Meno: Michal Korbela
trieda: sexta
škola: Gymnázium J.J. Bánovce
úloha č.6

Dá sa to spraviť aj tak, že pre každý vrchol túto hodnotu vyrobíme rekurzívnym prehľadávaním .
To je však príliš zdĺhavé – čas – $O(n^2)$

Preto si všimnime niektoré zaujímavosti
keď bude graf jedna čiara – žiadny vrchol nebude mať viac ako 2 spojenia, tak potom ak poznáme celkové nepohodlie, tak vieme v konštantnom čase vyrátať aj nepohodlie toho susedného.

A to ale len v tedy, keď poznáme nepohodlie pred a nepohodlie za ním + súčet batožiny pred a súčet batožiny za ním.

Toto môžeme aplikovať aj v tedy, keď majú vrcholy aj viac ako 2 hrany.

1. vyberieme si nejaký vrchol
2. Chceme vyrátať predné nepohodlie všetkých miest ktoré s ním susedia
3. na to potrebujeme vedieť zadné nepohodlie (nepohodlie zo strany toho prvku, odkiaľ sme prišli) a predné nepohodlia všetkých ostatných prvkov.
4. Na vypočítanie predných nepohodlí týchto prvkov potrebujeme potrebujeme predné nepohodlia prvkov ostatných, ktoré s ním susedia.....

takže vyberieme si nejaký vrchol a s neho rekurzívne prehliadneme všetky vrcholy a vypočítame všetky predné nepohodlia pre všetky prvky okrem začiatočného(ten nemá predná a ani zadné nepohodlie).

Potom vieme vypočítať celkové nepohodlie začiatočného prvku a keď vieme celkové nepohodlie začiatočného vrcholu, tak potom vieme vypočítať aj nepohodlie susedného vrcholu.
To rozoberiem neskôr.

keďže každý vrchol prehliadneme konštantný počet krát – časová zložitosť O(n) n – počet vrcholov pamäťová zložitosť O(n) – zapamätáme si pre každý vrchol konštantný počet údajov – jeho predné nepohodlie, jeho celkové nepohodlie a jeho súčet prednej batožiny, jeho batožina a jeho cesta k ďalšiemu vrcholu.

Takže pamäťová zložitosť O(n)

Vysvetlivky:
celkové nepohodlie daného vrcholu sa vypočíta ako súčet zadného nepohodlia a predného nepohodlia.
Začiatočný vrchol nemá zadné nepohodlie, ale jeho celkové nepohodlie vieme vyrátať ako súčet nepohodlí za každej jednej hrany(cesty)(vrcholy, ku ktorým sa dostaneme danou cestou, tak nepohodlie ich)
Takže keby graf bola čiara – každý vrchol susedí s max 2 vrcholmi, tak potom zadné nepohodlie daného vrcholu je nepohodlie všetkých vrcholov, ktoré sú za ním a predné nepohodlie je súčet nepohodlí vrcholov, ktoré sú pred ním.
Začiatočnému vrcholu sú všetky vrcholy pred ním, takže začiatočný vrchol nemá zadné nepohodlie.
Keď už prejdeme zo začiatočného vrchola na jeho susedný, tak predné nepohodlie tohto vrchola bude súčet všetkých nepohodlí vrcholov, ak nemusia prejsť cez predchádzajúci vrchol. Zadné nepohodlie bude súčet všetkých nepohodlí, ktoré musia prejsť cez predchádzajúci vrchol.

Keď chceme vypočítať predné nepohodlie v prvej funkcii zo všetkých predných nepohodlí jeho predných prvkov, tak nepohodlie z jedného susedného vrchola sa vypočíta ako predné nepohodlie toho vrchola + (súčet batožiny toho predného vrchola + jeho batožina)*dĺžka ktorú musia prejsť z toho predného vrchola do daného vrchola.
Takto vyrátame predné nepohodlie.
V druhej funkcii poznáme celkové nepohodlie zadného vrchola(vrchola z ktorého ideme) a predné nepohodlie tohto daného vrchola.
Potom predné nepohodlie vrchola, z ktorého ideme, ale len vrcholov, ktoré musia prejsť cez daný vrchol ako celkové nepohodlie predchádzajúceho vrchola – (predné nepohodlie daného vrchola + (predná batožina daného vrchola + batožina daného vrchola)*vzdialenosť medzi daným a pôvodným(predchádzajúcim) vrcholom.
Ostatné je zadné nepohodlie predchádzajúceho vrchola(aspoň si to tak nazvime, lebo to nieje ozajstné(môže, ale nemusí byť) zadné nepohodlie predchádzajúceho vrchola, ale keď k tomu pripočítame ešte (zadnej batožiny + batožinu zadného vrchola)*vzdialenosť medzi daným a pôvodným vrcholom, tak nám vznikne zadné nepohodlie daného vrchola a vieme, že celkové nepohodlie = zadné + predné a obidve vieme, takže vieme vypočítať aj celkové nepohodlie.

