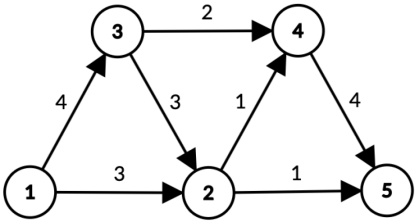
Tema 2 Seminar

Tender Laura-Maria

- 31 decembrie 2020 -

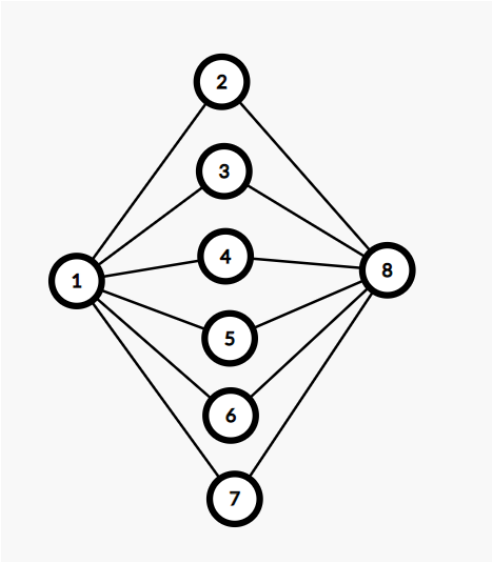
Exercițiul 1



Pe rețeaua de mai sus căutăm fluxul maxim de la $S = 1$ la $T = 5$. Având în vedere dimensiunile rețelei, privind figura, observăm ușor că fluxul maxim este 4.
Putem demonstra acest lucru folosind teorema de corectitudine care ne spune că fluxul maxim este egal cu tăietura minimă. Împărțim nodurile în 2 mulțimi distincte astfel încât sursa și destinația să se afle în mulțimi diferite. Tăieturile sunt mulțimea muchiilor ce despart cele două mulțimi de noduri. O căutăm pe cea de cost minim.
Există următoarele posibilități:
{1} , {2, 3, 4, 5} cu tăietura formată din muchiile 1-2, 1-3 de cost 7;
{1, 2}, {3, 4, 5} cu tăietura formată din muchiile 1-3, 2-4, 2-5 de cost 6;
{1, 3}, {2, 4, 5} cu tăietura formată din muchiile 1-2, 3-2, 3-4 de cost 8;
{1, 2, 3}, {4, 5} cu tăietura formată din muchiile 2-5, 2-4, 3-4 de cost 4;
{1, 3, 4}, {2, 5} cu tăietura formată din muchiile 1-2, 3-2, 4-5 de cost 10;
{1, 2, 3, 4}, {5} cu tăietura formată din muchiile 4-5, 2-5 de cost 5.
Observăm că tăietura minimă este 2-5, 2-4, 3-4 de cost 4. Deci, fluxul maxim este 4.

Exercițiul 2

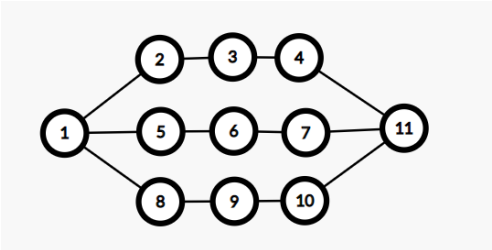
Fie un graf precum următorul:



Ne vom gândi la acest graf ca la o rețea de flux cu sursa 1 și destinația 8. Capacitatea muchiilor nu este reprezentată în figură. Algoritmul Edmonds-Karp caută cele mai scurte drumuri către destinație. Cât timp găsește un drum pe care mai poate fi transmis flux, determină cantitatea de flux care mai poate fi pompată și o adaugă. Pentru acest graf algoritmul Edmonds-Karp va face 8 parcurgeri BFS (numărul poate să difere în funcție de implementare).

Vom generaliza acestă rețea la una cu n noduri dintre care unul este sursa și celălalt destinația. Celelalte $n-2$ noduri vor fi conectate doar la sursă și la destinație. Astfel, pentru acest graf, algoritmul Edmonds Karp va face $O(n)$ parcurgeri BFS care au complexitate $O(n+E)$. Deci, are o complexitate $O(nE)$.

