UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiectarea Algoritmilor

21. Algoritmul lui Prim

Bibliografie

[1] http://monalisa.cacr.caltech.edu/monalisa__Service_Applications__Monitoring_VRVS.html

[2] http://www.cobblestoneconcepts.com/ucgis2summer2002/guo/guo.html

[3] Giumale – Introducere in Analiza Algoritmilor cap. 5.5

[4] R. Sedgewick, K Wayne – curs de algoritmi Princeton 2007 www.cs.princeton.edu/~rs/AlgsDS07/ 01UnionFind si 14MST

[5] http://www.pui.ch/phred/automated_tag_clustering/

[6] Cormen – Introducere în Algoritmi cap. 24

Algoritmul lui Prim

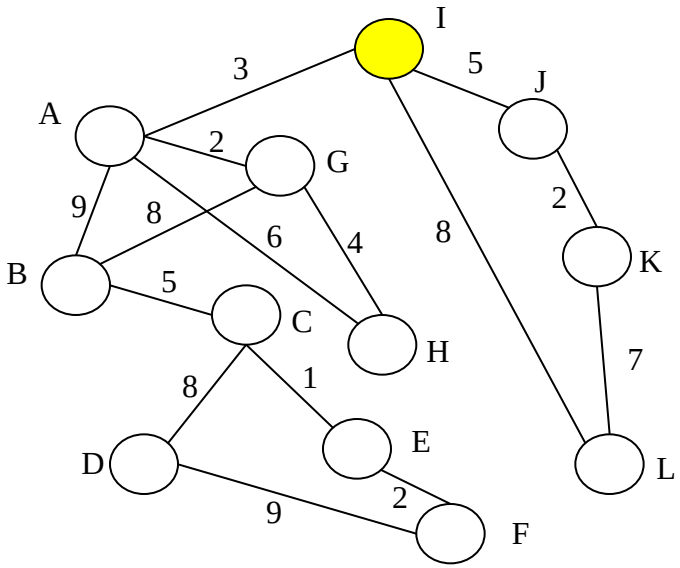
- Prim(G,w,s)

Implementare în Java la [4] !

  - A = ∅ // AMA
  - **Pentru fiecare** (u ∈ V)
    - d[u] = ∞; p[u] = null // inițializăm distanța și părintele
  - d[s] = 0; // nodul de start are distanța 0
  - Q = constrQ(V, d); // ordonată după costul muchiei
    // care unește nodul de AMA deja creat
  - **Cât timp** (Q != ∅) // cât timp mai sunt noduri neadăugate
    - u = ExtrMin(Q); // extrag nodul aflat cel mai aproape
    - A = A ∪ {(u,p[u])}; // adaug muchia în AMA
    - **Pentru fiecare** (v ∈ succs(u))
      - **Dacă** d[v] > w(u,v) **atunci**
        - d[v] = w(u,v); // actualizăm distanțele și părinții nodurilor
        - p[v] = u; // adiacente care nu sunt în AMA încă
  - **Întoarce** A - {(s,p(s))} // prima muchie adăugată

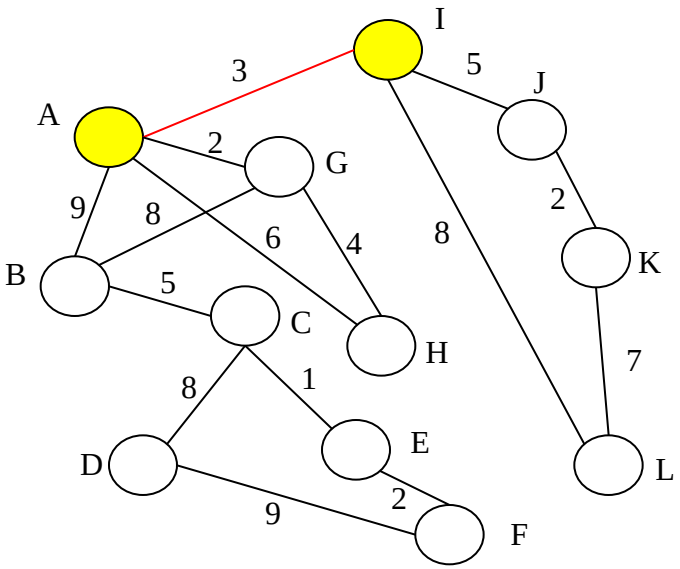
Exemplu (I)

