Algel IX. gyakorlat

DAG!

2011. április 4.

4. **[ZH: 2008. március 28.] Egy $n \times n$ méretű táblázat minden eleme egy egész szám. A táblázat bal alsó sarkából akarunk eljutni a jobb felső sarkába úgy, hogy egy lépésben a táblázatban vagy felfelé vagy jobbra egyet lépünk. Azt szeretnénk, hogy a lépegetés során látott elemek növekvő sorrendben kövessék egymást. Egy ilyen út értéke a benne szereplő számok összege. Adjon $O(n^2)$ futási idejű algoritmust, ami meghatározza, hogy az adott táblázatban a szabályok szerinti utak értékei között mekkora a legnagyobb!** *(Persze, dinprog is kézenfekvő, de most DAG-gal!)*
   Felveszünk egy gráfot: csúcsai a táblázat mezői, élei a mezők közötti megengedett lépések (a növekedési feltételt is figyelembe véve), az élsúlyok a mezők értékei. Könnyű látni, hogy egy, a bal alsó saroknak megfelelő csúcsból a jobb felsőig tartó út pont egy megengedett lépegetés, így ebben a gráfban egy leghosszabb utat kell keresni a bal alsó csúcsból indulva. A lépési szabályok miatt a gráf DAG, így $O(|V| + |E|)$ lépésben ez megtehető, ami jelen esetben $O(n^2 + 2n^2) = O(n^2)$, hiszen n^2 csúcsunk van, és minden csúcsból legfeljebb két él indul.

5. **[Vizsga: 2008. május 27.] Éllistával adott egy n pontú e élű irányított gráf. Azt szeretnénk tudni, hogy van-e benne olyan minden pontot tartalmazó részgráf, ami egy, a gyökerétől a levelek felé irányított fa. Adjon $O(ne + n^2)$ lépésszámú algoritmust, ami ha van, talál egy ilyen részgráfot.**
   Minden csúcsból egy mélységi bejárással ellenőrizzük, hogy ilyen-e. Ha találunk, akkor megállunk, ha nem, akkor pedig nem is lehet (különben valamelyik bejárás megadta volna). Ez összesen $O(n(n + e)) = O(ne + n^2)$.

6. **[ZH: 2007. április 27.] Az $n \times n$ méretű tábla minden mezőjére egy pozitív egész szám van írva, az i-edik sorának j-edik elemére $A[i, j]$, ahol $0 \leq i, j < n$. Feladat, hogy az első oszlopból eljussunk az utolsó oszlopba úgy, hogy egy lépésben mindig a következő oszlopba lépünk, és azon belül, ha az i-edik sorban voltunk, akkor a következő lépésben vagy az $(i - 1) \pmod n$, vagy az i, vagy az $(i + 1) \pmod n$ számú sorba kerülhetünk. Adjon $O(n^2)$ lépésszámú algoritmust, ami meghatározza, hogy az első oszlop melyik eleméből induljunk, ha azt akarjuk, hogy a bejárt mezőkön lévő számok összege minimális legyen (az utolsó oszlop bármelyik mezője lehet az utolsó olyan mező, amire rálépünk).**
   A tábla mezőit vesszük egy gráf csúcsainak, az élek a megengedett lépéseket jelölik, súlyuk a mező súlya, ahonnan indulnak. Legyen egy forrás, ami az összes első oszlopbelihez hozzá van kötve 0 súllyal, és egy nyelő, amibe az utolsó oszlopból vezetnek élek. Így egy megengedett lépkedés pont egy útnak fog megfelelni a gráfban a forrástól a nyelőig. A gráf DAG, a lépések definíciója miatt. $|V| = O(n^2)$, $|E| = O(n^2)$ (minden csúcsból legfeljebb 3 él indul, a forrásból pedig n db), így a legkisebb összegű lépkedés pont egy legrövidebb út a forrásból a nyelőbe, ami $O(|V| + |E|) = O(n^2)$ lépésben megvan.

