Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Adrian Beker, Daniel Paleka, Ivan Paljak, Stjepan Požgaj, Tonko Sabolčec i Paula Vidas. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima.

Zadatak Autoritet

Pripremili: Adrian Beker i Paula Vidas
Potrebno znanje: teorija grafova, DFS-stablo, artikulacijske točke i dvostruko povezane komponente

Prvo, očito su potezi komutativni i involutorni pa nas samo zanima skup čvorova koje smo odabrali. Broj najkraćih nizova poteza tada jednostavno možemo dobiti množenjem broja najmanjih skupova s $k!$, gdje je k veličina najmanjeg skupa. Osnovna je ideja da ćemo u većini slučaja moći postići cilj koristeći malo poteza. Za graf ćemo reći da je *klika* ako je potpun i ima bar dva čvora. Razlikujemo četiri (disjunktna) slučaja:

1. Graf je povezan
2. Graf se sastoji od točno dviju disjunktnih klika
3. Graf se sastoji od barem triju disjunktnih klika
4. Preostali slučaj

Prvi je slucaj očit. U drugom slučaju, nije teško vidjeti da je optimalno uzeti čitavu manju kliku (koji god podskup čvorova odabrali, graf se raspada na najviše dva potpuna grafa). U trećem slučaju trebamo bar 2 poteza i to se postiže uzimanjem bilo kojih dvaju čvorova iz različitih klika (ili dvaju iz iste ukoliko je ona veličine 2).

U četvrtom slučaju graf nije povezan i postoji komponenta K koja nije klika. Tada očito trebamo bar 1 potez, a tvrdimo da to možemo i postići. Recimo da je čvor x *dobar* ako primjenom operacije na x dobivamo povezan graf. Recimo da se uklanjanjem čvora x njegova komponenta raspada na komponente $C_1, C_2, \ldots, C_k$. Tada x nije dobar ako i samo ako je povezan sa svim cvorovima neke C_i.

Ako je K izolirani čvor, tada je taj čvor dobar, stoga pretpostavimo da K nije potpun graf. Recimo da je čvor u K *centralan* ako je povezan sa svima ostalima.

Tvrdnja 1. Postoji čvor y u K koji nije centralan i nije artikulacijska točka.

Dokaz: Ako ne postoji centralan čvor, onda odaberemo čvor u K koji nije artikulacijska točka, npr. list u razapinjućem stablu. Ako postoji centralan čvor, uzmemo bilo koji čvor koji nije centralan (takav postoji jer K prema pretpostavci nije potpun graf). □

Sada je očito je da je čvor y iz tvrdnje dobar. Konačno, preostaje prebrojati dobre čvorove, a to se moze učiniti na sljedeći način. Primijetimo da neki neizolirani čvor nije dobar ako i samo ako je ili centralan u čitavoj komponenti ili istodobno jedinstvena artikulacijska točka i centralan u nekoj dvostruko povezanoj komponenti u kojoj se nalazi. Stoga problem možemo riješiti nalaženjem artikulacijskih točaka i rastavom grafa na dvostruko povezane komponente. Postoje i druga rješenja koja koriste standardne informacije iz DFS-stabla. Ukupna je složenost $\mathcal{O}(N+M)$ ili $\mathcal{O}(N+M\log N)$. Za implementacijske detalje pogledajte službene kodove.

Zadatak Restoran

Pripremili: Adrian Beker, Ivan Paljak, Tonko Sabolčec
Potrebno znanje: dokazivanje točnosti pohlepnog algoritma, tournament stablo

Za početak, promatrajmo jednostavniji slučaj u kojemu se skup gostiju koji čekaju pred restoranom ne mijenja. Najprije uočimo da postoji optimalno rješenje u kojemu trenutci kuhanja i jedenja svakog gosta čine interval. Zaista, primjerice ako i-ti gost završava kuhanje u minuti t, tada on može kuhati u vremenskom intervalu $[t - a_i + 1, t]$, a trenutke kuhanja ostalih gostiju možemo po potrebi pomaknuti ranije, čime se ukupno vrijeme ne povećava (analogno se pokazuje za trenutke jedenja). Ispostavlja se da vrijedi i nešto jača tvrdnja:

Tvrdnja 1. Postoji optimalno rješenje u kojemu gosti jedu hranu istim redoslijedom kojim su je i kuhali.