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <stdlib.h>
#include <time.h>

using namespace std;
int bato[200009];                    //batožina pre daný vrchol
struct po{
int b;                               //do ktorého vrcholu cesta smeruje
int d;                               //dĺžka cesty
};

struct f{
long long batop;                     //batožina pred vrcholom
long long sump;                      //nepohodlie pred vrcholom
long long sum;                       //celkové nepohodlie
};

f data[200009];                      //data o vrcholoch


long long batsum=0;
long long mini=1000000000000;        //konštanta nekonečna

int n;
```

```
vector < vector < po > > P;                    //tu si zapamätáme cesty

void p(int vrch,int z){                              //vypočítame predné nepohodlie

vector<po>:: iterator it;
long long sump=0;
long long batop=0;

for(it=P[vrch].begin(); it<P[vrch].end(); it++)
if((*it).b!=z){
p((*it).b, vrch);
sump+=data[(*it).b].sump+(data[(*it).b].batop+bato[(*it).b])*(*it).d;       //vypočítame predné
nepohodlie
batop+=data[(*it).b].batop+bato[(*it).b];                                   //vypočítame súčet
prednej batožiny
}
data[vrch].batop=batop;                                       //zapíšeme všetky údaje
data[vrch].sump=sump;


}


void s(int z, int cur,int d){                  //vypočítame celkové nepohodlie pre každý vrchol

long long sump1=data[cur].sump+(data[cur].batop+bato[cur])*d;  // nepohodlie prechádzajúceho,
ale len s vetvou, kde sa nachádza daný vrchol
long long sumz1=data[z].sum-sump1;          //zadné nepohodlie predchádzajúceho vrcholu
long long sumz = sumz1+(batsum-bato[cur]-data[cur].batop)*d; //zadné nepohodlie daného vrcholu

data[cur].sum=sumz+data[cur].sump;                  //celkové nepohodlie sa rovná súčet zadného a
predného
vector<po>:: iterator it;
for(it=P[cur].begin(); it<P[cur].end(); it++)           //pozrieme sa aj na ďalšie prvky
if((*it).b!=z)
s(cur,(*it).b, (*it).d);




}



int main(){
srand ( time(NULL) );                          //naštartujeme


cin>>n;
P.resize(n+1);
for(int i=1; i<=n; i++)
scanf("%d",&bato[i]);                          //nahádžeme batožinu
```

```
for(int i=1; i<n; i++){
int a,b,d;
//cin>>a>>b>>d;
scanf("%d%d%d",&a,&b,&d);                    //načítame cesty
po temp;
temp.b=b;                                    //keďže cesty sú obojsmerné
temp.d=d;
P[a].push_back(temp);                        //zapíšeme
temp.b=a;
P[b].push_back(temp);
}
srand ( time(NULL) );                        //zoberieme randomný prvok
int zac=rand() % n + 1;

for(int i=1; i<=n; i++)                      //spočítame všetku batožinu
batsum+=bato[i];




vector<po>:: iterator it;
for(it=P[zac].begin(); it<P[zac].end(); it++)          //vypočítame predné nepohodlie pre každý
vrchol okrem začiatočného
p((*it).b,zac);

long long sum=0;                                       //vypočítame celkové nepohodlie začiatočného
for(it=P[zac].begin(); it<P[zac].end(); it++){
sum+=data[(*it).b].sump+(data[(*it).b].batop+bato[(*it).b])*(*it).d;
}
data[zac].sum=sum;                                     //zapíšeme to
mini=sum;                                              //nastavíme toto nepohodlie ako najmenšie

for(it=P[zac].begin(); it<P[zac].end(); it++)          //vypočítame všetky celkové nepohodlia
s(zac,(*it).b,(*it).d);


for(int i=1; i<=n; i++)
if(data[i].sum<mini) mini=data[i].sum;                 //nájdeme najmenšie nepohodlie

cout<<mini<<endl;

}
```