Korešpondenčný seminár z programovania
XXXIII. ročník, 2015/16
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Návody k úlohám 1. kola zimnej časti kategórie T

V kategórii T neuvádzame vzorové riešenia ale skôr návody na riešenie úloh. Ich cieľom je prezradiť vám hlavnú myšlienku riešenia, aby ste podľa návodu mohli riešenie domyslieť sami. Občas teda vynecháme niektoré drobné detaily, neuvádzame implementácie dátových štruktúr a nerozpisujeme niektoré kroky.

Neuvádzame ani vzorové programy, pretože chceme, aby ste po prečítaní návodu naprogramovali tie úlohy, ktoré ste nespravili počas trvania série. Keď si tieto riešenia sami naprogramujete, naučíte sa tým oveľa viac ako pozeraním sa na zdrojové kódy.

Implementácie všeobecne známych algoritmov a dátových štruktúr môžete nájsť vo vzorových riešeniach kategórií Z a O starších ročníkov KSP alebo aj na internete.

1. Toľko možností...

návod písal Jano
(max. 0 b za popis, 20 b za program)

Označme si $n_1 = n$, $n_5 = \lfloor \frac{n}{5} \rfloor$ (t.j. delenie zaokrúhlené nadol), $n_{10} = \lfloor \frac{n}{10} \rfloor$, $n_{20} = \lfloor \frac{n}{20} \rfloor$.

Potom vieme, že môžeme použiť najviac n_{20} dvadsaťcentových mincí, a keď ich použijeme i, tak už môžeme použiť len $n_{10} - 2i$ desaťcentových mincí. A keď tých použijeme j, tak môžeme použiť už len $n_5 - 4i - 2j$ päťcentových. Zvyšok doplatíme jednocentovými mincami.

Celkový výsledok teda je

$$\sum_{i=0}^{(n_{20})} \sum_{j=0}^{(n_{10}-2i)} \sum_{k=0}^{(n_5-4i-2j)} 1$$

Celá úloha je teda o tom, ako spočítať túto sumu čo najrýchlejšie. Použiť tri vnorené cykly by nebol najrýchlejší prístup :) Tak poďme zjednodušovať. Budeme využívať nasledovné vzorce:

Konštanty sa dajú vybrať pred sumu: $\sum_{i=0}^{n} kf(i) = k \sum_{i=0}^{n} f(i)$

Suma jednotiek: $\sum_{i=0}^{n} 1 = n + 1$

Suma tzv. úsečiek: $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

Suma tzv. štvorcov: $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

(Všetky 4 vzorce sa dajú pomerne ľahko odvodiť.)

Napríklad vzorec *suma jednotiek* nám umožňuje spočítať: $\sum_{k=0}^{n_5-4i-2j} 1 = n_5 - 4i - 2j + 1.$

Celkový výsledok teda je $\sum_{i=0}^{(n_{20})} \sum_{j=0}^{(n_{10}-2i)} (n_5 - 4i - 2j + 1)$ čím sme si ušetrili jeden forcyklus v programe.

Použitím vzorcov o vyberaní konštánt, vzorca pre súčet jednotiek, a vzorca pre súčet jednotiek dostaneme:

$$\sum_{i=0}^{n_{20}} \left(\sum_{j=0}^{(n_{10}-2i)} (n_5 - 4i + 1) + \sum_{j=0}^{(n_{10}-2i)} (-2j) \right)$$

sa rovná

$$\sum_{i=0}^{n_{20}} ((n_{10} - 2i + 1)(n_5 - 4i + 1) - (n_{10} - 2i + 1)(n_{10} - 2i))$$

a to je po úpravách

$$\sum_{i=0}^{n_{20}} (n_{10} - 2i + 1)(n_5 - n_{10} + 1 - 2i)$$

Keby ste toto naprogramovali ako forcyklus, už by ste mali 12 bodov z 20.

Ale vieme ísť ešte o krok ďalej, rozpísať si $(n_{10} - 2i + 1)(n_5 - n_{10} + 1 - 2i) = \text{niečo} + \text{niečo} \cdot i + \text{niečo} \cdot i^2$ a vypočítať celú sumu. Ale to už zvládnete sami :).