Dokaz: Neka gost y dolazi neposredno poslije gosta x u redoslijedu jedenja. Ukoliko y dolazi prije x u redoslijedu kuhanja, moguće ih je zamijeniti u poretku jedenja tako da i dalje svi uvjeti budu zadovoljeni, a ukupno se vrijeme ne poveća. Tvrdnja slijedi jer se opisanom transformacijom smanjuje broj inverzija u redoslijedu jedenja u odnosu na redoslijed kuhanja. □

Primijetimo dalje da ako fiksiramo redoslijed kuhanja (a time prema Tvrdnji 1 i redoslijed jedenja), tada je optimalno pohlepno dodijeliti najprije intervale kuhanja, a zatim intervale jedenja. To nam omogućava da, za fiksan redoslijed gostiju, ukupno vrijeme evaluiramo u linearnoj složenosti. Sada je lako riješiti prvi podzadatak u složenosti $\mathcal{O}(N! \cdot N)$ – naprosto ispitamo sve moguće poretke.

Međutim, već za drugi podzadatak potrebna je sljedeća opservacija, koja se pokazuje bitnom i za ostatak rješenja:

Tvrdnja 3. Ako je $p_1, p_2, \ldots, p_N$ poredak gostiju, gdje je p_i oznaka i-tog gosta u poretku, tada je najmanje ukupno vrijeme jednako

$$\max_{i=1}^{N} \left\{ \sum_{j=1}^{i} a_{p_j} + \sum_{j=i}^{N} b_{p_j} \right\}.$$

Dokaz: Označimo s T vrijednost gornjeg izraza. Tada ukupno vrijeme očito nije manje od T. Kako bismo dokazali da ono nije veće od T, uočimo posljednjeg gosta koji počinje jesti čim završi s kuhanjem (primijetimo da će to uvijek biti slučaj za prvog gosta, stoga takav gost uvijek postoji), neka je to k-ti gost u poretku. Budući da je k maksimalan s tim svojstvom, za sve $k < i \leq N$ mora vrijediti da i-ti gost počinje jesti čim $(i-1)$-vi gost završi. Stoga je ukupno vrijeme jednako $\sum_{j=1}^{k} a_{p_j} + \sum_{j=k}^{N} b_{p_j}$, što očito ne prelazi T. □

Primijetimo da se izraz za vrijednost T iz Tvrdnje 2 može malo drugačije zapisati:

$$\sum_{j=1}^{N} b_j + \max_{i=1}^{N} \left\{ a_{p_i} + \sum_{j=1}^{i-1} (a_{p_j} - b_{p_j}) \right\},$$

Zbog toga se problem svodi na nalaženje poretka $p_1, p_2, \ldots, p_N$ koji minimizira maksimum od $c_i = a_{p_i} + \sum_{j=1}^{i-1} (a_{p_j} - b_{p_j})$ po svim $1 \leq i \leq N$. Sada je drugi podzadatak moguće riješiti dinamičkim programiranjem s bitmaskama – za svaki podskup gostiju, dinamika pamti rješenje spomenutog problema. Prijelaze između stanja nije teško izvesti u linearnoj složenosti na način da fiksiramo svakog mogućeg posljednjeg gosta u poretku. Stoga je ukupna složenost ovog pristupa $\mathcal{O}(N \cdot 2^N)$.

Konačno dolazimo i do trećeg podzadatka, za koji je potrebna sljedeća ključna opservacija o optimalnom poretku gostiju:

Tvrdnja 2. Za gosta ćemo reći da je *kuhar* ako mu je vrijeme kuhanja strogo manje od vremena

jedenja, a u suprotnom ćemo reći da je *gurman*. Tada postoji optimalan poredak u kojemu prvo dolaze svi kuhari sortirani uzlazno prema vremenu kuhanja te potom svi gurmani sortirani silazno prema vremenu jedenja.

Dokaz: Dokaz se temelji na standardnom *exchange argumentu*. Promotrimo neki optimalan poredak $p_1, p_2, \ldots, p_N$ te pogledajmo što se događa kada zamijenimo neka dva susjedna gosta p_i, p_{i+1}. Možemo primijetiti da se vrijednosti c_j za $j \notin \{i, i+1\}$ ne mijenjaju, stoga je samo potrebno pratiti vrijednosti c_i te c_{i+1}. Nije teško uvjeriti se da je uvijek optimalno napraviti zamjenu u skladu s opisanim načinom sortiranja. Detalje ostavljamo čitateljici za vježbu. □

Tvrdnja 3 sada nam omogućava da treći podzadatak riješimo sortiranjem svih gostiju na odgovarajući način te jedostavnom evaluacijom ukupnog vremena, u ukupnoj složenosti $\mathcal{O}(N \log N)$.