- Pornim din I



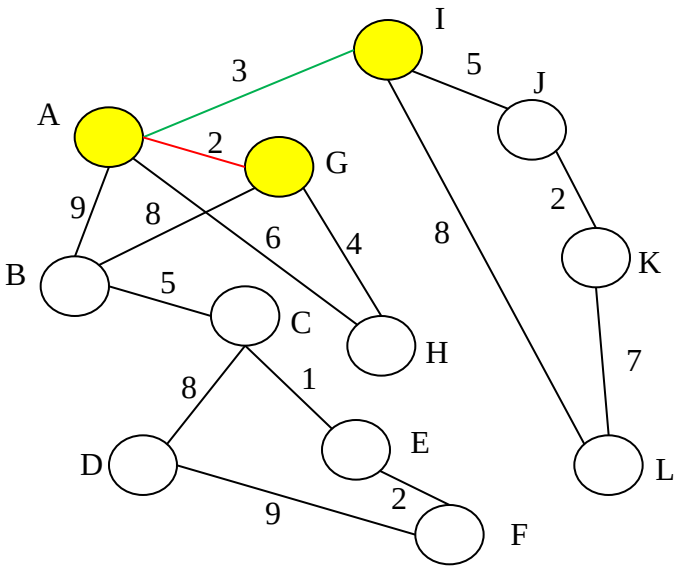
- Q: A(3), J(5), L(8), B(∞), C(∞), D(∞), E(∞), F(∞), G(∞), H(∞), K(∞) → A

Exemplu (II)



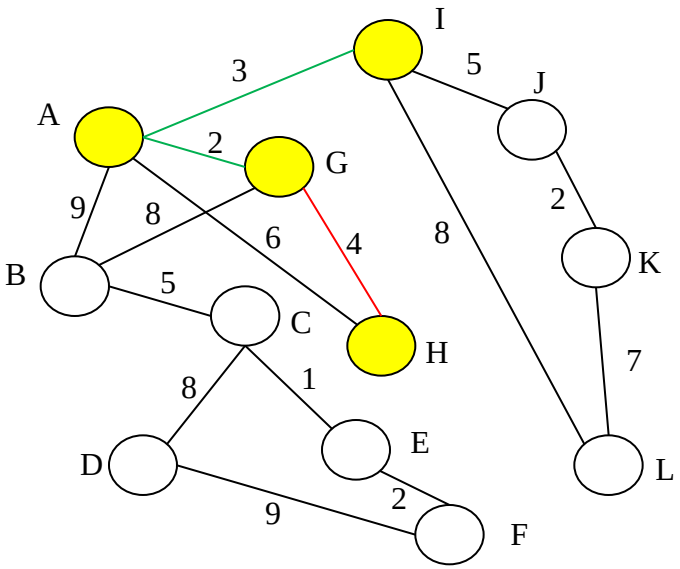
- Q: G(2), J(5), H(6), L(8), B(9), C(∞), D(∞), E(∞), F(∞), K(∞) → G

Exemplu (III)



