Algoritmi fundamentali
Notițe

Laura Tender

- Ianuarie 2021 -

CURS 1 2 Notiuni fundamentale

Drum(walk): secvență de vârfuri cu proprietatea că între oricare două vârfuri consecutive există arc.
În graf orientat:

- Drum **simplu** (trail): nu conține un arc de mai multe ori
- Drum **elementar** (path): nu conține un vârf de mai multe ori

Lungimea unui drum: numărul muchiilor parcurse
Circuit: drum cu capete identice (circuit simplu = drum simplu cu capete identice, circuit elementar = drum elementar cu capete identice)
Graf **partial**: aceleași vârfuri, submulțime de muchii
Subgraf: submulțime de vârfuri, submulțime de muchii
Subgraf **indus** de V_1: vârfurile V_1 și toate muchiile cu capetele în V_1
Graf **complementar** lui $G = (V, E)$: $V = V$, muchiile = muchiile din graful complet cu vârfurile V–E.

CURS 1 3 Havel Hakimi

Construcția de grafuri cu secvența gradelor dată
Condiții necesare pentru existența lui G:

- $d_1 + ... + d_n$ – număr par
- $d_i \leq n$–$1, \forall i \in \overline{(1, n)}$

Algoritm Havel Hakimi

- Dacă $d_1 + ... + d_n$ este impar sau există vârf cu $d_i > n - 1$, atunci ne oprim
- cât timp avem vârfuri execută: alege vârful cu cel mai mare grad d_i, îl elimină, scade gradul a d_i noduri cu cel mai mare grad, adaugă la G muchia, trece la următorul nod
  Dacă obținem grad –1 < 0, atunci oprim.

Complexitate O(n^2) swap, O($n^2 \log n$) sortare

Teorema Havel-Hakimi
O secvență de $n \geq 2$ numere naturale $\{d_1 \geq \ldots \geq d_n\}$ cu $d_1 \leq n-1$ este secvența gradelor unui graf neorientat (cu n vârfuri) $\iff$ $\{d_2 - 1 \geq d_{d_1} - 1 \geq \ldots \geq d_n\}$ secvența este secvența gradelor unui graf neorientat (cu $n-1$ vârfuri).
BFS O($V + E$)
DFS O($V + E$)

Curs 2 2 Sortare Topologică

Propoziţie. Dacă G este aciclic atunci G are o sortare topologică.
Lemă. Dacă G este aciclic, atunci G are cel puţin un vârf v cu gradul intern 0 (d(v) = 0).
Algoritm
Pornim cu toate vârfurile cu grad intern 0 și le adăugăm într-o coadă.
Repetăm:

- extragem un vârf din coadă
- îl eliminăm din graf (scădem gradele interne ale vecinilor, nu îl eliminăm din reprezentare)
- adăugăm în coadă vecinii al căror grad intern devine 0

Complexitate: O($V + E$)

Curs 2 3 Muchii critice

O muchie este **critică** dacă nu este conținută într-un ciclu.
Un vârf v este **punct critic** $\iff$ există două vârfuri $x, y \neq v$ astfel încât v aparține oricărui lanț x,y.
Rădăcina este punct critic. Un alt vârf i din arbore este critic $\iff$ are cel puțin un fiu j cu niv min[j] $\geq$ nivel[i].

Curs 2 4 Componente tare conexe

Un graf orientat este **slab conex** dacă exista un drum de la oricare nod la oricare altul considerând muchile grafului neorientate.
Un graf orientat este **tare conex** dacă există un drum de la orice nod la oricare altul.
Algoritm componente tare conexe **Kosaraju**
Folosește două căutări în adâncime, una în graful G şi una în graful transpus

GT.
Complexitate: O($V + E$).
Algoritm Tarjan
Complexitate: O($V + E$).

Curs 3 1 Arbori parțiali de cost minim Kruskal

Proprietate. Orice graf neorientat conex conține un arbore partial.
Algoritmi de determinare a unui arbore parțial al unui graf conex: arborele asociat unei parcurgeri este arbore partial, determinăm un arbore parțial printr-o parcurgere.
Algoritm Kruskal
Alegem muchiile cu cel mai mic cost care nu formează un ciclu. Ținem muchiile sortate crescător după cost. Pentru fiecare vârf reținem din ce componentă conexă face parte. Dacă vârfurile unei muchii nu sunt în aceeași componentă conexă adăugam muchia la APM și actualizăm componenta conexă din care fac parte nodurile.
Complexitate: O(m log n $+n^2$).
Obs: mlog m = m log(n^2) = 2 m log n.
Memorăm componentele conexe ca arbori, folosind vectorul tată. Reprezentantul componentei va fi rădăcina arborelui.
Complexitate: O(m log n).

Curs 3 2 Arbori parțiali de cost minim Prim

Algoritmul lui Prim
La un pas este selectată o muchie de cost minim de la un vârf deja adăugat la arbore la unul neadăugat.
Complexitate: O(m log n) - heap

Curs 4 1 Drumuri minime Dijkstra

Algoritmul Dijkstra
Algoritm pentru grafuri orientate cu cicluri și cu costurile muchiilor pozitive.
Dacă nodul e nevizitat îl marcăm ca vizitat și adăugăm în heap vecinii cu costurile lor.
Complexitate: O(m log n).

Curs 5 1 Bellman Ford

Algoritm pentru grafuri orientate cu circuite, nu funcționeaza pentru circuite de cost negativ dar le detectează
Reținem dacă nodul se află în coadă. Dacă relaxăm muchia îi adăugam vecinii dacă nu sunt în coadă. Dacă un nod a fost în coadă de mai mult de n ori există ciclu de cost negativ.
Complexitate: O(m n).