7. **[Vizsga: 2010. június 3.] Egy falutörténet írója n korábbi lakosról gyűjtött információkat. A kérdésekre kapott válaszok a következő típusúak voltak:**

   - **S_i személy meghalt S_j születése előtt;**
   - **S_i személy élete során született S_j;**
   - **S_i személy korábban született, mint S_j;**
   - **S_i korábban halt meg, mint S_j.**

Egy S_i, S_j párra nem biztos, hogy szerepel minden választípus, és olyan pár is lehet, amely egyetlen válaszban sem szerepel együtt. Mivel az emberek időnként rosszul emlékeznek, nem biztos, hogy minden kapott információ helyes. Adjon algoritmust, amivel k db fenti típusú válaszról $O(n+k)$ lépésben eldönthető, hogy van-e közöttük ellentmondás.
Vegyünk fel egy $2n$ csúcsú gráfot, ahol az egyes csúcsok adott személy születésének és halálának felelnek meg (S_i-re $v_{i,sz}$ és $v_{i,h}$). A gráf élei jelentik az időbeli megelőzést, azaz v_k-ból v_l-be futó él azt jelenti, hogy v_k esemény v_l előtt történt. A születési és halál csúcsokat személyenként értelemszerűen összekötjük. A többi él az információkból jön, pl. S_i *személy élete során született* S_j esetén egy $(v_{i,sz}, v_{j,sz})$ és $(v_{j,sz}, v_{i,h})$ élet húzunk be (a többi is értelemszerűen). Az információkban pontosan akkor nincs ellentmondás, ha a gráf DAG (egyik irány: ha lenne ellentmondás, akkor az kört jelentene a gráfban; másik irány: ha kör van a gráfban, az egy esemény saját maga előtti bekövetkeztét jelenti, azaz ellentmondás). $|V| = n$, $|E| = O(n + 2k)$ („élet" élek és állításonként legfeljebb két él), DAG-ság eldöntése mélységi bejárással $O(|V|+|E|) = O(n+k)$.

8. **[ZH: 2005. április 8.] Cirkuszi akrobaták egymás vállára állva minél nagyobb tornyot szeretnének létrehozni (a toronyban minden szinten csak egy akrobata lesz). Esztétikai és gyakorlati szempontok miatt egy ember vállára csak egy olyan állhat, aki nála alacsonyabb és könnyebb is. A cirkuszban n akrobata van, adott mindegyikük magassága és súlya. Adjon algoritmust, amely $O(n^2)$ lépésben megadja a lehetséges legtöbb emberből álló torony összeállítását.**
Felveszünk egy irányított gráfot, ahol a csúcsok az akrobaták, két akrobata között akkor fut él (értelemszerű irányítással), ha az egyik ráállhat a másikra. Egy irányított út a gráfban pont egy helyes egymásraállásnak felel meg. A gráf DAG, ha nem lenne az, akkor valaki saját magánál nehezebb és magasabb lenne. A leghosszabb utat keressük, ami mélységi bejárás segítségével $O(|V| + |E|) = O(n + n^2) = O(n^2)$ ebben az esetben.

9. **[ZH: 2007. április 27.] Tekintsük az olyan G irányított gráfokat, amelyekben ha eltekintünk az élek irányításától, akkor a kapott irányítatlan G' gráf összefüggő. A G gráf egy mélységi bejárásánál maximálisan hány olyan csúcs lehet, amelyre a mélységi és a befejezési szám megegyezik?**
n, ilyen például egy irányított út, ahol pont az iránnyal ellentétesen vesszük a csúcsokat. Több nem lehet, hiszen n csúcsa van a gráfnak.

10. **Adjunk algoritmust, mely egy éllistával megadott irányítatlan gráfban vagy talál egy kört, vagy igazolja a gráf körmentességét $O(|V|)$ időben (függetlenül attól, hogy $|E|$ akár sokkal nagyobb is lehet, mint $|V|$)!**
Ha egy irányítatlan gráfban $n - 1$ élnél több van, akkor biztos van benne kör (de ha ennél kevesebb, attól még lehet benne kör!). Így indítunk egy mélységi bejárást, amit n él vizsgálata után automatikusan leállítunk, egyébként meg magától is a megadott lépéskorlátban megáll.

