Korešpondenčný seminár z programovania
XXXII. ročník, 2014/15
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Návody k úlohám 2. 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. T2 mení okná

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

Možno ste si všimli, ale tento príklad bol zároveň aj posledný príklad v kategórii O. Vo vzorových riešeniach tejto kategórie nájdete celé riešenie tohto príkladu, tu je uvedená len časť "rekapitulácia".

Ako teda vyzerá celé riešenie pokope? Na začiatku načítame vstup a spravíme nejaké predrátania. Presnejšie, zrátame odpoveď pre prvých 63 čísiel a predpočítame si Pascalov trojuholník do hĺbky 63. Ten nám bude slúžiť na to, aby sme vedeli rýchlo povedať hodnotu jednotlivých kombinačných čísiel. Stačí nám to počítať do dvojrozmerného poľa postupne od malých hodnôt k väčším, klasickým prístupom – sčítame dve susedné z predošlého riadku.

Pokračujeme tým, že vyrátame výsledok pre horné ohraničenie h. Toto číslo si zmeníme na binárne. Potom prechádzame prvých 63 čísiel a vždy, keď vidíme, že nejaké číslo a má riešenie $k-1$ komprimácií, zistíme počet čísiel menších ako h, ktoré obsahujú a jednotiek v binárnom zápise. Toto zistíme tradičným dynamickým programovaním cez jednotlivé cifry.

Od výsledku odrátame výsledok pre horné ohraničenie $l-1$ a máme výsledok pôvodnej úlohy. Zostáva už len odhadnúť časovú a pamäťovú zložitosť. Nebudeme sa tváriť, že 63 je konštanta, lebo to vzniklo ako logaritmus čísla h a tak to aj budeme označovať. Pamätať si musíme binárny zápis čísla h a Pascalov trojuholník, ktorý je kvadratický, takže pamäťová zložitosť bude $O(\log^2 h)$. A časová zložitosť bude úplne rovnaká, keďže pre najviac $\log h$ čísiel pustíme dynamiku so zložitosťou $O(\log h)$.

2. Triedenie ponožiek

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

Písmená reťazca P postupne označíme $P[0], P[1], ..., P[n-1]$. Úsek P od ℓ-tého písmena po r-té (vrátane) budeme zapisovať $P[\ell]..P[r]$.

Pre každý prefix P, (teda úsek tvaru $P[0]..P[r]$) si spočítame charakteristické číslo, ktoré charakterizuje paritu výskytov jednotlivých písmen. Toto charakteristické číslo bude mať v binárnom zápise na i-tej pozícii od konca 0, ak i-te písmeno sa v prefixe vyskytuje párny počet krát a 1, ak nepárny. Napr. pre reťazec `ababebaa` budeme mať číslo $10010_2 = 18$. Keď má abeceda 26 písmen, charakteristické číslo je menej ako 2^{26} a zmestí sa do 32 bitovej celočíselnej premennej (napr. `int` v C++).

Charakteristické číslo pre $P[0]..P[r]$ označíme c_r. Všetky charakteristické čísla vieme spočítať v lineárnom čase od dĺžky P. Taktiež vieme spočítať charakteristické číslo Q, ktoré označíme q.

Odpoveď na úlohu, teda počet rozdelení P budeme počítať zvlášť pre všetky konce P_2. Spočítať, koľko dobrých P_2 končí r-tým písmenom P vieme nasledovne:

Nájdeme najväčšie ℓ také, že $P[\ell]..P[r]$ má aspoň toľko výskytov každého písmena ako Q. Pokiaľ takéto ℓ neexistuje, pokračujeme ďalším r. Potom odpoveď (počet P_2) je počet takých i, že $i \leq \ell$ a zároveň $P[i]..P[r]$ má rovnakú paritu počtu výskytov písmen ako Q. To je v skutočnosti to isté ako počet $i \leq \ell$, takých, že počty výskytov písmen $P[0]..P[i]$ majú rovnakú paritu, ako počty výskytov v Q mínus počty v $P[0]..P[r]$. Toto však je počet $i \leq \ell$ takých, že $c_i = q$ `xor` c_r. To je o dosť ľahší problém, nie?

Ako to však celé robiť efektívne?