Curs 5 2 Drumuri minime aciclice

DAGs = Directed Acyclic Graphs
Arcele pot avea și cost negativ.
Drumuri minime de sursă unică în grafuri aciclice DAGs – cu sortare topologică
Complexitate: O(m+n).

Curs 6 2 Drumuri minime între toate perechile

G graf orientat, ponderile pot fi și negative dar nu există circuite cu cost negative în G.
Algoritm Floyd-Warshall
Pentru fiecare nod intermediar k actualizăm distanta de la i la j.
Complexitate: O(n^3).

Curs 7 1 Fluxuri

Algoritm Ford-Fulkerson
Cât timp există un s-t lanţ nesaturat în graf: determină un astfel de lanţ, revizuim fluxul de-a lungul lanţului.
Complexitate: O(f m), f flux maxim = tăietură minimă, m număr de muchii
Algoritm Edmonds-Karp
Parcurgerea BF ⇒ determinăm s-t lanțuri nesaturate de lungime minimă
Complexitate: O(nm^2)

Curs 9 1 Cuplaj maxim

Algoritm de determinare a unui cuplaj maxim în $G = (X \cup Y, E)$:
1.Construim N reţeaua de transport asociată
2.Determinăm f^* flux maxim în N
3.Considerăm $M = \{xy | f^*(xy) = 1, x \in X, y \in Y, xy \in N\}$ (pentru fiecare arc cu flux nenul xy din N care nu este incident în s sau t, muchia xy corespunzătoare din G se adaugă la M)

4. return M
Complexitate: O(mn), C = 1

Curs 9 2 aplicații

Teorema Hall
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.

Curs 9 1 Grafuri euleriene

Ciclu eulerian – traseu închis care trece o singură dată prin toate muchiile.
Teorema lui Euler
Fie G=(V,E) un (multi)graf neorientat, conex, cu $E \neq \emptyset$. Atunci G este eulerian $\iff$ orice vârf din G are grad par.
G are un lanț eulerian $\iff$ G are cel mult două vârfuri de grad impar.

Algoritmul lui Hierholzer
Determinarea unui ciclu eulerian într-un graf conex(sau un graf conex+ vârfuri izolate) cu toate vârfurile de grad par.
Complexitate: O(m)

Teoremă – Descompunere eulerianǎ
Fie G=(V, E) un graf orientat, conex (= graful neorientat asociat este conex), cu exact 2k vârfuri de grad impar($k > 0$). Atunci există o k-descompunere eulerianǎ lui G și k este cel mai mic cu această proprietate.

Teorema lui Menger
Fie G graf orientat, s, t două vârfuri distincte în G. Numărul minim de arce care trebuie eliminate pentru ca s şi t să nu mai fie conectate printr un drum (să fie separate) este numărul maxim de drumuri arc disjuncte de la s la t.

Curs 10 2 Grafuri planare

G = (V, E) graf neorientat s.n. **planar** $\iff$ admite o reprezentare în plan a.î. muchiilor le corespund segmente de curbe continue care nu se intersectează în interior unele pe altele.

Teorema poliedrală a lui EULER
Fie G=(V, E) un graf planar conex și M = (V, E, F) o hartă a lui. Are loc relația $|V| - |E| + |F| = 2$.

Fie G=(V, E) un graf planar conex cu $n = |V| > 2$ și $m = |E|$. Atunci:
a) $m \leq 3n$–6
b) $\exists x \in V$ cu $d(x) \leq 5$.

Fie G=(V, E) un graf planar conex bipartit cu $n = |V| > 2$ și $m = |E|$. Atunci:
a) $m \leq 2n$–4
b) $\exists x \in V$ cu $d(x) \leq 3$.

Curs 11 1 Grafuri hamiltoniene

Un graf este **hamiltonian** dacă admite un ciclu hamiltonian adică un ciclu care conține toate nodurile grafului.

Teorema lui Dirac: Fie G un graf cu cel puțin 3 noduri. Dacă $\delta(G) \geq \frac{n}{2}$ (fiecare nod are cel puțin $\frac{n}{2}$ vecini) atunci G este hamiltonian.

Teorema lui Ore: Fie G un graf cu ordinul $n \geq 3$, dacă avem pentru oricare pereche de noduri neadiacente deg(x) + deg(y) $\geq n$ atunci graful este hamiltonian.

Teorema lui Chvatal și Erdos
Fie G un graf conectat cu ordinal $n \geq 3$, conectivitatea K(G), și numărul de independență $\alpha(G)$. Dacă $K(G) \geq \alpha(G)$, atunci G este Hamiltonian. K(G) = conectivitatea $\alpha(G)$ = numărul de independeță.

Teorema lui Goodman si Hedetniemi
Daca G este un graf 2-conectat și liber de $K_{1,3}$, Z_1 atunci G este hamiltonian.

Genereare toate permutările **Complexitate** O(n! n) (generarea permutărilor și parcurgerea lor, dar pot fi parcurse doar permutările dintr-un nod, dacă ar exista un ciclu ar fi suficient pentru a îl găsi O(n!)
Dinamică **Complexitate** $O(2^n \cdot n^2)$

Curs 12 3 P, NP, EXP

Teorema Konig
Pentru orice graf bipartit G, dimensiunea celei mai mici acoperiri cu noduri MIN-VC(G) este egală cu dimensiunea cuplajului maxim MAX-MTC(G).

Algoritmi Monte Carlo
Furnizează totdeauna un rezultat, care însă nu este neapărat corect. Probabilitatea ca rezultatul să fie corect creşte pe măsură ce timpul disponibil creşte.

Algoritmi Las Vegas
Nu furnizează totdeauna un rezultat, dar dacă furnizează un rezultat atunci acesta este corect. Probabilitatea ca algoritmul să se termine creşte pe măsură ce timpul disponibil creşte.