1 PEA-NIM :: zadanie

PEA-NIM je hra dvoch hráčov. Na začiatku hry sa na stole spraví n kôpok na ktorých je postupne $a_1, \ldots, a_n$ kamienkov a zvolí sa číslo $k \leq n$. Hráč na ťahu si vždy vyberie aspoň jednu a nanajvýš k kôpok a z každej vybratej kôpky odstráni ľubovoľný kladný (nie nulový) počet kamienkov. Hráč, ktorý už nevie spraviť platný ťah (keďže všetky kôpky sú prázdne) prehráva.
Nájdite čo najefektívnejší algoritmus, ktorý pre dané n, k a $a_1, \ldots, a_n$ zistí, či je dotyčná pozícia vyhrávajúca pre hráča na ťahu. (Hint: Chcete vhodne zovšeobecniť riešenie pre $k = 1$.)

PEA-NIM :: riešenie

Ako pre $k = 1$ si zapíšme veľkosti kôpok v dvojkovej sústave: každé a_i si vyjadríme ako $\sum b_{i,j} \cdot 2^j$. Na hodnoty $b_{i,j}$ sa môžeme dívať ako na 2D tabuľku bitov.
Tentoraz vieme meniť až k hodnôt a_i, teda k riadkov tabuľky, naraz. Ak teda napríklad máme v nejakom stĺpci samé 1, vieme až k z nich zmeniť na 0. A naopak. Čo teda tesne nevieme je spraviť zmenu počtu jednotiek v stĺpci o $k + 1$. To nás môže priviesť k nasledovnej hypotéze:
Povedzme, že pozícia je červená, ak je v každom stĺpci našej tabuľky počet jedničiek deliteľný $k + 1$. (Formálne, pre každé j platí, že $k + 1$ delí $\sum_i b_{i,j}$.) Ostatné pozície budú modré. Tvrdíme, že práve všetky červené pozície sú prehrávajúce.
Dôkaz je analogický s prípadom $k = 1$, len technickejší. Je zjavné, že každý ťah z červenej pozície vedie nutne do modrej. To, že z každej modrej pozície **existuje** ťah do červenej, dokážeme konštruktívne, podobne ako pre $k = 1$: ideme po stĺpcoch zľava doprava a postupne si vyberáme riadky, ktoré zmenšíme. (Prvá zmena v každom riadku musí byť z 1 na 0.)

Hra sa volá Moorov NIM, pán E. H. Moore publikoval jej riešenie už v roku v 1910.

---

2 Logika :: zadanie

Daný je výrok v tvare $(Q_1 x_1)(Q_2 x_2) \ldots (Q_n x_n)\varphi(x_1, x_2, \ldots, x_n)$, kde každé Q_i je kvantifikátor (buď $\forall$ alebo $\exists$, v ľubovoľnom poradí) a kde $\varphi(\ldots)$ je 2-CNF-SAT formula. Nájdite efektívny algoritmus ktorý zistí, či je tento výrok pravdivý.

Logika :: riešenie

Toto je o čosi všeobecnejšia verzia úlohy riešenej na prednáške. Kompletné riešenie je napr. v článku Aspvall, Plass, Tarjan: A linear-time algorithm for testing the truth of certain quantified boolean formulas.

Začneme tým, že si všetky klauzuly zapíšeme ako implikácie a zaznačíme ako orientované hrany v grafe. Nájdeme silno súvislé komponenty, čím zistíme, ktoré dvojice literálov sú ekvivalentné. Ako v 2-SATe, aj tu platí, že ak v nejakom komponente máme aj premennú aj jej negáciu, odpoveď je nie. Navyše však pribudne niekoľko nových typov problémov:

- Žiaden všeobecne kvantifikovaný vrchol nesmie byť dosiahnuteľný z iného všeobecne kvantifikovaného vrcholu. Ak takéto niečo nastane, výrok určite nie je pravdivý: nepriateľ vie tieto dva vrcholy ohodnotiť tak, aby vyrobil spor.
- Navyše ešte nesmie nastať situácia že v nejakom silno súvislom komponente máme aj existenčne kvantifikovaný vrchol (x_i alebo $\neg x_i$), aj všeobecne kvantifikovaný vrchol (x_j alebo $\neg x_j$), pričom všeobecne kvantifikovaný vrchol má väčšie číslo (teda $i < j$). Totiž nech by sme akokoľvek zvolili hodnotu x_i, jedna z dvoch možných hodnôt x_j vyrobí spor.