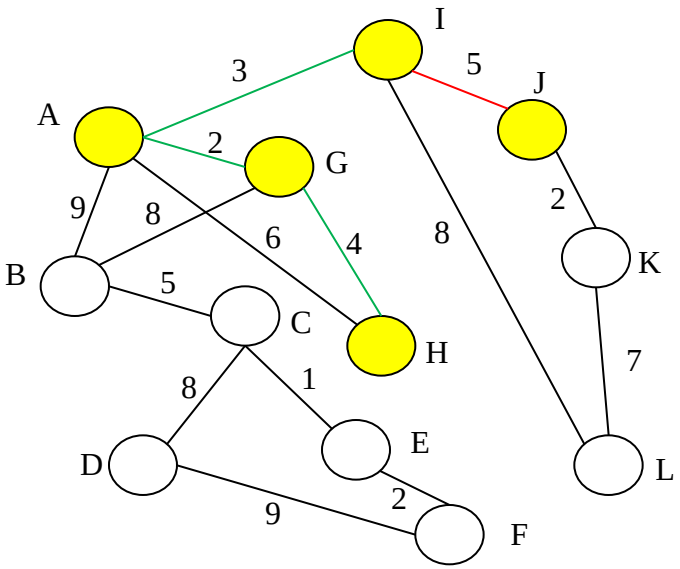
- Q: G(2), J(5), H(6), L(8), B(9), C(∞), D(∞), E(∞), F(∞), K(∞) → G
- Q: H(4), J(5), L(8), B(8), C(∞), D(∞), E(∞), F(∞), K(∞) → H

Exemplu (IV)



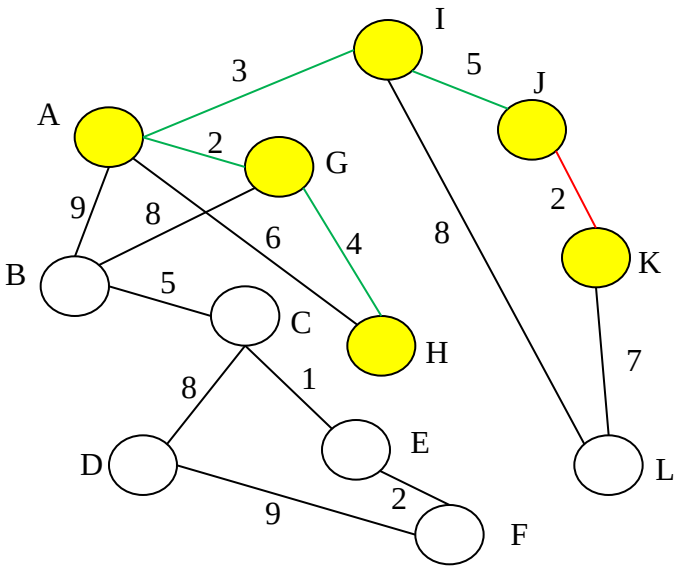
- Q: J(5), L(8), B(8), C(∞), D(∞), E(∞), F(∞), K(∞) → J

Exemplu (V)



- Q: K(2), L(8), B(8), C(∞), D(∞), E(∞), F(∞)
  → K

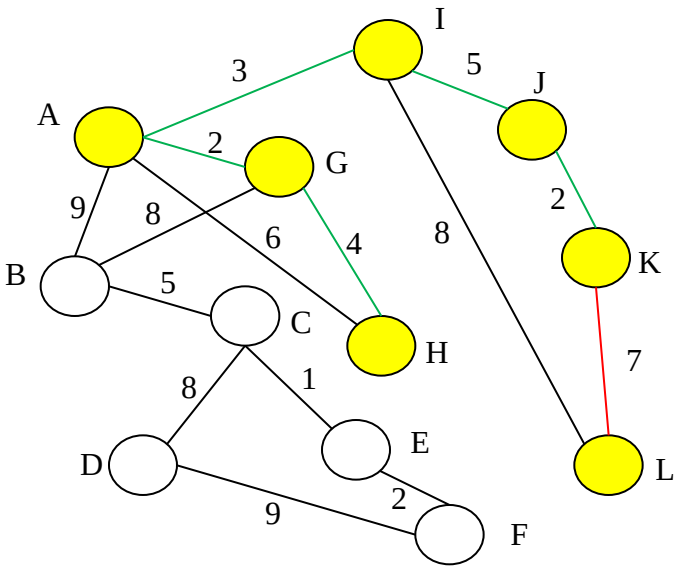
Exemplu (VI)