Takto dostanete algoritmus s časovou zložitosťou $O(1)$. Na začiatku len spočítate n_1, n_5, n_{10} a n_{20}, hneď ako ich spočítate, zmodulujte ich 10^9+7. Potom len spočítajte jednoduchý vzorec, ktorý ste si odvodili, nezabudnite použiť dosť veľké premenné na uloženie čísel a tiež všetky medzivýsledky modulovať $10^9 + 7$, aby vám tam nič nepretieklo.

Navyše sa vám môže stať, že vo výpočte budete potrebovať deliť dvoma alebo troma. Keďže počítate modulo 10^9+7, potrebujete namiesto toho násobiť inverznými prvkami týchto čísel. Poradíme vám, že $500000004 \cdot 2 \equiv 1 \mod 1\,000\,000\,007$ a $333333336 \cdot 3 \equiv 1 \mod 1\,000\,000\,007$.

Ak vám to nebude fungovať, môžete si naprogramovať aj algoritmus s jedným forcyklom na to, aby ste ľahšie zistili, kde máte chybu.

návod písal Jano

2. Toľko možností v Amerike

(max. 0 b za popis, 20 b za program)

Úloha sa rieši veľmi podobne, ako predošlá úloha *Toľko možností*, preto sa najprv uistite, že rozumiete riešeniu tej úlohy.

V predošlej úlohe sme využili, že hodnota každej mince je deliteľná všetkými hodnotami menších mincí. Napríklad každé použitie 20-centovej mince nám akoby ubralo možnosť použiť dve 10-centové mince alebo 4 päťcentové alebo 20 1-centových. V tejto úlohe však máme 25-centovú mincu, ktorá nie je deliteľná 10-centovou.

Dajú sa vymyslieť komplikované finty, ako sa s tým vysporiadať ale lepšia cesta je využiť nasledujúci trik. Buď pri platení použijeme párny počet alebo nepárny počet 25-centových mincí. Tým pádom si môžeme preformulovať zadanie tak, že pri platení môžeme používať nové 50-centové mince a najviac jednu 25-centovú.

Vlastne len sčítame počet možností ako zaplatiť sumu n pomocou mincí $1, 5, 10, 50$ a sumu $n - 25$ pomocou rovnakej sady mincí. Tieto podúlohy vyriešime veľmi podobne ako predošlú úlohu.

Nasledujúci program obsahuje kostru riešenia s chýbajúcim vzorcom. Je na vás, aby ste daný vzorec doplnili.

Listing programu (C++)

```
#include <iostream>
using namespace std;
typedef long long ll;
#define MOD 1000000007LL
#define idva 500000004LL
#define itri 333333336LL

inline ll mod(ll x) { return x%MOD; }

ll solve(ll n) {
    ll n5 = mod(n/5);
    ll n10 = mod(n/10);
    ll n50 = mod(n/50);

    // Tuto spočítame výsledok, ale sami musíte vymyslieť, ako.
    ll vysledok = 0;

    return mod(mod(vysledok) + MOD);
}

int main() {
    ll n;
    cin >> n;
    cout << mod(solve(n) + (n>=25)*solve(n-25)) << endl;
}
```

návod písal Buj

3. Technologická singularita

(max. 0 b za popis, 20 b za program)

Predstavme si graf s n vrcholmi, pričom z vrcholu a vedie orientovaná hrana do b práve vtedy, keď $P[a] = b$. Potom vrchol b nazveme následníkom a, a vrchol a nazveme predchodcom b. Môžeme si všimnúť, že každý vrchol má práve jedného následníka a práve jedného predchodcu. To je nutná a zároveň postačujúca podmienka na to, aby graf pozostával z orientovaných cyklov (ktoré zodpovedajú cyklom v permutácii) a žiadnych ďalších hrán. Dôkaz tohto tvrdenia nie je ťažký.

Potom vieme ľahko pre ľubovoľný vrchol v nájsť všetky vrcholy, ktoré patria do toho istého cyklu — začneme vrcholom a, potom sa pozrieme na jeho následníka, potom na následníka následníka a, potom na ďalšieho následníka, ... až kým nenarazíme opäť na vrchol a. Všetky takto nájdené vrcholy patria do cyklu s a.