Putem generaliza rețeaua și mai mult, astfel încât între sursă și destinație să se afle mai multe lanțuri disjuncte. Un exemplu pentru a vizualiza este următorul graf:



Pentru un astfel de graf cu k lanțuri distincte, algoritmul ar face, în funcție de implementare, aproximativ k parcugeri BFS. Deci am obține aproximativ $k(n+E) < 2E^2$ pași, $k < n < E$.

Astfel, adaptarea Edmonds Karp a algoritmului de determinare a fluxului maxim are o complexitate de timp ce se încadrează în $\Omega(E^2)$ pe cazul nefavorabil.

Exercițiul 3

În orice nod, fluxul care intră este egal cu cel care iese.
Întrucât T este destinația, ne intereseaza fluxul care intră în T. Dacă într-o rețea există muchii care pleacă din destinația T, pe acestea nu se poate duce mai mult flux decât a intrat, acestea nu pot face decât ca fluxul să rămănă egal (în cazul unui ciclu în care muchiile se întorc în rețea) sau să scadă. Acestea nu influențează fluxul maxim, deci pot fi eliminate.

Întrucât S este sursa, ne interesează fluxul care pleacă din S. Fluxul care pleacă din S poate fi maxim fluxul muchiilor care pleacă din S. Dacă într-o rețea există muchii care ajung în sursa S, acestea pot fi de două tipuri: care nu sunt conectate la drumul dintre S și T (pot fi eliminate întrucât nu influențează fluxul care pleacă inițial din S) și care sunt conectate la drumul dintre S și T (pe acestea nu se poate întoarce decât fluxul care a plecat din S, ar intra într-o buclă și nu ar influența fluxul maxim, deci și acestea pot fi eliminate).

A se întoarce flux pe muchiile care ajung în T este echivalent cu a exista o muchie care pleacă din T, doar că aceasta este de întoarcere. Conform demonstrației anterioare, nu este posibil. A se întoarce flux pe muchiile care pornesc din S este echivalent cu a exista o muchie care ajunge în S, doar că aceasta este de întoarcere. Conform demonstrației anterioare, nu este posibil.

Exercițiul 4

În problema noastră, în plus față de problema clasică de flux în care muchiile u-v au capacitatea $c(u, v)$, și nodurile au capacitate. Putem reduce această problema la cea anterioară transformând constrângerea la nivel de noduri în una la nivel de muchii. Orice nod v va fi înlocuit de nodurile v' și v'', între care există muchie orientată de la v' la v'', cu costurile $c(v', v'') = c(v)$ și $c(v') = c(v'') = 0$. În nodul v' vor intra în muchiile care întră în v, iar din nodul v'' vor ieși muchiile care ies din v. Pe această rețea, în care muchiile au cost, iar nodurile nu, putem aplica un algoritm de flux maxim.

La graful inițial cu N noduri și M muchii am adaugat N noduri și N muchii. Algoritmul Dinic are o complexitate $O(N^2M)$.
$N < M$, $(N + N)^2 \cdot (M + N) = 4N^2M + 4N^3 < 8N^2M$. Deci, folosind algoritmul Dinic, prin această modificare, problema inițială poate fi rezolvată în aceeași complexitate $O(N^2M)$ ca cea inițială.

Exercițiul 5

T este un arbore cu n noduri, în fiecare nod i, $i \in \overline{(1,n)}$ se află s_i oameni și d_i locuințe. Un om poate locui în orașul său, în rădăcină sau în orice oraș situat pe drumul către rădăcină. Fiecare om se poate stabili într-o locuință dacă putem cupla fiecare om cu o locuință. Fie două mulțimi cu câte n noduri: mulțimea orașelor $i \in \overline{(1,n)}$ și mulțimea oamenilor ce trebuie să plece din orașul i. Din fiecare nod trebuie să plece $s_i - d_i$ oameni, dacă $s_i - d_i > 0$, altfel 0.
Folosind aceste două mulțimi de noduri, numărul oamenilor în surplus în orașe și ținând cont de regula de relocare vom crea rețeaua de flux astfel:

1. Adăugăm o sursă S și o destinație T.
2. Conectăm fiecare nod din mulțimea oamenilor de sursa S, capacitatea muchiei fiind $s_i - d_i$, dacă $s_i - d_i > 0$, altfel putem exclude nodul din graf pentru a optimiza rezolvarea.
3. Conectăm fiecare nod din mulțimea orașelor de destinația T, capacitatea muchiei fiind $d_i - s_i$, dacă $d_i - s_i > 0$, , altfel putem exclude nodul din graf pentru a optimiza rezolvarea.
4. Creez muchie de capacitate infinit de la oamenii din orașul i către toate orașele care există în graf în care se pot reloca - orașele de pe drumul la i la capitală, inclusiv capitala, dacă aceasta se află în graf.

