Korešpondenčný seminár z programovania
XXXIV. ročník, 2016/17
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Návody k úlohám 2. kola letnej 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.

návod písal Buj

1. Totálny intervaláč

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

Niekedy je riešenie úlohy jednoducho popísateľné, ale cesta k nemu nemusí byť vôbec intuitívna. Čitateľ môže nadobudnúť pri čítaní vzorového riešenia pocit, že to predsa nemôže fungovať, to nemôže byť dosť rýchle!

Toto je jedna z takých úloh. Aj preto veľkú časť návodu tvorí zdôvodnenie časovej zložitosti.

Všetky operácie okrem delenia (teda pripočítanie ku každému prvku v intervale, minimum na intervale, a súčet na intervale) zvládne obyčajný lazy-loading intervalový strom. Takisto zvláda aj operáciu maximum na intervale. Na čo by nám ale bola táto ďalšia operácia dobrá?

Pomocou nej budeme vedieť šikovne obísť delenie číslom d: začneme vo vrchole reprezentujúcom celý interval. Označme minimum na intervale m a maximum M. Ak platí $\lfloor \frac{m}{d} \rfloor - m = \lfloor \frac{M}{d} \rfloor - M$, tak ku všetkým prvkom v intervale pripočítame $\lfloor \frac{m}{d} \rfloor - m$ a končíme. V opačnom prípade zavoláme (túto) deliacu procedúru osobitne pre ľavého a pravého syna. (Je to teda akási hrubá sila s orezávaním.)

Dôkaz správnosti

Prečo si môžeme dovoliť v horeuvedenom prípade namiesto delenia pripočítavať $\lfloor \frac{m}{d} \rfloor - m$? Rozmyslite si, že ak platí predpoklad $\lfloor \frac{m}{d} \rfloor - m = \lfloor \frac{M}{d} \rfloor - M$, tak musí platiť buď $m = M$ alebo $m + 1 = M$. Takže každý prvok v intervale má hodnotu buď m, alebo M. V oboch prípadoch je rozdiel spôsobený delením rovnaký (vďaka predpokladu).

Časová zložitosť

V podstate každá operácia delenia na intervale ho rozdelí na niekoľko podintervalov, a na každom z nich spravíme operáciu pripočítania (na rôznych intervaloch pripočítavame rôzne konštanty). Ukážeme, že dokopy týchto pripočítaní nemôže byť veľa.

Zamyslime sa nad tým, kedy môže na hranici medzi prvkami i a $i+1$ končiť podinterval. Prečo nás to zaujíma? Ak chceme vypočítať počet pripočítaní, ktorými daná operácia delenia prispela, tak je to počet podintervalov, a ten je rovný 1+ počet hraníc podintervalov (nerátajúc začiatok celého intervalu ani jeho koniec).

Ak bol pred delením rozdiel prvkov i a $i+1$ v absolútnej hodnote rovný s, tak po delení bude nanajvýš $\lceil \frac{s}{2} \rceil$. Podinterval nám tam končí práve vtedy, keď sa tento rozdiel zmenšil.

Koľkokrát sa tento rozdiel môže takto zmenšiť? Ak neberieme do úvahy, že sa tento rozdiel môže zmeniť aj vtedy, keď dostaneme pripočítanie/delenie ovplyvňujúce iba jeden z prvkov $i, i+1$, tak sa môže zmenšiť najviac $O(\log C)$-krát, kde $C = 10^9 + q \cdot 10^4$. (Na začiatku je rozdiel rádovo nanajvýš C. Pretože sa pri každom delení rádovo spoloviční, môžeme ich mať len $O(\log C)$. A nakoniec sa ešte raz môže zmenšiť z 1 na 0, ale to je len konštanta.)

Zoberme teraz do úvahy zmeny spôsobené operáciami, ktoré ovplyvňujú iba jeden z prvkov $i, i+1$. V najhoršom prípade nám rozdiel nastavia na nejaké číslo rádovo C, a opäť ho budeme môcť znížiť $O(\log C)$-krát. Každá takáto operácia nám týmto spôsobom ovplyvní iba 2 rozdiely – na začiatku a na konci intervalu.

Môžeme sa na to teda pozerať takto: na začiatku máme pole čísel dĺžky $n-1$. Číslo na pozícii i hovorí, koľkokrát ešte môžeme zmenšiť rozdiel prvkov i a $i+1$. Na začiatku je pole vyplnené nejakými číslami od 0 po k (kde k je rádovo $\log C$). Ďalej budeme mať rádovo q krokov, a v každom kroku môžeme spraviť jedno z nasledovných:

- Nastavíme niektoré číslo na nejakú hodnotu od 0 po k. (Toto reprezentuje zmenu rozdielu na okraji intervalu, v ktorom robíme operáciu.)
- Zoberieme si súvislý úsek, každé číslo v ňom väčšie ako 1 znížime o 1, a každé číslo rovné 1 môžeme znížiť na 0. K počítadlu pripočítame 1, a ďalších 1 za každé číslo v úseku, ktoré sme znížili. (Reprezentuje delenie na intervale.)

