#include <bits/stdc++.h>

using namespace std;

typedef long long int ll;

int N, Q;

struct LazySegmentTree {

private:

int n;

vector<ll> node, lazy;

public:

LazySegmentTree(vector<ll> v) {

int sz = (int)v.size();

n = 1; while(n < sz) n \*= 2;

node.resize(2\*n-1);

lazy.resize(2\*n-1, 0);

for(int i=0; i<sz; i++) node[i+n-1] = v[i];

for(int i=n-2; i>=0; i--) node[i] = node[i\*2+1] + node[i\*2+2];

}

void eval(int k, int l, int r) {

if(lazy[k] != 0) {

node[k] += lazy[k];

if(r - l > 1) {

lazy[2\*k+1] += lazy[k] / 2;

lazy[2\*k+2] += lazy[k] / 2;

}

lazy[k] = 0;

}

}

void add(int a, int b, ll x, int k=0, int l=0, int r=-1) {

if(r < 0) r = n;

eval(k, l, r);

if(b <= l || r <= a) return;

if(a <= l && r <= b) {

lazy[k] += (r - l) \* x;

eval(k, l, r);

}

else {

add(a, b, x, 2\*k+1, l, (l+r)/2);

add(a, b, x, 2\*k+2, (l+r)/2, r);

node[k] = node[2\*k+1] + node[2\*k+2];

}

}

ll getsum(int a, int b, int k=0, int l=0, int r=-1) {

if(r < 0) r = n;

eval(k, l, r);

if(b <= l || r <= a) return 0;

if(a <= l && r <= b) return node[k];

ll vl = getsum(a, b, 2\*k+1, l, (l+r)/2);

ll vr = getsum(a, b, 2\*k+2, (l+r)/2, r);

return vl + vr;

}

};

int main() {

cin >> N >> Q;

LazySegmentTree seg( vector<ll>(N, 0) );

for(int i=0; i<Q; i++) {

int query; cin >> query;

if(query == 0) {

int s, t, x; cin >> s >> t >> x;

seg.add(s-1, t, x);

}

else {

int s, t; cin >> s >> t;

cout << seg.getsum(s-1, t) << endl;

}

}

return 0;

}