Vom aplica un algoritm de flux maxim. Dacă fluxul maxim obținut este egal cu cel care a plecat din sursa S atunci fiecare persoană găsește o locuință în care să se stabilească.
Graful acestei rețele conține $2n + 2$ noduri și maxim aproximativ n^2 muchii. Deci, în funcție de algoritmul de flux folosit, putem afla fluxul maxim într-o complexitate $O(mn^2) = O(n^4)$ (prin algoritmul lui Dinic).

O soluție Greedy de complexitate $O(n)$ ce folosește structura de arbore este următoarea: pornim o parcurgere DFS din rădăcină, "trimițând" oameni pe muchiile de întoarcere. Saturăm nodurile la maxim și trimitem "în sus" oamenii în surplus. Dacă atunci când ajungem din nou în rădăcină există oameni care nu au locuință atunci nu există o strategie de relocare.

Întrucât graful are structură de arbore, fiecare nod are un tată, mai puțin rădăcina. Adică prin DFS există o singură cale de întoarcere. Voi descrie mai detaliat și mai formal acest algoritm. Pentru fiecare nod i vom reține s_i numărul de oameni stabiliți în acel oraș, d_i numărul de locuințe, a_i numărul de oameni care tocmai a fost trimis dintr-un oraș-fiu și t_i numărul de oameni care vor fi trimiși către orașul-tată. Pentru fiecare nod, dacă $s_i > d_i$ atunci $s_i = d_i$ și $t_i = s_i - d_i$, altfel s_i rămâne cu valoarea inițială și $t_i = 0$. Prima întoarcere va porni dintr-o frunză. Toate frunzele au $a_i = 0$. La întoarcerea din nodul v, tatăl său u primește t_v oameni. a_u devine t_v. Dacă $d_u - s_u >= a_u$, atunci $s_u += a_u$, altfel $t_u += a_u - (d_u - s_u)$, iar $s_u = d_u$. Dacă t_1 (am considerat că

rădăcina are indexul 1) > 0 atunci problema nu are soluție.

În rezolvarea aceastei problemei nu ținem cont în ce oraș este așezată o persoană anume, dacă este mai aproape sau mai departe de capitală, cât timp respectă cerința problemei. Strategia soluției este de a le acorda oamenilor prima locuință liberă găsită. Presupunem că acestă strategie ar crea probleme, adică procedând astfel, nu am găsi locuințe pentru un număr maxim de oameni. Atunci există o altă soluție optimă, diferită de cea propusă. Vom studia primul punct în care soluția optimă ar fi diferită de cea optimă. Ajungem într-un nod u în care alegem să așezăm un număr de oameni mai mic decât capacitatea maximă, atunci acești oameni vor fi trimiși către orașele de pe drumul către capitală. Asfel, din nodul u vom trimite un număr de oameni cel puțin egal cu cel din soluția propusă. Numărul de oameni cuplați cu locuințe este egal cu $\sum_{i=1}^{n} s_i - t_1$. Astfel, nefolosind toate locuințele de pe drum la capacitatea maximă, în rădăcină va ajunge un număr de oameni cel puțin egal cu cel din soluția propusă, iar acesta poate fi mai mare decât capacitatea.

În concluzie, obținem o contradicție, nu există o altă soluție prin care să putem găsi locuințe pentru un număr mai mare de oameni. Deci, soluția propusă este corectă.

Exercițiul 6

O echipă poate transfera maxim un jucător, iar dacă transferă un jucător trebuie să și primească unul. Căutăm suma prețurilor transferurilor maximă. Vom modela problema ca una de cuplaj maxim de cost minim.

Vom efectua un cuplaj între două mulțimi a câte n noduri, prima mulțime reprezentând echipele care transferă jucători, iar cea de-a doua echipele în care se transferă jucătorii. Vom crea muchii de la a_i (echipă care se află în prima mulțime de noduri) către b_i (echipă din a doua mulțime) cu costul $-p_i$ și cuplajul 1. Am atribuit prețul negativ pentru a putea afla costul minim. Conectăm sursa cu nodurile din prima mulțime prin muchii de cost 0 și cuplaj 1. Conectăm destinația cu nodurile din a doua mulțime prin muchii de cost 0 și cuplaj 1.