Na konci máme v počítadle počet pripočítaní, ktorými prispeli operácie delenia. Aká najväčšia hodnota teda môže byť v počítadle? Ak by sme krok prvého typu nemali povolený, tak by bola odpoveď rádovo $nk + q$. Za každý krok prvého typu sa nám toto obmedzenie zvýši najviac o k. Takže najväčšia možná hodnota počítadla je rádovo $nk + qk + q$, čo je $O((n + q) \log C)$.

Každé pripočítanie má časovú zložitosť $O(\log n)$, a "normálnych" pripočítaní máme $O(q)$. Dokopy tak dostávame časovú zložitosť $O((n + q) \log C \log n)$.

návod písal Buj

2. Teleportuj sa a zarábaj!

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

Krátka reklama pre skúsených: Táto úloha vám dá úplne nový pohľad na štandardný algoritmus umocňovania matice. Zistíte, že sa dajú umocňovať matice s operáciami inými ako + a ·.

Ako to súvisí s úlohou? Čítajte.

Pomalé riešenie

Predstavme si, že už máme tabuľku najhodnotnejších sledov dĺžky k. Pomocou nasledovnej úvahy z nej získame tabuľku najhodnotnejších sledov dĺžky $k+1$: každý sled dĺžky $k+1$ začínajúci v A a končiaci v B vieme získať ako kombináciu nejakého sledu dĺžky k začínajúceho v A a končiaceho v nejakom vrchole C, a hrany z C do B.

Takže by sa nám stačilo pozerať na všetky takéto kombinácie, a spomedzi nich zobrať tú najhodnotnejšiu. Týchto kombinácii je veľa, avšak si uvedomme, že keď hľadáme najhodnotnejší sled dĺžky $k + 1$ (z A do B), stačí sa pozerať iba na najhodnotnejšie sledy dĺžky k (začínajúce v A). A o týchto máme informáciu v aktuálnej tabuľke. Postupne teda vyskúšame každý vrchol C ako predposledný vrchol sledu.

Dostávame tak algoritmus s časovou zložitosťou $O(n^3 \cdot k)$.

Mocnenie

Sled dĺžky k nemusíme zostrojovať ako sled dĺžky $k-1$ plus sled dĺžky 1. Môžeme ho zostrojiť ako sled dĺžky $k - a$ plus sled dĺžky a pre ľubovoľné a.

Takže ak $k = 2 \cdot k'$, stačí nám rekurzívne vypočítať tabuľku najhodnotnejších sledov dĺžky k', a tú následne skombinovať samú so sebou. Ak $k = 2 \cdot k' + 1$, tak postupujeme tak, že vypočítame tabuľku najhodnotnejších sledov dĺžky $k - 1$, a tú skombinujeme s tabuľkou hrán (ako pri pomalom riešení).

Výsledný algoritmus má rovnakú časovú zložitosť, ako umocňovanie matíc, teda $O(n^3 \cdot \log k)$. V podstate totiž umocňujeme maticu s operáciami $min, +$. Záujemcovia sa môžu zamyslieť nad všeobecným kritériom, ktoré by mali spĺňať dve operácie $\oplus, \otimes$ na to, aby pre ne "fungovalo" násobenie matíc.

návod písal Hodobox a Buj

3. Turistika na Orave

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

Niečo o probléme najdlhšej cesty

Tak čo to teda tá úloha po nás vlastne chce? Máme Oravu, ktorá je orientovaným grafom. Máme v ňom nájsť najdlhšiu cestu. Hľadanie najdlhšej cesty v grafe znie ako celkom štandardný problém. Tak ak nevieme, pogooglime, pohľadáme, a čo nezistíme. Pre všeobecný graf je to NP-ťažký problém, teda ho deterministicky vieme riešiť len exponenciálnym algoritmom. Na druhú stranu, ak vieme že graf je strom alebo acyklický orientovaný graf, vieme ho vyriešiť v lineárnom čase od počtu vrcholov a hrán.

Tesne! Orava je totiž orientovaná, no nie acyklická. No a čo to hovorí tá špeciálna záruka v zadaní?

"Nech by ste svoju prechádzku začali pri ktorejkoľvek prírodnej kráse, ak ju na tom istom mieste chcete aj skončiť, je zaručené, že viete navštíviť nanajvýš štyri rôzne iné prírodné krásy – a to bez ohľadu na to, ako budete chodiť a či niektoré prírodné krásy (vrátane tej, kde ste začali a skončili) navštívite viackrát."

Inak povedané, na Orave neexistuje silno súvislý komponent (odteraz SSK) ktorý by mal viac ako päť vrcholov. Od Oravy a acyklickým orientovaným grafom, na ktorom by sme úlohu vedeli vyriešiť jednoducho, nie je veľmi ďaleko. Musíme jej len trošku pomôcť.