Na kraju, za sve bodove potrebno je osmisliti kako efikasno održavati trenutni skup gostiju te evaluirati formulu za ukupno vrijeme. U tu svrhu uzmimo sve goste koji se ikad pojavljuju te ih sortirajmo na način kako je opisano u Tvrdnji 3. Time dobivamo niz parova (a_i, b_i) nad kojim je potrebno podržavati sljedeće upite:

- Označi neki par aktivnim/neaktivnim.
- Evaluiraj formulu za parove koji su trenutno aktivni.

Primijetimo da ako parove koji trenutno nisu aktivni zamijenimo s $(0, 0)$, tada rješenje dobivamo evaluacijom formule za čitav niz parova. Stoga nad nizom možemo izgraditi tournament stablo koje u svakom čvoru pamti rješenje, sumu a_i-ova te sumu b_i-ova za pripadajući interval. Tada nije teško spojiti spomenute informacije za dva čvora te na uobičajen način odgovarati na upite u složenosti $\mathcal{O}(\log(N + K))$, stoga je ukupna složenost rješenja $\mathcal{O}((N + K) \log(N + K))$. Za implementacijske detalje pogledajte službene kodove.

Zadatak Totoro

Pripremili: Stjepan Požgaj i Daniel Paleka
Potrebno znanje: matematika, pretraživanje u dubinu/širinu (dfs/bfs), linearnost očekivanja, teorija grupa ili Markovljevi lanci, ad hoc

Prvo definirajmo notaciju. Neka $\mathcal{I}(\pi)$ označava broj inverzija permutacije π. Neka za neku tvrdnju T oznaka $[T]$ označava *boolean* vrijednost te tvrdnje: 1 ako je točna, 0 ako je netočna. (Ta notacija naziva se Iverson bracket.) Kompoziciju permutacija τ i π označavat ćemo s $\tau\pi$.

Skup S je standardan primjer *grupe*: skupa na kojem je definirana neka asocijativna operacija (ovdje kompozicija permutacija), te postoje inverzi svih elemenata i indentiteta . Ovdje nećemo ulaziti u dokaz da je definicijom skupa S zadana podgrupa grupe permutacija, jer čitatelj mora to jednom raspisati sam. Za uvod u grupe, pogledate izvrsni Groups chapter iz Napkina: `https://web.evanchen.cc/napkin.html`.

Permutacije p_i iz ulaza zovu se *generatori* grupe.

Za prvu parcijalu, dovoljno je izračunati grupu S (koja nije pretjerano velika) i prebrojiti inverzije u svakoj od dobivenih permutacija. Ako je slučajno presporo, primijetimo da je prosječan broj inverzija jednak $\frac{1}{2}\binom{N}{2}$ u slučaju da je grupa S jednaka grupi svih permutacija S_N, pa je naivno rješenje moguće ubrzati za konstantni faktor.

Za $K = 1$, dana grupa S je konačna *ciklička*, tj. oblika je $\{1, p, p^2, p^3, \ldots, p^m\}$ za neki m. (Ovdje s p^t označavamo permutaciju p primijenjenu t puta.) Poznato je i lako za dokazati da je m točno najmanji zajednički višekratnik veličina svih ciklusa permutacije.

Za drugu parcijalu, vrijedi $m = N$, pa je dovoljno u $O(N \log N)$ prebrojati inverzije svake permutacije u S. Ako permutacija nije ciklus, tada m može biti jako velik (npr. ciklusi veličina prvih tridesetak prostih brojeva), pa moramo napraviti nešto pametnije.

Za treću parcijalu i puno rješenje, trebamo koncept *linearnosti očekivanja*. Sve što slijedi može se izreći i elementarno, ali je vjerojatnosna terminologija puno prirodnija.

Ako nasumce (s uniformnom vjerojatnošću) biramo permutaciju π iz S, tada je izraz $\frac{1}{|S|}\mathcal{I}(\pi)$ prosječna vrijednost ili *očekivanje* $\mathbb{E}\mathcal{I}(\pi)$ broja inverzija $\mathcal{I}$.