Začneme s $r = 0$ a zvyšujeme r až dovtedy, kým $P[0..P[r]$ nemá z každého písmena aspoň toľko výskytov, čo Q. V takomto prípade vieme, že existuje ℓ a je aspoň 0. Nájdeme príslušné ℓ. Spočítame počet c_i rovných q `xor` c_r, takých že $i \leq \ell$. Zvýšime r o 1, nájdeme ℓ, spočítame dobré c_i, zvýšime r o 1, nájdeme ℓ, spočítame dobré c_i a tak ďalej až po koniec. Toto zaberie $O(n)$ času, plus čas strávený hľadaním ℓ-iek plus čas strávený počítaním dobrých c_i.

Všimnime si, že keď zvýšime r, tak sa ℓ nezníži. Vďaka tomuto poznatku sa dá nachádzať ℓ dosť rýchlo, rozmyslite si ako, tak aby sme pri tom dokopy obetovali najviac $O(n)$ času. Počas toho celého si budeme vo vhodnej dátovej štruktúre udržiavať pre každú hodnotu v počet c_i rovných v. Tieto počty budeme aktualizovať spolu s tým ako rastie ℓ, aby sme brali do úvahy len $i \leq \ell$. Ak je použitá dátová štruktúra vyvažovaný binárny vyhľadávací strom, tak zistiť počet c_i vieme v čase $O(\log n)$, ak je dátová štruktúra pole, tak vieme zistiť c_i v čase $O(1)$ ale potrebujeme $O(2^s)$ pamäte, kde s je počet písmen v abecede.

Potom kompletne celá časová zložitosť je buď $O(n \log n)$ s $O(n)$ pamäťovou zložitosťou alebo môžeme mať $O(n)$ čas a $O(n + 2^s)$ pamäť. Obe tieto riešenia mohli dostať plný počet bodov. n je dĺžka reťazca P, (predpokladáme, že Q je kratšie ako P, inak je odpoveď 0).

návod písal Jano

3. Tajná misia

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

Ak v meste nie sú budovy, tak je riešenie ľahké, ďalej predpokladajme, že v meste je aspoň jedna budova.

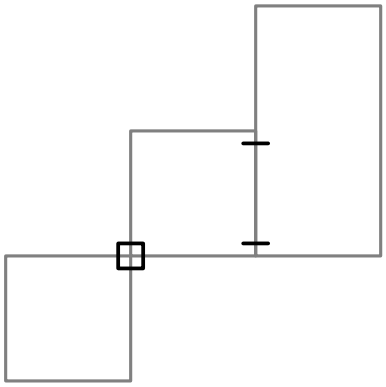
Označme si body, v ktorých sú zločinci, A, B a bod, v ktorom je James Bond, si označme C. Potom dôležitá myšlienka celého riešenia spočíva v tom, že pokiaľ má úloha riešenie (t.j. existuje miesto, kam si môže James sadnúť), tak existuje aj také riešenie, že roh nejakej budovy leží na priamke AC a roh nejakej budovy leží na priamke BC.

Pokiaľ napríklad na priamke AC nie je žiaden roh, tak vieme posunúť bod C aby sa priamka dotkla nejakého rohu, bez toho aby James prestal vidieť niektorého zločinca. Nakreslite si a uvidíte to.

Tým pádom vieme, že riešenie stačí hľadať medzi priesečníkmi priamok typu AV s priamkami typu BU, kde U a V sú nejaké rohy budov. Priamok AV je najviac 40, BV tiež, takže priesečníkov je najviac 1600. Z každého takéhoto priesečníku skontrolujeme, či by z neho James videl oboch zločincov.

Ku kompletnému riešeniu nám ostávajú už len dve veci:

- Rozmyslieť si príslušnú geometriu – ako nájsť priesečník dvoch priamok, ako zistiť či sa dve úsečky pretínajú, ako zistiť, či bod leží na priamke. Treba si tiež dávať pozor na nepresnosti pri počítaní s reálnymi číslami. Prekážky je najlepšie rozložiť na úsečky.
- Vysporiadať sa s tým, že cez niektoré rohy sa dá vidieť a cez niektoré nie, že James nemôže sedieť vo vnútri budovy a niekedy ani v rohu. Spôsobov ako to vyriešiť je mnoho, napríklad sa dajú všetky prekážky o máličko zmenšiť, aby sa dalo vidieť okolo rohov a pozdĺž stien. Ak máme problém s nepresnosťami a je vidieť cez rohy prekážok, môžeme pridať budovám maličké vnútorné výstuže. Tiež do rohov, ktoré nemajú byť priesvitné, dáme maličké úsečky, ktoré zabránia pohľadu skrz. Na obrázkoch je vidieť znázornenie výstuží a úsečiek.



Celková časová zložitosť je $O(n^3)$, kde n je počet prekážok v meste.