Riešenie pre acyklické grafy

Na obyčajnom orientovanom acyklickom grafe vieme nájsť najdlhšiu cestu nasledovne: zoraďme si vrcholy topologicky, prechádzajme vrcholy v tomto poradí (začínajúc od vrcholov, do ktorých nevedie žiadna hrana) a počítajme pre každý z nich dĺžku najdlhšej cesty končiacej v ňom. Najdlhšia cesta končiaca v A je najdlhšia cesta do niektorého vrcholu B, od ktorého vedie k A hrana, plus hrana z B do A.

Takže keď chceme zistiť dĺžku najdlhšej cesty končiacej v A, iba sa pozrieme na všetky vrcholy, z ktorých do A vedie hrana, a vyberieme ten, ktorý má svoju "dĺžku" najväčšiu (jeho dĺžku už poznáme, lebo vrcholy prechádzame podľa topologického usporiadania). Toto číslo zväčšíme o 1 (zarátame poslednú hranu BA), a toto je hľadaná dĺžka pre A.

Kde je problém so silno súvislými komponentmi?

SSK nám robia problémy preto, že nám umožňujú chodiť "donekonečna", pričom navštívime niektoré vrcholy viackrát. To nechceme, lebo to nesúhlasí s definíciou cesty – v ceste sa na rozdiel od sledu vrcholy neopakujú. (V acyklických grafoch tento problém nie je, lebo v nich je najdlhší sled zároveň aj najdlhšou cestou.)

Chceli by sme preto každý SSK nahradiť niečím tak, aby dĺžka najdlhšej cesty zostala rovnaká, ale aby sme dostali acyklický graf. Na tomto grafe by sme potom mohli spustiť algoritmus z úvodu.

Takže náš algoritmus bude mať nasledovnú formu: nájdi všetky SSK (napr. pomocou Tarjanovho algoritmu na hľadania SSK), zostrojíme nový graf, a na ňom pustíme algoritmus z úvodu. Ako ale upraviť SSK tak, aby sme dostali acyklický graf a informácia o najdlhšej ceste zostala zachovaná?

Transformácia

Uvedomme si: každú cestu v grafe vieme rozdeliť na niekoľko častí takých, že každá časť leží celá v jednom SSK. Časť môže byť aj prázdna (0 hrán) a je to vlastne cesta medzi dvoma (nie nutne rôznymi) vrcholmi z toho istého SSK. Medzi týmito časťami sa presúvame hranami, ktoré vedú medzi rôznymi SSK.

Časti vieme "odsimulovať" nasledovne. Časť cesty vlastne znamená, že sme vošli do nejakého SSK (alebo sme v ňom začali), a potom z neho vyjdeme (alebo v ňom skončíme). Každý vrchol SSK teda zdvojíme – budeme mať *vstupné* vrcholy, a *výstupné* vrcholy. Každá hrana medzi týmito vrcholmi bude reprezentovať cestu v pôvodnom SSK, a bude môcť vychádzať iba zo vstupných vrcholov, a vchádzať len do výstupných. Dĺžka hrany bude rovná dĺžke cesty, ktorú reprezentuje. (Napríklad ak sme mali v pôvodnom SSK vrcholy A, B, C a hrany AB, BC, CA, tak v novom grafe budeme mať vrcholy $A_{in}, A_{out}, B_{in}, B_{out}, C_{in}, C_{out}$, cesta ABC by bola reprezentovaná hranou $A_{in}C_{out}$ s dĺžkou 2, a cesta CA by bola reprezentovaná hranou $C_{in}A_{out}$ s dĺžkou 1. Takisto netreba zabudnúť na prázdne cesty, ktoré sú reprezentované hranami dĺžky 0.)

Kedže nás zaujíma najdlhšia cesta, tak medzi dvoma vrcholmi A_{in}, B_{out} pridáme iba jednu hranu, a to tú, ktorá reprezentuje najdlhšiu cestu z A do B. Tú si môžeme dovoliť nájsť exponenciálnym algoritmom, nakoľko SSK obsahujúci A a B má nanajvýš 5 vrcholov – takže to bude len konštantne veľa operácii. Toto spravíme pre každú dvojicu (nie nutne rôznych) vrcholov z toho istého SSK, čo je nanajvýš $5 \cdot 5 = 25$ dvojíc pre každý SSK – opäť konštanta. A počet SSK je $O(n)$, čiže si to celé môžeme dovoliť.

Nakoniec, prechody medzi rôznymi SSK zariadime tak, že pre každú hranu AB v pôvodnom grafe, kde A a B sú v rôznych SSK, pridáme hranu $A_{out}B_{in}$.

Výsledný algoritmus má časovú zložitosť $O(n + m)$, teda úmernú veľkosti pôvodného grafu. Nový graf je totiž len konštantne-veľakrát väčší od pôvodného.