Po definiciji vrijedi

$$\mathcal{I}(\pi) = \sum_{1 \le i < j \le N} [\pi(i) > \pi(j)].$$

Koristeći da očekivanje možemo rastaviti po pribrojnicima, dobivamo

$$\begin{aligned}\mathbb{E}\mathcal{I}(\pi) &= \sum_{1 \le i < j \le N} \mathbb{E}[\pi(i) > \pi(j)] \\ &= \sum_{1 \le i < j \le N} \mathbb{P}[\pi(i) > \pi(j)].\end{aligned}$$

Zato je dovoljno izračunati vjerojatnost da vrijedi $\pi(i) > \pi(j)$ kada uzimamo uniformno nasumičnu permutaciju π iz S.

U slučaju kada je $K = 1$, zadatak se može lijepo matematički riješiti u složenosti $\sum_{C_1, C_2 \text{ciklusi}} |C_1| \cdot |C_2| = O(N^2)$, jer možemo računati $\mathbb{P}[\pi(i) < \pi(j)]$ za sve $i \in C_1, j \in C_2$ istovremeno. Za detalje pogledajte službeni kod rješenja za $K = 1$.

Kada je $K > 1$, moramo zaboraviti na rješenje za $K = 1$, jer je jako teško analizirati strukturu grupe s više generatora.

Ključna ideja je promatrati graf G_{parovi} od N^2 čvorova $(i, j) : 1 \le i, j \le N$, gdje su povezani vrhovi (i, j) i $(p_k(i), p_k(j))$ za svaki $1 \le k \le K$. Ako primijenimo permutaciju π iz S, jasno je da to možemo interpretirati kao "vrh (i, j) ide u vrh $(\pi(i), \pi(j)$".

Vrh (i, j) neka permutacija π iz S može poslati samo u vrhove pripadajuće povezane komponente. Ako bi znali s kojom vjerojatnošću nasumična permutacija π šalje vrh (i, j) u svaki vrh njegove komponente, mogli bi smo eksplicitno izračunati tražene vjerojatnosti. Tu pomaže sljedeća ključna činjenica:

Tvrdnja: Uniformno nasumična permutacija π iz S šalje vrh (i, j) u uniformno nasumičan vrh u njegovoj povezanoj komponenti.
Prvi dokaz: Elementaran, bit će u finalnoj verziji editoriala.
Drugi dokaz: Primijetimo da svaki vrh ima stupanj K, tj. graf je *regularan*. Promatrajmo Markovljev lanac M na vrhovima grafa, tj. slučajnu šetnju koja svaki brid bira s jednakom vjerojatnošću. Lako se dokaže da *stacionarna distribucija* (jedinstvena vjerojatnost na vrhovima grafa koja se ne mijenja u koraku lanca M) svakom vrhu neke povezane komponente pridružuje istu vjerojatnost, jer je graf regularan.

Promotrimo Cayleyjev graf grupe S generirane permutacijama $p_1, \ldots, p_K$, gdje su vrhovi elementi $\pi \in S$, a bridovi između permutacija π i $p_k\pi$ za svaki $1 \leq k \leq K$. Graf je također regularan stupnja K, pa ako definiramo sličnu slučajnu šetnju M_S, stacionarna distribucija je uniformna po svim elementima grupe S.

Sada samo koristimo da slučajna šetnja iz nekog vrha po vjerojatnosti konvergira u stacionarnu distribuciju, kako u lancu M, tako i u lancu M_S. [1]

Slučajne šetnje na lancima M i M_S možemo bijektivno upariti, jer svaki korak u oba lanca odgovara nekoj ulaznoj permutaciji p_k.

Zato vrijedi

$$\begin{aligned}\text{uniformna permutacija } \pi \in S &\equiv \text{slučajna šetnja permutacijama } p_k \\ &\equiv \text{stacionarna distribucija na komponentama grafa } G_{\text{parovi}}.\end{aligned}$$

Stoga, dovoljno je za svaku povezanu komponentu u G_{parovi} izračunati broj čvorova (a, b) takvih da je $a > b$ (*invertiranih* čvorova), jer se čvor (i, j) šalje u svaki čvor komponente s jednakom vjerojatnošću. Tražena vjerojatnost dobiva se dijeljenjem broja invertiranih čvorova s veličinom komponente. Komponente je lako izračunati u složenosti $O(N^2K)$ pretraživanjem u dubinu ili širinu.

[1] Postoje manji problemi s periodičnošću, ali se oni lako riješe takozvanim *lazy* bridovima – ne ulazimo u tehnikalije.