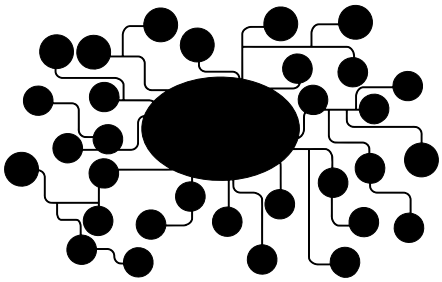
Úloha č. 3
Narušení Systému



Rozmysli, popiš a naprogramuj! 10 b

Tato úloha se skládá ze dvou částí. Tvým úkolem je napsat program a zároveň zdůvodnit proč funguje.

Sotva jsme rozdělili okolní síť do podsítí, začaly se dít věci.

Ze sousedního uzlu z černé, logicky bílé, díry přiběhl uřícený Aragorn a z jeho očí prýštila naléhavost. „Majáky z Katedry softwarového inženýrství jsou zažehnuty!“ křičel, „majáky hoří! Katedra softwarového inženýrství hlásí změnu vzdáleností v Systému.“

Země se zachvěla, dunění jakoby přicházelo od Fakulty elektrotechnické. Šířilo se pod zemí dávnými tunely a chodbami, které kdysi vybudovali hardwaráři, když se stavěl Systém. Gimli okamžitě zbystřil a zaposlouchal se do tónů trpasličích trub, které mocně zněly zemí. „Hardwaráři hlásí změny v Systému,“ oznámil, hrdý na to, jak trpasličí komunikační kanály fungují.

Najednou se otevřena černá, logicky bílá, díra, vylétl z ní šíp a zabodl se do pryskyřice na druhé straně uzlu. Legolas se okamžitě vrhnul k šípu, strhl z něj stuhu, kterou byl ovázaný svitek, a četl: „Drazí elfové, buďte uvědoměni, že doba přesunu mezi uzly Katedry teoretické informatiky a Laboratoře programovacích jazyků se zvýšila na dvojnásobek původní hodnoty. Mezi těmito uzly byla našimi elfskými průzkumníky spatřena zvláštní bytost částečně připomínající stroj a částečně elfa. Domníváme se, že tyto události spolu souvisí.“

Gandalf právě držel v ruce motýla, který mu přinesl další podobnou zprávu. V tom přiletěl další šíp, třetí a čtvrtý a zem duněla dalšími, překřikujícími se troubami.

„Není na co čekat,“ řekl rozhodně Gandalf, „komunikace v Systému nesmí být narušena. Tolik změn ve vzdálenostech naše běžné komunikační kanály nestihnou zpracovat. Je potřeba pomoci udržovat informace o Systému aktuální. Gimli, zruš zbytečná spojení s uzly v síti v našem okolí a zachovej jen ta nejnutnější. Ať zůstane jenom kostra.[1]“

„Gandalfe, ale ani tak nestihneme všechny dotazy na stav Systému vyřizovat dostatečně rychle,“ namítal Legolas. „Ale stihneme,“ řekl jsem, „napadl mě algoritmus!“

Vstup

Na prvním řádku je přirozené číslo $T \leq 10^2$, které určuje počet zadání, která budou následovat.

Následuje T zadání, j-té zadání začíná řádkem s přirozenými čísly $n_j \leq 10^6, e_j \leq 10^6, u_j \leq 10^7, q_j \leq 10^7$, kde n_j značí počet vrcholů, e_j počet hran, u_j počet změn a q_j počet dotazů.

Zbytek zadání tvoří $e_j + u_j + q_j$ řádků organizovaných následovně:

1. e_j řádků, i-tý řádek tvoří čísla $0 \leq a_i < n_j, 0 \leq b_i < n_j, v_i \in \mathbb{Z}$ značící existenci neorientované hrany mezi vrcholy a_i a b_i s hodnotou v_i.

2. u_j řádků, i-tý řádek má tvar `add` t_i a_i b_i v_i a značí úpravu hodnoty hrany mezi vrcholy a_i a b_i v čase $t_i \in \mathbb{Z}$ přidáním celého čísla v_i.

[1] Kostra souvislého grafu je souvislý podgraf, který neobsahuje cykly – je to strom a nemá tedy žádné zbytečné hrany navíc.

3. q_j řádků, i-tý řádek má tvar `max?` t_i a_i b_i a značí dotaz na maximální hodnotu na cestě mezi vrcholy a_i a b_i v čase t_i.

Prvotní seznam hran (e_j řádků) udává neorientovaný strom s ohodnocenými hranami. Každá z následujících u_j úprav mění hodnoty hran přičtením celého čísla k dané hraně, ale hranu nemůže smazat ani přidat[2]. Čas určují celá čísla t_i, s jejíž pomocí se aktualizace hodnot hran řadí na časovou osu. Dotaz v čase t vidí všechny minulé úpravy (tj. úpravy ve všech časech $t_k \leq t$), ale na jeho výsledek nemají vliv úpravy v budoucnosti, tj. v časech ostře větších než t.

Hlavním úkolem je efektivně zodpovědět dotazy na maximální hodnotu na nějaké cestě ve stromě v libovolném bodě na časové ose. Složitost naivního algoritmu poroste lineárně nebo hůř s počtem úprav u_j. Optimální řešení by zvládlo dotazy zodpovídat online v sublineárním čase, tj. zodpovědět dotaz bez předchozí znalosti celé posloupnosti dotazů.

Výstup

Výstupem je $\sum_{j=1}^{T} q_j$ řádků, každý s jedním celým číslem, která odpovídají dotazům v každém zadání.

Vstup

```
1
3 2 2 3
1 2 0
2 3 0
add 2 1 2 3
add 1 1 2 -1
max? 2 1 3
max? 0 1 3
max? 1 1 3
```

Výstup

```
2
0
0
```

V řešení nezapomeňte vysvětlit, jak zkompilovat a spustit váš program, ideálně na Linuxovém systému ve vhodném Docker kontejneru.

[2]Některé hrany ovšem mohou mít nulové či záporné hodnoty.