Dacă am obține un cuplaj perfect, acesta ar fi egal cu n. Cum o echipă nu poate transfera mai mult decât un jucător (nu poate fi cuplată cu mai multe noduri), fiecare nod ar fi cuplat cu un singur alt nod $\iff$ ar fi transferat un jucător la o singură altă echipă. Cuplajul muchiilor din a doua mulțime către destinație este 1, deci o echipă ar fi primit un jucător de la o singură altă echipă. Astfel, am știi că fiecare echipă a transferat și a primit un jucător.

Pentru a ne asigura că îndeplinim cerința că echipele vor conține în urma transferurilor același număr de jucători, vom adauga în rețea o muchie de la fiecare echipă din prima mulțime la aceeași echipă din a doua mulțime cu costul 0 și

cuplajul 1. Astfel, vom obține întotdeauna cuplaj perfect. Conform explicației de mai sus, acest aspect este echivalent cu îndeplinirea cerinței. Putem implementa folosind un algoritm precum Bellman Ford și vom obține costul minim P. Răspunsul căutat este $-P$, iar transferurile ce au fost efectuate pot fi determinate analizând matricea cuplajului la final.

Exercițiul 7

Fie n fete și n băieți. Fiecare are o listă de k persoane distincte de sex opus cu care preferă să danseze. Pentru a respecta preferințele lor, o pereche poate fi formată dacă fiecare se regăsește pe lista preferințelor celuilalt.

Pentru fiecare băiat vom determina câte dintre cele k fete îl au de asemenea pe lista lor de preferințe. Analog, pentru fiecare fată. Dacă unul dintre aceste numere este mai mic decât k nu pot fi organizate k runde fără a încălca nicio preferință.

În cazul în care fiecare persoană poate forma o pereche cu k persoane de sex opus vom costrui un graf bipartit. Vom reprezenta băieții prin n noduri b_i, $i \in \overline{(1,n)}$, și, de asemenea, fetele prin n noduri f_j, $j \in \overline{(1,n)}$. Vom crea o muchie de la un nod b_i la un nod f_j dacă aceștia pot forma o pereche în cadrul unei runde, în urma preferințelor lor. Astfel am obține un graf k-regulat.

Teorema Hall spune: Fie un graf bipartit G cu părțile A și B. Pentru orice submulțime $X \subset A$, $|N(X)| \geq |X|$, unde $N(X)$ este submulțimea nodurilor din mulțimea B care sunt conectate cu nodurile din submulțimea X. Într-un graf k-regulat, fiecare nod este conectat cu k noduri. Astfel submulțimea de noduri X ar avea exact $k \cdot |X|$ muchii adiacente. Cum numărul de muchii incidente în nodurile din a doua mulțime este tot k, fiecare mulțime de $k \cdot |X|$ muchii va fi incidentă în cel puțin $|X|$ noduri. Conform teoremei Hall, un graf k-regulat are un cuplaj perfect. Odată ce efectuăm acest cuplaj, putem elimina muchiile corespunzătoare lui și obținem un graf $k-1$ regulat. Cât timp $k \geq 1$ vom găsi un cuplaj perfect. Deci putem găsi k cuplaje perfecte. Acestă demonstație susține că această coregrafie poate fi realizată cât timp fiecare persoană poate forma o pereche cu alte k persoane de sex opus.

Pentru a determina o configurație, putem modela aceasta problema ca pe una de flux maxim, adăugând un nod sursă și unul destinație. Vom construi muchii de la sursă către nodurile b_i de capacitate k și de la nodurile f_j către destinație de capacitate k. Muchiile au capacitate k întrucât fiecare persoană va dansa k runde. Muchiile dintre b_i și f_j au capacitea egală cu 1, întrucât o pereche nu poate dansa mai multe runde împreună. Dacă în această rețea, aplicând un algoritm de flux am obține fluxul maxim egal cu $n \cdot k$ atunci coregrafia poate fi realizată și poate fi determinată o configurație. Dacă fluxul maxim este mai mic, atunci coregrafia nu poate fi realizată. (Nu putem obține flux mai mare

decât $n \cdot k$ întrucât atât pleacă din sursă).

În concluzie, dacă în urma preferințelor dansatorilor putem trasa câte k muchii, obținem un graf k regulat și problema are soluție, în caz contrar nu. Pentru a afla o posibila configurație a perechilor folosim rețeaua de flux asupra căreia aplicăm algoritmul.