- Q: K(2), L(8), B(8), C(∞), D(∞), E(∞), F(∞) → K

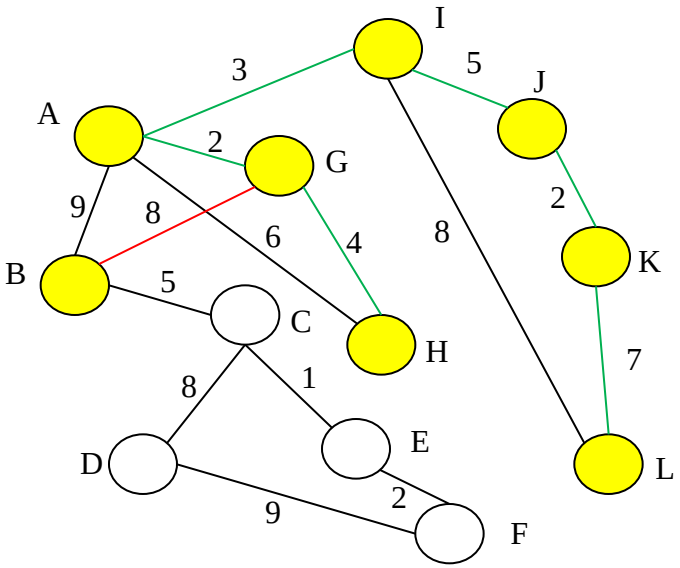
- Q: L(7), B(8), C(∞), D(∞), E(∞), F(∞) → L

Exemplu (VII)



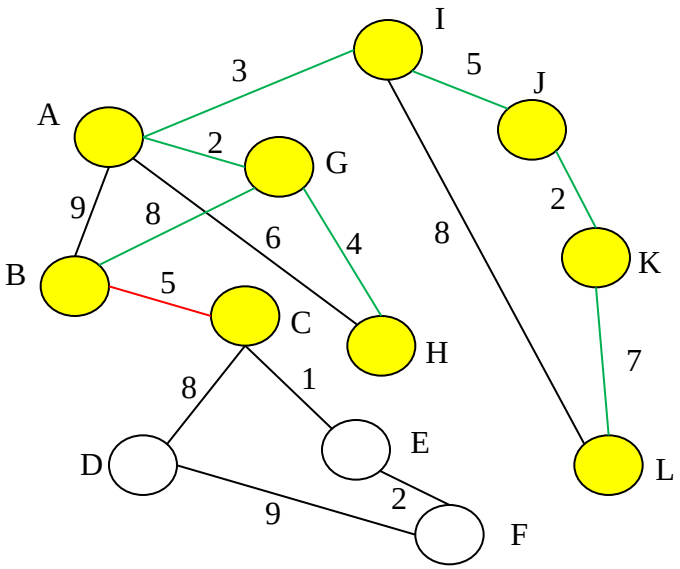
- Q: B(8), C(∞), D(∞), E(∞), F(∞) → B

Exemplu (VIII)