Obe tieto podmienky vieme vhodným algoritmom kontrolovať v lineárnom čase.

Pridám ešte poznámku, že na zisťovanie, či je náš výrok pravdivý, sa oplatí pozerať ako na hru. Ideme cez premenné zľava doprava, pričom my volíme hodnoty existenčne kvantifikovaných premenných a náš nepriateľ volí tie všeobecne kvantifikované. Hru vyhráme vtedy, ak po zvolení všetkých hodnôt vznikne z φ pravdivý výrok. No a pôvodný výrok je pravdivý práve vtedy, ak v tejto hre máme vyhrávajúcu stratégiu.

---

3 Kombinatorika :: zadanie

Označme $S(n)$ počet spôsobov ako zapísať n ako súčet práve troch prvočísiel, pričom záleží na poradí a navyše požadujeme, aby aspoň jedno z prvočísiel malo v sebe podreťazec 47. Napr. $S(51) = 3$ lebo $51 = 2 + 2 + 47 = 2 + 47 + 2 = 47 + 2 + 2$. Nájdite $\sum_{n=1}^{1\,000\,000} (S(n))^2$. Áno, programom. Odovzdajte aj ten (a jeho stručný popis), nie len číslo.

Kombinatorika :: riešenie

Pre $i = 1 \dots 10^6$ položme $p_i = 1$ ak i je prvočíslo a $q_i = 1$ ak i je prvočíslo **neobsahujúce** podreťazec 47. Ostatné p_i a q_i sú nulové. (Hodnoty p_i nájdeme napr. Eratostenovym sitom, hodnoty q_i z nich priamočiarou kontrolou.)
Uvažujme teraz polynóm $p(x) \sum_i p_i x^i$. Zjavne koeficient pri x^n v polynóme $p^3(x)$ predstavuje počet trojíc prvočísiel ktoré majú súčet n, čiže takmer presne nami hľadanú hodnotu $S(n)$. Rozdiel je už len v tom, že v našom koeficiente p^3 sú zarátané aj tie trojice, v ktorých žiadne prvočíslo neobsahuje 47.
Koľko je týchto zlých trojíc? To nám povie koeficient pri x^n v polynóme $q^3(x)$ definovanom analogicky hodnotami q_i.
Stačí teda pomocou DFT/NTT spočítať polynómy p^3 a q^3. V poslednom kroku (pri zisťovaní skutočnej hodnoty hľadanej sumy) sa oplatilo buď použiť jazyk, ktorý vie veľké čísla, alebo spočítať výsledok modulo dve rôzne prvočísla a presnú hodnotu dostať napr. z Wolfram Alpha alebo ručne pomocou čínskej zvyškovej vety.

---

4 Kombinatorika pre hackerov :: zadanie

Python má v sebe efektívnu implementáciu násobenia veľkých čísel. Popíšte, ako pomocou nej vyriešiť predchádzajúcu úlohu – bez toho aby ste museli (okrem samotného nájdenia prvočísiel) implementovať akýkoľvek pokročilý algoritmus. Stačí sa sústrediť na popis hlavnej myšlienky, nemusíte vyrobiť kompletný funkčný program.

(Kvôli horším konštantám neodporúčam použiť tento program na riešenie predchádzajúcej úlohy. Your call.)

Kombinatorika pre hackerov :: riešenie

Intuitívne si riešenie tejto úlohy môžete predstaviť nasledovne: Namiesto polynómu x^2+3x+2 budeme mať číslo 1 00003 00002. Keď teraz toto číslo napr. umocníme na druhú, dostaneme 1 00006 00013 00012 00004. Vidíte analógiu s $(x^2 + 3x + 2)^2$?