Počet cyklov vieme potom zistiť jednoducho — prechádzame postupne všetkými vrcholmi, a vždy, keď narazíme na neoznačený vrchol, zvýšime počet cyklov o 1 a označíme všetky vrcholy v tom istom cykle ako ten

vrchol. Toto beží v čase $O(n)$. Takto vieme implementovať riešenie hodné 2 bodov bežiace v čase $O(nq)$ — na začiatku a po každej z q výmen spustíme výpočet počtu cyklov.

Chceme viac ako 2 body !!!

Budeme musieť nejako využiť, že po výmene sa **nezmení** permutácia **na hocijakú** inú. Dá sa ľahko dokázať, že po výmene sa počet cyklov zvýši o 1 ak čísla (vrcholy), ktoré sme vymieňali, ležia v rovnakom cykle; ak ležia v rôznych cykloch, počet cyklov sa zníži o 1.

Vyzbrojení týmto tvrdením vieme implementovať nasledujúce riešenie: na začiatku spočítame počet cyklov. Po každej výmene na pozíciách a, b zistíme, či sú a, b v rovnakom cykle, a príslušne upravíme počet cyklov. Zistiť, či a, b ležia v rovnakom cykle vieme napríklad nasledovne: nájdeme všetky pozície v cykle s a, a zistíme, či medzi nimi je aj b. Lepším (ako neskôr uvidíme) sa ale ukáže nasledujúci spôsob: nájdeme všetky pozície v cykle s a, a najväčšiu z nich označíme m_a. Podobne najväčšiu pozíciu v cykle s b označíme m_b. Potom a, b ležia v rovnakom cykle práve vtedy, keď $m_a = m_b$.

Takto dostávame riešenie s časom $O(nq)$. Vo vstupných sadách 2 a 3 sa ale dá dokázať lepší čas $O(n + q^2)$ a máme riešenie hodné 6 bodov.

Chceme viac ako 6 bodov !!!

Nastáva čas zamyslieť sa nad tým, ako rýchlejšie zistiť, či sú vymieňané pozície v rovnakom cykle. **Najprv si ale vyriešime pomocný problém:**

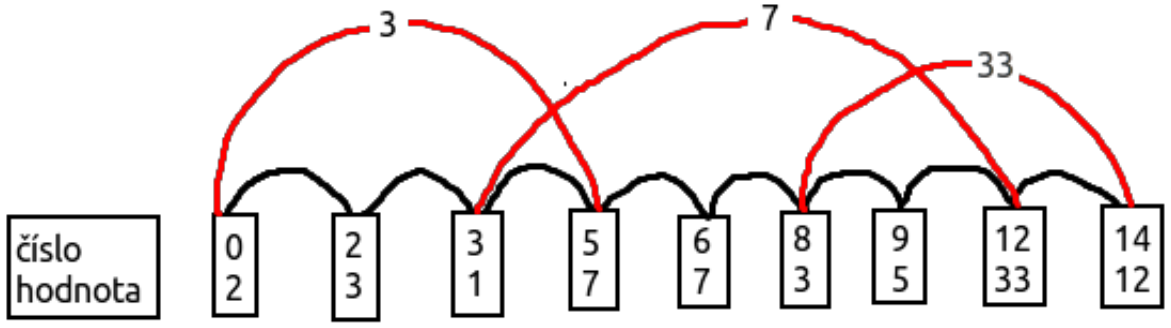
Na začiatku máme vrchol s nejakým číslom a nejakou hodnotou. Postupne prichádzajú udalosti troch typov:

- pýtame sa na najväčšiu hodnotu na vrcholoch s číslami v intervale $\langle a; b)$
- vznikol nový vrchol s číslom a a hodnotou b (máme zaručené, že neexistoval)
- vrchol s číslom a zanikol (máme zaručené, že existoval)

a na každú by sme chceli vedieť rýchlo odpovedať. Pozrime sa najprv na naivné riešenie, ktoré implementuje spájaný zoznam v ktorom si vrcholy pamätá v poradí podľa čísla:

- keď príde udalosť prvého typu, nájde v zozname v $O(n)$ vrchol s najmenším číslom väčším rovným a, a v $O(n)$ sa pozerá na následníkov a až kým nenarazí na vrchol s číslom aspoň b. Spomedzi nájdených vrcholov (okrem b) vyberie ten s najväčšou hodnotou a vráti ju.
- pri udalosti druhého typu v $O(n)$ nájde vrchol s najväčším číslom x, ktoré je menšie ako a. Jeho následník — vrchol s číslom y, spĺňa $y > a$. Vložíme vrchol a "medzi" tieto dva vrcholy (príslušne nastavíme následníkov a predchodcov).
- pri udalosti tretieho typu v $O(n)$ nájde v zozname vrchol s číslom a, zmaže ho, a jeho predchodcovi a následníkovi nastaví následníka / predchodcu.

Výrazne by sa to urýchlilo, keby sme vedeli cestovať spájaným zoznamom rýchlejšie (prejsť viac hrán v jednej operácii). Do nášho spájaného zoznamu teda môžeme pridať akési hrany navyše. V takýchto hranách si pamätáme, akú najväčšiu hodnotu by sme objavili, keby sme tak cestovali v pôvodnom spájanom zozname (toto nazveme hodnotou hrany):

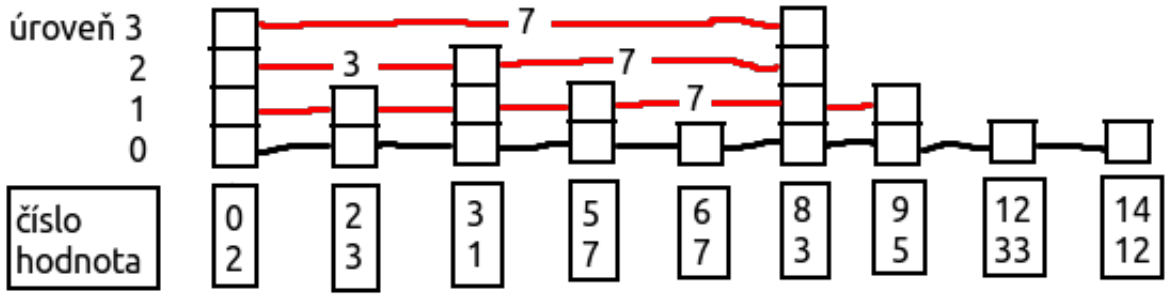


Po pridaní ďalších hrán sa ale naskytá viacero otázok.

- Ktorou hranou sa pri cestovaní zoznamom pohnúť ďalej? Z aktuálneho vrchola ich totiž môže viesť viacero. Toto veľmi závisí od pridaných hrán, no my sa uspokojíme s tým, že si vždy zvolíme najdlhšiu možnú hranu dopredu, ktorou nedosiahneme b (= hranu do vrchola s najväčším číslom menším ako b), ak je interval čísel, ktorých hodnoty nás zaujímajú $\langle a; b)$.

- Keď pridáme nový vrchol x (udalosť druhého typu), každej hrane ab, ktorá ho obsahuje (teda pri ceste z a do b v pôvodnom spájanom zozname by sme prešli x), musíme upraviť jej hodnotu (nastaviť ju na maximum z jej pôvodnej hodnoty a hodnoty x). Musíme preto vedieť rýchlo zistiť, ktoré hrany obsahujú nový vrchol. Tiež s pridaním nového vrchola potrebujeme niekedy pridať nové hrany — inak by sme si pomohli iba o konštantný faktor.
- Keď odoberieme vrchol x, musíme odobrať všetky hrany, ktoré mali aspoň jeden koncový bod x. A tiež upraviť hodnoty hrán, ktoré x obsahovali — to ale nie je také jednoduché ako pri udalostiach druhého typu (rozmyslite si prečo).

Vhodnou dátovou štruktúrou, pomocou ktorej vieme všetky tieto problémy vyriešiť, je **skiplist**. Princíp je nasledovný: pre každý vrchol x si pamätá jeho úroveň $l(x)$. Na každej úrovni k má navyše hrany spájajúce vrcholy s úrovňou aspoň k, a tieto vrcholy s hranami na úrovni k tvoria spájaný zoznam. Inak povedané: pre každý vrchol x a pre každú úroveň $k \leq l(x)$ má skiplist navyše hrany na úrovni k vedúce do najbližších vrcholov s úrovňou aspoň k.