11. **[Vizsga: 2007. június 12.] Egy számítógéphálózatban n számítógép van. Minden olyan eseményt, hogy az i-edik gép üzenetet küld a j-ediknek (i, j, t) formában feljegyezünk, ahol a t egész szám az üzenet küldésének időpontját jelöli. Ugyanabban a t időpontban egy gép több gépnek is küldhet üzenetet. Ha a t időpontban az i-edik gép vírusos volt, akkor egy (i, j, t) üzenet hatására a j-edik gép mefertőződhet, ami azt jelenti, hogy a $t + 1$ időponttól kezdve már a j-edik gép is vírusos lehet. Legyen adott az (i, j, t) hármasoknak egy m hosszú listája, valamint x, y és $t_0 < t_1$ egész számok. Azt kell eldöntenünk, hogy ha az x-edik gép a t_0 időpontban vírusos volt, akkor lehet-e emiatt az y-adik gép a t_1 időpontban vírusos. Adjon algoritmust, ami ezt a kérdést $O((t_1 - t_0)n + m)$ lépés után megválaszolja.**
Vegyünk fel egy gráfot, csúcsai a számítógépek minden időpillanatban t_0 és t_1 között, élei (irányítottak) az üzenetek. Vegyünk fel még éleket az ugyanahhoz a géphez tartozó szomszédos időpontok között is előrefele! Ebben a gráfban az x géppel pontosan azok vannak egy kom-

ponensben, akik lehetnek vírusosak (ezt kicsit indokolni kell). Így x-ből egy bejárást indítva megtaláljuk a kérdéses gépeket. $|V| = O(n(t_1 - t_0))$, $|E| = O(n(t_1 - t_0) + m)$ (saját élek és legfeljebb m üzenet), így a bejárás lépésszáma $O((t_1 - t_0)n + m)$.

12. **[Vizsga: 2007. június 19.] Egy előre rögzített útvonalon úgy indulunk el, hogy az autó L literes tankja tele van. Úticélunkhoz úgy akarunk eljutni, hogy legalább egy fél tanknyi benzin maradjon az autóban. Tudjuk, hogy az utunkba eső n benzinkút közül melyikben mennyibe kerül a benzin, továbbá, hogy két szomszédos benzinkút között, valamint a kiindulóponttól az első benzinkútig, illetve az utolsó benzinkúttól a célunkig mennyi benzint fogyaszt az autó. Az egyszerűség kedvéért ha megállunk egy benzinkútnál, akkor mindig tele tankolunk. Adjon algoritmust, ami $O(Ln^2)$ lépésben megmondja, hogy hol álljunk meg tankolni ha azt akarjuk, hogy utunk során a benzinköltség minimális legyen.** *(Javítási útmutatóban: ELNEZEST, a feladatba bele akartam irni, de kimaradt, hogy a fogyasztas mindig egesz liter. Ha valaki megoldotta e nelkul (es meg jobb is a lepesszama), annak orulunk. Ha valaki feltette, hogy minden egesz, annak is orulunk, mert ezt akartuk es meg gondolatot is tud olvasni.)*
Építünk egy gráfot, ami mind az n közbeeső benzinkúthoz (+ a kezdő- és végponthoz) tartalmaz L csúcsot; $v_{i,l}$ jelentése: az i-edik kúthoz l liternyi benzinnel érkeztünk. Az élek egyértelműen behúzhatók a fogyasztás alapján (vagy tankolunk és úgy megyünk tovább, vagy nem tankolunk), így minden csúcsból legfeljebb 2 él indul ki; a feltételek alapján tiltottakat (elfogyó benzin, kevés benzinnel célbaérés) eleve nem húzzuk be. Az élek súlya 0, ha nem tankolunk, és a tankolás költsége (a tankolandó mennyiség és az ár ismeretében konstans időben meghatározható), ha tankolunk. Ebben a gráfban egy legrövidebb út pont a legolcsóbb utazást fogja jelenteni. A gráf DAG (csak benzinkutak között, előrefele mennek élek), így a legrövidebb út megkeresésének lépésszáma $O(|V| + |E|) = O(Ln + 2Ln) = O(Ln)$, ami pont jó is. (Ha nem csak a szomszédos kutakat néznénk, hanem minden kútból behúznánk az összes élet, ahova eljuthatunk, akkor kutanként és literenként 2 helyett $O(n)$ él indulhatna ki, így adódna ki $O(Ln^2)$ – természetesen $O(Ln) = O(Ln^2)$, így a leírt megoldás is helyes).