Teraz poriadnejšie.
Rozmyslime si, že každá z hodnôt $S(i)$, ktoré nás zaujímajú, je pomerne malá. Prvočísel do 10^6 je menej ako 10^5, pre každé konkrétne $n \leq 10^6$ je teda menej ako 10^{10} trojíc prvočísiel so súčtom n. (Keď zvolíme prvé dve prvočísla, tretie číslo je jednoznačne určené.) Tento odhad je veľmi veľmi voľný, ale poslúži.
Namiesto násobenia polynómov si teraz môžeme definovať čísla $P = \sum_i p_i \cdot 2^{40i}$ a $Q = \sum_i q_i \cdot 2^{40i}$. Keď teraz spočítame napríklad hodnotu P^3, tak jej bity $40n$ až $40n+39$ budú obsahovať tú istú hodnotu, ktorú sme v úlohe 3 dostali ako koeficient pri x^n v p^3. (Prečo? Lebo $2^{40} > 10^{10}$, a teda nenastane pretečenie.) Analogicky spracujeme Q a sme hotoví.

Inými slovami, namiesto toho, aby sme počítali $p^3(x)$, použili sme existujúcu implementáciu násobenia na vypočítanie hodnoty $p^3(2^{40})$, no a z tej vieme jednoznačne určiť koeficienty $p^3(x)$.

---

5 Ježibaba :: zadanie

Ježibaba pozná niekoľko receptov. Všetky recepty sú nasledovného typu: „ak zoberieš elixír mladosti a povaríš ho s babím uchom, dostaneš elixír pravdovravnosti“. Existuje e rôznych elixírov a s rôznych surovín. Každá surovina má svoju cenu.
Ježibaba by chcela lexikón, pomocou ktorého bude vedieť *v asymptoticky optimálnom čase* odpovedať na otázky nasledovného typu: „mám elixír pomáhajúci proti otlakom na nohách, chcem z neho spraviť elixír proti smradľavému dychu, akým postupom je to najlacnejšie?“
Popíšte, ako taký lexikón *čo najefektívnejšie* vyrobiť a ako podľa nej pre ľubovoľné dva elixíry zostrojiť (jeden ľubovoľný) optimálny postup v optimálnom čase (alebo oznámiť, že daná úloha nemá riešenie).

(Nikde nie je povedané, že by lexikón mal byť tvorený e^2 disjunktnými návodmi. Za ľubovoľné riešenie, ktorého čas predspracovania od e závisí lepšie ako kubicky, bude veľa bodov.)

Ježibaba :: riešenie

Toto je úloha o najkratších cestách v grafe.
Ak je receptov málo, ježibabe sa najviac oplatí e-krát spustiť Dijkstru: raz pre každý elixír ako cieľ. V rámci Dijkstrovho algoritmu pôjdeme z cieľa proti smeru hrán, spočítame si do každého vrcholu najkratšiu cestu, a taktiež si pre každý vrchol zapamätáme hranu, ktorou bolo dotyčné minimum dosiahnuté – čiže hranu hovoriacu, kadiaľ sa najlepšie ide do cieľa.
Ak je receptov veľa, lepším riešením je upraviť all-pairs shortest paths algoritmus z prednášky. Tam sme si ukazovali, ako prerobiť Floydov-Warshallov algoritmus z času $\Theta(n^3)$ na lepšiu verziu bežiacu v čase násobenia matíc.
To, čo teraz potrebujeme dorobiť, je pamätanie si, kadiaľ ísť. Intuitívne by ste si zrejme chceli pamätať pre každú dvojicu elixírov e_1, e_2 **prvý** vrchol na optimálnej ceste – teda odpoveď na otázku „ak mám e_1 a chcem e_2, na aký elixír mám prerobiť e_1?“ Toto síce ide, ale implementácia má zbytočne veľa špeciálnych prípadov.
Omnoho lepším riešením je pamätať si **nejaký ľubovoľný** vrchol na optimálnej ceste. Rozmyslite si, že aj pomocou takejto informácie vieme ľubovoľnú cestu zrekonštruovať v čase priamo úmernom jej dĺžke.
No a kde túto informáciu vezmeme? Pri klasickom Floydovom-Warshallovom algoritme stačí robiť nasledovné: vždy, keď nejakú hodnotu $D[i, j]$ zlepším pomocou $D[i, k] + D[k, j]$, zapamätám si, že z i do j sa ide cez k. Analógiu tejto myšlienky vieme dorobiť aj do našej efektívnejšej implementácie.