Aby bol skiplist efektívny, úrovne vrcholov sa volia náhodne tak, aby všetky mali úroveň aspoň 0, aby rádovo polovica mala úroveň aspoň 1, rádovo štvrtina s úrovňou aspoň 2, ... A teraz na konkrétnejšie riešenie spomínaných problémov:

- Pri pridaní nového vrchola x mu najprv náhodne (s vhodným rozložením pravdepodobností) určíme úroveň $l(x)$. Následne pôjdeme od najnižšej úrovne po najvyššiu: označme aktuálnu úroveň u. Prvého predchodcu na úrovni u označíme a. Tiež nájdeme prvého následníka na tej úrovni a označíme ho b. Zrejme existuje hrana ab. Ak $u \leq l(x)$, tak ju nahradíme hranami ax, xb (ktorých hodnoty vieme vypočítať vďaka hodnotám na nižšej úrovni). Ak $u > l(x)$, tak upravíme hodnotu hrany ab. (Implementačné detaily, ako napríklad "čo ak neexistuje predchodca alebo následník" prenechávame na domyslenie vám.)
- Odobratie vrchola x: opäť začneme od najnižšej úrovne. Označíme aktuálnu úroveň u, prvého predchodcu na úrovni u označíme a a prvého následníka b. Ak $u \leq l(x)$, zmažeme hrany ax, xb a nahradíme ich hranou ab (ktorej hodnotu spočítame podľa nižšej úrovne). Ak $u > l(x)$, tak upravíme hodnotu hrany ab podľa nižšej úrovne.

Čas každej z týchto operácií závisí na počte úrovní, ktorých bude približne $\log n$.

Vráťme sa teraz k riešeniu pôvodnej úlohy o cykloch v permutáciách. Nie je náročné zovšeobecniť skiplist na cyklický skiplist. Na tomto cyklickom skipliste vykonávame trochu iné operácie — nepridávame a nemažeme žiadne vrcholy, ale vymieňame následníkov. Na to nám stačí implementovať pridanie hrany a odobratie hrany v skipliste, nakoľko výmena na pozíciách a, b pričom následníkmi sú c, d je to isté, ako keby sme najprv zmazali hrany ac, bd a následne pridali hrany ad, bc.

- Pri mazaní hrany xy na úrovni 0 ju zmažeme. Následne postupujeme od najnižšej úrovne po najvyššiu: nech aktuálna úroveň je u. Nájdeme prvého predchodcu x, ktorý má úroveň aspoň u, a označíme ho a. Tiež nájdeme prvého následníka x, ktorý má úroveň aspoň u, a označíme ho b. Zmažeme hranu ab.
- Pri pridaní hrany xy na úrovni 0 ju pridáme. Postupujme od najnižšej úrovne po najvyššiu: nech aktuálna úroveň je u. Nech a, b sú vrcholy s rovnakým významom ako v predchádzajúcom prípade. Pridáme hranu ab, pričom jej hodnotu vieme spočítať z nižšej úrovne.

Môžete si premyslieť implementačné detaily, prípadne naprogramovať si skiplist.

Pomocou skiplistu vieme potom každú výmenu vykonať v čase $O(\log n)$. Celkový čas je teda $O(n + q \log n)$. Toto riešenie dostane 20 bodov.

A dá sa získať aj niečo medzi 6 a 20 bodmi?

Existuje "odmocninové" riešenie bežiace v čase $O(n + q\sqrt{n})$, ktoré implementuje skiplist len s dvoma úrovňami, pričom na druhej je $O(\sqrt{n})$ vrcholov. Toto riešenie bolo odmenené 14 bodmi, ale s nejakou optimalizáciou a tlačením konštánt sa dalo pretlačiť na 20.

Ako vyriešiť úlohu bez skiplistu?

