Michal Korbela
septima
Gym. J.J. Bánovce nad Bebravou
úloha 6

Okrem toho ako spočítať všetky súčty po prvku, tak nám to v testovači nezbehne a to ja preto, lebo sa to dá aj rýchlejšie.
Vieme, že nasledujúce 2 intervaly sa líšia od seba len 1 prvkom.
Preto som si spravil takú štruktúru, ktorá si bude udržiavať prvky ktorým treba dať znamienko, bude mať najviac k prvkov.
Ostatné prvky budú uložené v druhej časti štruktúry.

Pridanie aj odobratie prvku zo štruktúry trvá O(log(n)) času.

Štruktúra pozostáva zo štyroch priority_queue, 2 slúžia na udržanie mínusových prvkov.
Ostatné 2 na udržanie ostatných prvkov.

Funguje to asi nasledovne
pridávanie prvku:

najskôr zistíme, či je prvok taký, že naň chceme dať mínus, aj je, tak ho vložíme do fronty high_top

potom ak je vo fronte high_top viac ako k prvkov, tak ten najmenší presunieme do fronty low_top.
Ak prvok nieje hodný mínusu, tak potom ide do low_top.

Odoberanie prvku:
najskôr zistíme, či patrí prvok do fronty high_top alebo low_top.
Keď zistíme do ktorej fronty patrí tak potom tento prvok pridáme buď do fronty low_del alebo high_del. (Samozrejme udržiavame si aj svoje vlastné veľkosti front)

Potom spustíme udržiavanie front a pokiaľ sa prvok nachádza na špici fronty, tak potom pokiaľ sú rovnaké top-y front top a del, tak potom vymažeme oba.

Vždy potom keď odoberieme alebo pridáme prvok, tak potom spustíme funkciu, ktorá skontroluje či nieje niektorá fronta preplnená, alebo zas prázdna a presunie daný prvok(vymaže ho a pridá).

Tu je kód:

```
priority_queue<long long , vector<long long>, less<long long> > high_top;        //minusové prvky
priority_queue<long long , vector<long long>, less<long long> > high_del;

priority_queue<long long , vector<long long>, greater<long long> > low_top;     //ostatné prvky
priority_queue<long long , vector<long long>, greater<long long> > low_del;
```

fronty sú samozrejme usporiadané tak, aby topy prvkov boli 2 nasledujúce prvky (v high sú mínusové, takže na top-e je najväčší prvok a low sú plusové, takže na top-e je najmenší prvok a tak isto sú zoradené aj del fronty)

```
long long high_top_size=0;          //veľkosť high fronty


long long pole[100047];             //tu sa ukladajú všetky čísla v rade
long long sum_high=0;               //súčet mínusových
long long sum_low=0;                // súčet plusových
long long naj=0;                    //najväčší súčet
long long max_minus=0;              //to je inak povedané k – najviac mínusových



void add(int p){                    // pridáme prvok
if(p<0){                            // ak je prvok mínusový, tak ho dáme do high fronty

high_top.push(p);
high_top_size++;                    // veľkosť high fronty
sum_high+=p;                        //jej súčet sa tiež zväčší
udrz();                             //udržiavame prvky – prehodíme z high do low alebo naopak
}
else{                               // inak ho hodíme do low

low_top.push(p);                    // tiež upravíme info o fronte
sum_low+=p;
}
}


void del(int p){                    // mazanie prvku
int del=0;                          // zistenie v ktorej fronte je
if(low_top.size()>0){               // aby nám nehodilo seg.fault keď bude fronta prázdna
if(low_top.top()<=p){               // aj sa nachádza v low fronte

low_del.push(p);                    // zmažme ho
sum_low-=p;                         //upravme súčet
del=1;                              // úspešne sme ho zmazali z low fronty
}
}
if(del==0) {                        // ak prvok ešte nebol zmazaný
high_del.push(p);                   // zmažeme ho z high fronty a upravíme info
sum_high-=p;
high_top_size--;

}

keep_low();                         // zmaž nepotrebné prvky z low
keep_high();                        // u high
udrz();                             // prehod prvky z low a high ak je potrebné
```

```
}

void keep_low(){                        // zmaže prvky na topoch ak nepatria do front – sú zmazané
if(low_top.size()>0 && low_del.size()>0)   // ak nieje ani jedna fronta prázdna
while(low_top.top()==low_del.top()){        // pokiaľ sú na top-och rovnaké prvky, tak ich zmaž
low_top.pop();
low_del.pop();
if(low_top.size()==0 || low_del.size()==0) break;    // ak sú fronty prázdne, tak skonči
}

}


void keep_high(){                           // podobne udrž high
if(high_top.size()>0 && high_del.size()>0)
while(high_top.top()==high_del.top()){
high_top.pop();
high_del.pop();
if(high_top.size()==0 || high_del.size()==0) break;
}

}


void udrz(){                                // prehoď prvky ak tam nepatria

if(high_top_size<max_minus){                // ak je v high fronte málo prvkov
if(low_top.size()>0){                       // ak je v low fronte nejaký prvok
if(low_top.top()<0){                        // ak je kandidát správny
high_top.push(low_top.top());               // hod prvok do high fronty a uprav info
high_top_size++;
sum_high+=low_top.top();
sum_low-=low_top.top();
low_top.pop();                              // zmaž ho z low
keep_low();                                 //udrž low frontu
}
}
}

else if(high_top_size>max_minus){           // inak ak je v high fronte moc prvkov
low_top.push(high_top.top());               // tak prehod 1 z high fronty do low
sum_low+=high_top.top();
sum_high-=high_top.top();                   // uprav info
high_top.pop();
high_top_size--;
keep_high();                                // udrž high
}
```

```
}
```

potom už len jednoduchým cyklom načítame prvky do pola a potom už len skontrolujeme všetky intervaly

```
for(int i=1; i<=n; i++){

add(pole[i]);

if(i==l)
if(sum_low-sum_high>naj)   //musíme otestovať už prvý možný interval, ale prvok ešte neodoberáme
naj=sum_low-sum_high;              // ak je súčet najväčší


if(i>l){                                   // ak je už interval na odoberanie

del(pole[i-l]);                    // odstránime prvok

if(sum_low-sum_high>naj)           // otestujeme naj súčet
naj=sum_low-sum_high;


}


}
```

potom však potrebujeme otestovať aj ku záporným prvkom dávať mínusy
tak nám stačí prenásobiť všetky prvky mínusom

```
for(int i=1; i<=n; i++)
pole[i]*=-1;
```

tak nejak

potom vyčistíme všetky fronty a údaje

```
while(high_top.size()>0)
high_top.pop();
while(low_top.size()>0)
low_top.pop();
while(high_del.size()>0)
high_del.pop();
while(low_del.size()>0)
low_del.pop();
```

high_top_size=0;

sum_high=0;
sum_low=0;

a urobíme súčty znova

Čo sa týka pamäte a času

pamäť – každý prvok si pamätáme najviac 5 krát – 4 krát vo frontách a 1 krát v poli, takže O(n)

čas – operácia add a del nám trvá konštantný počet push a pop, takže O(log(n)) keďže každý prvok musíme najviac 2 razy pridať a najviac 2 razy odobrať.

Cyklus while nebude spôsobovať vo funkciách keep žiadnu zmenu času, pretože každý prvok môžeme z fronty odobrať aj z fronty del najviac raz.