13. **A $G(V, E)$ összefüggő, irányított gráf minden éle az $1, 2, \ldots k$ számok valmelyikével van súlyozva. Egy út értéke legyen az úton található élek súlyainak maximuma. Adjunk $O(|E| \log k)$ futásidejű algoritmust az adott $x, y \in V$ csúcsok közti legkisebb súlyú út értékének meghatározására!**
Adott i számra meg tudjuk nézni, hogy x és y csúcs között van-e legfeljebb i súlyú út, azaz az i-nél nem nagyobb súlyú élek kihagyásával x és y egy komponensben van-e. A legkisebb olyan i szám, amire x és y egy komponensben vannak, pont a minimális súlyú út súlya. Mivel csak $1, \ldots, k$ közötti élsúlyaink vannak, bináris kereséssel meg tudjuk keresni a megfelelő i-t. A komponensbe tartozás ellenőrzése mehet bejárással, ami $O(|E| + |V|)$, viszont itt az összefüggőség miatt $|V| = O(|E|)$. Összesen tehát a lépésszám $O(|E| \log k)$.

14. **[Vizsga: 2003. május 30.] Éllistával adott egy G gráf, melynek n csúcsa és e éle van. A gráf minden csúcsához hozzá van rendelve egy 1 és k közötti egész szám (címke). Találjunk (ha létezik) olyan *tarka* utat a gráfban, amelyben minden $1 \leq i \leq k$ címke pontosan egyszer fordul elő. Az algoritmus lépésszáma legyen $O(k!(e + n))$.**
Ha adott egy π permutáció, akkor egy bejárással $O(e + n)$ lépésben el tudom dönteni, hogy van-e megfelelő, k hosszú út (csak a π szerinti aktuálisan következő címkéjű éleket vesszük létezőnek a bejárás lépései során). Összesen $k!$ permutáció lehet, így mindet végig tudjuk nézni $O(k!(e + n))$ lépésben.

15. **Bizonyítsuk be, hogy minden $G = (V, E)$ irányított gráf felbontható két DAG-ra; pontosabban az élhalmazának van olyan E_1, E_2 partíciója ($E = E_1 \cup E_2$ és $E_1 \cap E_2 = \emptyset$), hogy a $G_1 = (V, E_1)$ és a $G_2 = (V, E_2)$ gráfok DAG-ok!**
Indukcióval a pontszámra. $n = 1$-re triviálisan igaz. Tfh n-re igaz, kérdés, hogy igaz-e $n + 1$ pontra? Ha van egy $n + 1$ pontú irányított gráfunk, akkor véletlenszerűen válasszuk ki egy

pontját. A maradék n az indukciós feltevés szerint felbontható megfelően két DAG-ra (persze simán lehet, hogy az egyik, másik, vagy mindkettő akár 0 élet tartalmaz, de ez minket nem zavar), ezeknek van egy topologikus sorrendje. Az $n+1$-edik pont bejövő éleit rendeljük az egyik DAG-hoz, ettől az DAG marad (a topologikus sorrendben az aktuális lesz az utolsó pont, a többit nem zavarja), a kimenő éleket pedig a másik DAG-hoz (hasonlóan az is DAG marad). Így az állítást igazoltuk.