Táto úloha sa dala riešiť aj bez ťažkých dátových štruktúr (akými sú skiplist, treap, a rôzne iné varianty vyvažovaných binárnych vyhľadávacích stromov). Riešenie je založené na nasledujúcej myšlienke:

Predpokladajme, že $n >> q$. Vrcholy, ktoré sa nachádzajú v aspoň jednej výmene, budeme nazývať *nestabilné*. V opačnom prípade vrchol nazveme *stabilný*.

Zamerajme sa na konkrétny nestabilný vrchol a. Pozrime sa na jeho predchodcu, na predchodcu predchodcu, ... až kým nenarazíme na nestabilný vrchol (nie nutne rôzny od a). Postupnosť týchto vrcholov (vrátane prvého a, bez nestabilného vrcholu, na ktorom sme skončili) označme P_0. Zopakujme to isté po prvej výmene a dostaneme postupnosť P_1. Rovnako po druhej, tretej, ... Dá sa ľahko dokázať, že všetky tieto postupnosti sú rovnaké ($P_0 = P_1 = \ldots = P_q$) — teda že výmenami sa "chvost" vrcholu a nemení. (Rozmyslite si, prečo.)

Nech chvost a je $a = a_0, a_1, \ldots, a_k$. Potom vieme chvost a nahradiť jedným novým vrcholom v, pričom jeho následníkom je pôvodný následník a_0, a jeho predchodcom je pôvodný predchodca a_k. Na začiatku teda v $O(n+q)$ zostrojíme reprezentantov chvostov (a upravíme príslušne všetky výmeny), a na tomto pustíme riešenie ktoré obdrží za normálnych okolností 2 body. Reprezentantov chvostov je $O(q)$, takže dostávame celkový čas $O(n + q^2)$ a riešenie dostane 8 bodov.

Na zvýšenie nášho počtu bodov použijeme prístup *divide and conquer (rozdeľuj a panuj)*:

Rozdelíme si všetky otázky na prvú polovicu a druhú polovicu. Permutáciu / graf zmenšíme tým, že si zostrojíme reprezentantov chvostov ako v predchádzajúcom riešení, a problém vyriešime rekurzívne (už iba s prvou polovicou otázok). Druhú polovicu otázok vyriešime podobne, až na to, že počiatočný stav na začiatku druhej polovice je iný od počiatočného stavu na začiatku úlohy. To ale vyriešime jednoducho — prvú polovicu výmen odsimulujeme (v lineárnom čase).

O tomto riešení sa dá dokázať, že má čas $O(n+q \log q)$, a dosiahne plný počet bodov. Jeho nevýhodou oproti riešeniam s ťažkými dátovými štruktúrami je, že nie je online, čo ale na vyriešenie úlohy nebolo potrebné.

4. Tradičné zimné slávnosti

návod písal Jaro
(max. 0 b za popis, 20 b za program)

Na začiatok si rozoberieme jeden triviálny prípad. Ak $n = 2$, odpoveď je snáď zjavná. Všetky ostatné prípady budeme riešiť stromovou rekurziou.

Náš stromček si zavesíme za ľubovoľný nelistový vrchol (vrchol stupňa aspoň 2). V každom vrchole nás budú zaujímať tri hodnoty: Najväčší možný počet ciest, ktoré môžu končiť v danom podstrome a prechádzať cez náš vrchol, koľko z nich **nevieme** uzavrieť priamo v tomto podstrome (a teda musia prechádzať cez náš vrchol) a nakoniec koľko ciest vieme najviac uzavrieť v našom podstrome, pokiaľ už poznáme predošlé dve hodnoty.

Tieto tri hodnoty z listových vrcholov by mali byť celkom zjavné, preto sa zameriame už len na vnútorné vrcholy stromu. V každom máme niekoľko synov s ich hodnotami. Najprv zistíme druhú hodnotu. Pozrieme sa na syna s najväčšou druhou hodnotou a odrátame od nej prvé hodnoty všetkých ostatných[1]. Prvú hodnotu získame tak, že nasčítame prvé hodnoty všetkých synov.