---

6 Lo mem knapsack :: zadanie

Máme inštanciu klasického 0-1 knapsacku, teda n predmetov, každý s nejakou váhou w_i a cenou c_i, a máme maximálnu celkovú váhu w_{max}. Všetky w_i aj w_{max} sú kladné celé čísla neprevyšujúce 10^6.
Chceli by sme nájsť jedno konkrétne optimálne riešenie (konkrétnu podmnožinu predmetov, nie len súčet ich cien). Máme však k dispozícii len $O(nw_{max})$ času a $O(w_{max})$ pamäte. Nájdite a dostatočne detailne popíšte algoritmus ktorý úlohu s týmito obmedzeniami vyrieši, alebo dokážte, že takýto algoritmus nemôže existovať.

Lo mem knapsack :: riešenie

Štandardná knapsacková rekurencia vyzerá nasledovne: Označme si $B[i, j]$ najlepšiu cenu riešenia, ak spomedzi prvých i predmetov smieme vybrať ľubovoľnú podmnožinu s váhou nanajvýš j. Vo všeobecnosti sú dve možnosti: buď $B[i, j] = B[i-1, j]$ (teda i-tu vec nevyberieme), alebo $B[i, j] = c_i + B[i - 1, j - w_i]$ (teda ju vyberieme). Prvú možnosť máme na výber vždy, druhú len ak $j \geq w_i$. My si samozrejme vždy, keď máme na výber, vyberieme tú možnosť, ktorá nám dá väčšiu cenu.
Cena optimálneho riešenia zadanej inštancie je hodnota $B[n, w_{max}]$. Pomocou vyššie uvedenej rekurencie vieme túto hodnotu vypočítať v čase a pamäti $\Theta(nw_{max})$.
Ak si všimneme, že $B[i, j]$ sa odvoláva len na nejaké hodnoty $B[i - 1, ?]$, vieme si pamätať len dva riadky a pamäť tak zredukovať na $\Theta(w_{max})$. Z takto upraveného riešenia síce nevieme priamo zostrojiť konkrétny optimálny výber predmetov, tu je ale ľahká pomoc – ako posledný myšlienkový krok teraz použijeme Hirschbergov trik (na prednáške ukazovaný pre LCS).

---

7 Domček :: zadanie

Predstavte si „domček“ – taký ten klasický obrázok, ktorý sa kreslí jedným ťahom. Na domček sa môžeme dívať ako na rovinný graf so 6 vrcholmi. (Jeden z vrcholov je priesečník uhlopriečok štvorca.) Uvažujme všetky sledy dĺžky n, ktoré začínajú na vrchu strechy. Chceme vedieť, koľko ich je.
Ako by ste napísali čo najefektívnejší program, ktorý to pre dané n exaktne spočíta?
Asymptoticky koľko je tých sledov v závislosti od n?

Domček :: riešenie

Nech A je matica susednosti domčeka. V matici A^n na súradniciach $[i, j]$ zjavne dostávame počet sledov dĺžky n, ktoré vedú z vrcholu i do vrcholu j. Na exaktné riešenie našej úlohy teda stačí efektívne spočítať A^n a následne sčítať všetky hodnoty na súradniciach $[0, j]$ (kde 0 je číslo vrcholu strechy).
Maticu umocniť na n-tú vieme pomocou $\Theta(\log n)$ násobení matíc (exponentiation by squaring). Keďže ide o maticu 6×6, netreba riešiť blbosti, samotné násobenie robíme kubickým algoritmom. Čo však treba riešiť? Koeficienty v matici A^n rastú exponenciálne s n-kom, násobíme a sčitujeme teda veľké čísla. Násobenie týchto veľkých čísiel treba robiť efektívne. Ak na to použijeme nejakú variáciu FFT, dostávame výslednú časovú zložitosť celého algoritmu $\Theta(n \log n)$.

Najväčšie vlastné číslo matice A má hodnotu $\alpha \approx 3.468$. Zodpovedajúci vlastný vektor nie je kolmý na $(1, 0, 0, 0, 0, 0)$. Počet sledov dĺžky n vieme zapísať ako $(1, 0, 0, 0, 0, 0) \cdot A^n$. Z vyššie uvedených skutočností vyplýva, že počet sledov rastie rovnako rýchlo ako α^n.