Recently Buber decided to transfer its infrastructure to a cloud. The company decided to rent CPU cores in the cloud for nn consecutive days, which are numbered from $1$1 to nn. Buber requires kk CPU cores each day.

The cloud provider offers mm tariff plans, the ii-th tariff plan is characterized by the following parameters:

- $l_i$li and $r_i$ri — the ii-th tariff plan is available only on days from $l_i$li to $r_i$ri, inclusive,
- $c_i$ci — the number of cores per day available for rent on the ii-th tariff plan,
- $p_i$pi — the price of renting one core per day on the ii-th tariff plan.

Buber can arbitrarily share its computing core needs between the tariff plans. Every day Buber can rent an arbitrary number of cores (from 0 to $c_i$ci) on each of the available plans. The number of rented cores on a tariff plan can vary arbitrarily from day to day.

Find the minimum amount of money that Buber will pay for its work for nn days from $1$1 to nn. If on a day the total number of cores for all available tariff plans is strictly less than kk, then this day Buber will have to work on fewer cores (and it rents all the available cores), otherwise Buber rents exactly kk cores this day.

Input

The first line of the input contains three integers nn, kk and mm ($1 \le n, k \le 10^6, 1 \le m \le 2 \cdot 10^5$ 1≤n,k≤106,1≤m≤2·105) — the number of days to analyze, the desired daily number of cores, the number of tariff plans.

The following mm lines contain descriptions of tariff plans, one description per line. Each line contains four integers $l_i$li, $r_i$ri, $c_i$ci, $p_i$pi ($1 \le l_i \le r_i \le n$ 1≤li≤ri≤n, $1 \le c_i, p_i \le 10^6$ 1≤ci,pi≤106), where $l_i$li and $r_i$riare starting and finishing days of the ii-th tariff plan, $c_i$ci — number of cores, $p_i$pi — price of a single core for daily rent on the ii-th tariff plan.

Output

Print a single integer number — the minimal amount of money that Buber will pay.

Examples

input Copy

```
5 7 3
1 4 5 3
1 3 5 2
2 5 10 1
```

output Copy

```
44
```

input Copy

```
7 13 5
2 3 10 7
3 5 10 10
1 2 10 6
4 5 10 9
3 4 10 8
```

output Copy

```
462
```

input Copy

```
4 100 3
3 3 2 5
1 1 3 2
2 4 4 4
```

output Copy

```
64
```

```cpp
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define MAXP 1000000
#define MAXN 1000000
using namespace std;
pair<ll,ll>tree[4*MAXP];
vector<int>add[MAXN+5], del[MAXN+5];
struct Data{
    int l,r,c,p;
}tariff[200005];
bool cmp(Data x, Data y){
    return x.p<y.p;
}
void update(int node,int b,int e,int p,int c,int f){
    if(b==p&&e==p){
        tree[node].first  += c*f;
        tree[node].second += (1LL)*b*c*f;
        return;
    }
    int lson = (node*2), rson = lson+1, m=(b+e)/2;
    if(p<=m) update(lson, b, m, p, c, f);
    else update(rson, m+1, e, p, c, f);
    tree[node].first  = tree[lson].first  + tree[rson].first;
    tree[node].second = tree[lson].second + tree[rson].second;
}
ll query(int node,int b,int e,int k){
    if(tree[node].first<=k)return tree[node].second;
    if(k==0) return 0;
    if(b==e) return (1LL)*b*k;

    int lson = (node*2), rson = lson+1, m=(b+e)/2;
    if(tree[lson].first>=k) return query(lson, b, m, k);
    else return tree[lson].second + query(rson, m+1, e, k-tree[lson].first);
}
int main(){
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int n,k,m; cin>>n>>k>>m;
    for(int i=0; i<m; i++){
        int l,r,c,p; cin>>l>>r>>c>>p;
        tariff[i].l = l, tariff[i].r = r;
        tariff[i].c = c, tariff[i].p = p;
    }
    //sort(tariff, tariff+m, cmp);
    for(int i=0; i<m; i++){
        add[tariff[i].l].push_back(i);
        del[tariff[i].r+1].push_back(i);
    }
    memset(tree,0,sizeof(tree));

    ll ans = 0;
    for(int i=1; i<=n; i++){
        for(int j=0; j<(int)add[i].size(); j++){
            int p = tariff[add[i][j]].p;
            int c = tariff[add[i][j]].c;
            update(1, 1, MAXP, p, c, +1);
        }
        for(int j=0; j<(int)del[i].size(); j++){
            int p = tariff[del[i][j]].p;
            int c = tariff[del[i][j]].c;
            update(1, 1, MAXP, p, c, -1);
        }
        ll ret = query(1,1,MAXP,k);
        ans += ret;
    }
    cout << ans << endl;
    return 0;
}
```