Pozor, hodnota hrany do otca nám môže prvú aj druhú hodnotu trošku zrezať. Porozmýšľajte, čo robiť, pokiaľ by nám po tomto orezaní ostal počet uzatvoriteľných ciest v podstrome ($prva\_hodnota - druha\_hodnota$) nepárny. Nad poslednou hodnotou sa budete musieť trošku zamyslieť. (Nedáme vám predsa celý návod, ako to vypočítať, nie?) Opäť to bude súvisieť s hodnotami synov a s tým, ako presne nám hodnoty oreže hrana do otca.

Taktiež na zamyslenie ostane doriešiť, ako z týchto troch hodnôt v koreni vypočítať celú odpoveď.

5. Turbulencie na dráhe

návod písal Mišo
(max. 0 b za popis, 20 b za program)

Táto úloha nebola veľmi ťažká, bolo treba vedieť základy mechaniky a dať si pozor na aritmetické zákernosti ako je delenie nulou, odmocňovanie záporného čísla a podobne.

[1] Ak dostaneme kladné číslo, je celkom zjavné, že aspoň toľko nám ich musí ostať neuzatvoriteľných, ak záporné, vieme uzavrieť všetky. Jedným možným spôsobom by bolo vždy uzatvoriť cestu medzi dvojicou synov, ktorí majú najvyššiu druhú hodnotu, skúste si dokázať prečo je to tak. Samozrejme, v riešení nás zaujímajú iba hodnoty a teda netreba riešiť, kto s kým a ako sa pospája.

Úloha sa riešila takto: ak sa vozík nachádza na začiatku nejakej úsečky (priamo na jej začiatočnom bode) a vieme, akou rýchlosťou a v ktorom smere sa hýbe, aký je sklon danej úsečky a aká je dlhá, tak vieme vypočítať, ako dlho bude vozíku trvať, kým prejde celou úsečkou, a tiež, ako rýchlo (a či vôbec) sa bude hýbať na jej konci.

Troška fyziky

Povedzme, že úsečka je sklonená pod uhlom α a má dĺžku d. (V skutočnosti nebudeme potrebovať vedieť presný uhol α, ale iba $\sin\alpha$ a $\cos\alpha$. Kosínus uhla zvieraného dvoma vektormi sa ľahko vypočíta ako podiel ich skalárneho súčinu a súčinu ich veľkostí. Sínus sa vypočíta ako podiel ich vektorového súčinu a súčinu ich veľkostí.) Ďalej povedzme, že vozík sa hýbe rýchlosťou v smerom k jej druhému koncovému bodu (čiže v je len nezáporné reálne číslo, vektor rýchlosti vozíka vieme vyrátať ako $(v \cdot \cos\alpha, v \cdot \sin\alpha)$).

Úsečka, po ktorej sa vozík bude hýbať je vlastne naklonená rovina. Pohyb po naklonenej rovine je istý druh zrýchleného pohybu, v ktorom sa zrýchlenie dá odvodiť od gravitačného zrýchlenia a sklonu naklonenej roviny. Oba parametre poznáme, preto zrýchlenie môžeme vypočítať ako:

$$a = g \cdot \sin\alpha$$

Je dôležité si uvedomiť, že podľa toho, či rovina klesá alebo stúpa bude zrýchlenie kladné alebo záporne. Ak je rovina vodorovná, zrýchlenie bude nulové.

Teraz nám už len zostáva vyrátať, za aký čas prejde vozík danou úsečkou a akú rýchlosť bude mať na jej konci. To, ako rýchlo sa bude hýbať vieme vypočítať jednoducho z času prejdenia, keďže po celý čas bude vozík rovnomerne akcelerovať zrýchlením a.

Ako vypočítame čas, za ktorý vozík prejde celou úsečkou? Z fyziky vieme vzťah medzi počiatočnou rýchlosťou v_0, rovnomerným zrýchlením a, časom t a dráhou s:

$$s = v_0 t + \frac{1}{2} a t^2$$

V našom prípade chceme, aby vozík prešiel celou úsečkou – musí teda prejsť dráhou dĺžky d. Tiež vieme, že počiatočná rýchlosť nášho vozíka je v, dosadíme teda tieto hodnoty do vzorca:

$$d = vt + \frac{1}{2} a t^2$$

Neznáma, ktorú z tohto vzorca potrebujeme vypočítať je t. Vidíme, že vzhľadom na t je toto kvadratická rovnica. Pomocou klasického vzorca s diskriminantom dostaneme dve možné riešenia:

$$D = 2ad + v^2$$
$$t_1 = (-\sqrt{D} - v)/a$$
$$t_2 = (\sqrt{D} - v)/a$$

Prečo sú však riešenia dve? Predstavme si, že namiesto úsečky máme polpriamku a táto polpriamka stúpa (akcelerácia je teda záporná). Pokiaľ sa vozík hýbe dostatočne rýchlo, môže prejsť po priamke dĺžku d. Čas, v ktorom dosiahne tento bod na priamke je prvé riešenie našej kvadratickej rovnice.

Následne bude vozík ešte chvíľu stúpať, zastaví sa a začne zrýchľovať opačným smerom. Bodom, ktorý je od počiatočnej pozície vozíka vzdialený d prejde vozík opäť, tentoraz z opačnej strany. Čas, v ktorom sa to stane je druhé riešenie našej kvadratickej rovnice.

Môže sa dokonca stať, že niektoré riešenie tejto rovnice bude záporné, čo znamená, že vozík prešiel daným bodom "v minulosti".

Pri riešení tejto kvadratickej rovnice si treba dať pozor na viacero vecí.

Prvá je, že diskriminant môže byť záporný. Ak sa to stane, znamená to, že vozík nikdy nedosiahne koniec úsečky, a teda môžeme vypísať `NEVER`.

Druhá vecou je prípad, kedy je akcelerácia nulová. Všimnime si, že v takom prípade by sme delili nulou a teda horeuvedené vzorce nebudú použiteľné. To však vôbec nie je problém, ak je $a = 0$, tak pôvodná rovnica prejdenej vzdialenosti sa dá zjednodušiť na:

$$d = vt$$

Z čoho si vieme čas odvodiť veľmi jednoducho:

$$t = \frac{d}{v}$$

Aj tu si však musíme dať pozor na prípad kedy $v = 0$. Vtedy opäť vypíšeme `NEVER`.

Výber správneho riešenia

Keď už sme ošetrili všetky veci, na ktoré si treba dať pozor, získame dve riešenia: t_1 a t_2. Ktoré je to správne? Záporné riešenia to určite nebudú a z tých nezáporných treba vybrať to najmenšie, keďže to je čas, kedy sa vozík prvý krát dotkne druhého bodu úsečky. Pokiaľ sú obe riešenia záporné, vozík sa do tohto bodu nedostane.

Zachovanie rýchlosti

Na to, aby sme vedeli vyššie spomenuté hodnoty rátať aj pre nasledujúce úsečky ešte potrebujeme zistiť, akú rýchlosť bude vozík mať na konci aktuálnej úsečky, a o koľko spomalí (prípadne zrýchli) keď narazí na ich zlom.

Rýchlosť na konci úsečky vieme vyrátať ľahko z počiatočnej rýchlosti a rovnomerného zrýchlenia, ktoré už máme.

$$v_1 = v + at$$

Keďže medzi dvomi úsečkami je zlom, z tejto rýchlosti sa pri prechode na ďalšiu úsečku zachová len časť, konkrétne zložka zodpovedajúca smeru ďalšej úsečky. Označme si vzájomný uhol medzi oboma úsečkami ako θ. Potom rýchlosť, ktorá sa zachová vypočítame, ako:

$$v_2 = v_1 \cos\theta$$

(Ak celkom nerozumiete, prečo zachovanú rýchlosť vypočítame takto, odporúčame si to nakresliť.)

Ak dostaneme zápornú v_2, znamená to, že vozík narazil na ostrý zlom. Na takomto zlome sa nezachováva žiadna rýchlosť, preto v takomto prípade nastavíme v_2 na nulu.

Všetko spolu

Keď už vieme počítať všetky veci popísané vyššie, postupne prejdeme všetky úsečky, posčítavame všetky časy a nakoniec vypíšeme celkový čas. Ak počas prechádzania úsečkami zistíme, že niektorou z nich už vozík neprejde, tak vypíšeme `NEVER` a skončíme.