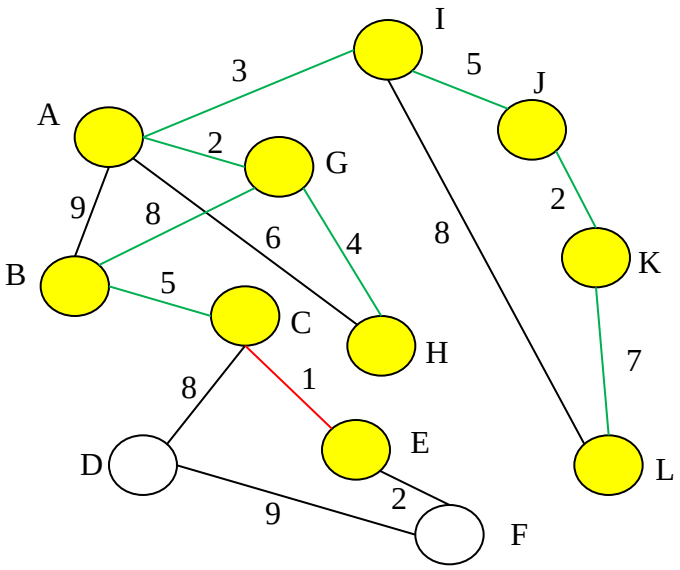
- Q: C(5), D(∞), E(∞), F(∞) → C

Exemplu (IX)



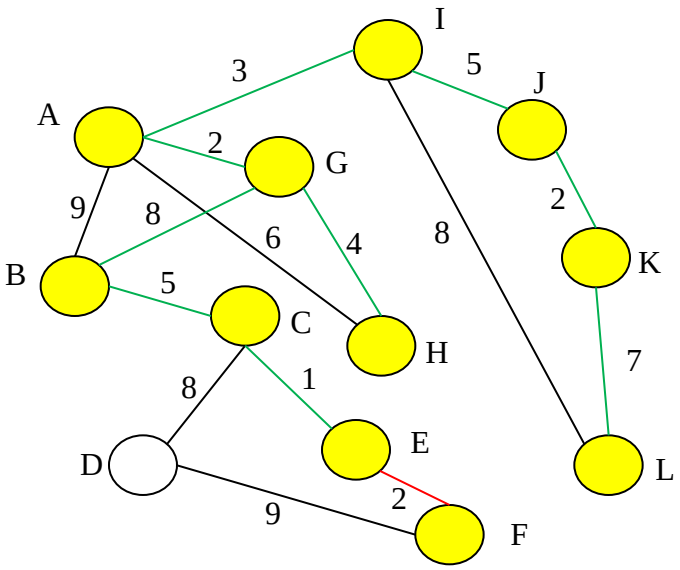
- Q: E(1), D(8), F(∞) → E

Exemplu (X)



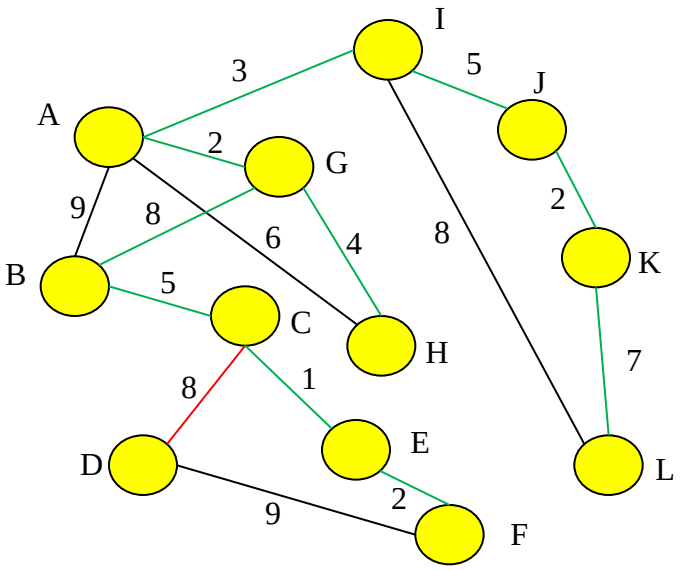
- Q: F(2), D(8) → F

Exemplu (XI)



- Q: D(8) → D

Exemplu (XII)



- Q: Ø

Corectitudine (I)

- 1. Arătăm că muchiile pe care le adăugăm aparțin Arb:
- Dem prin inducție după muchiile adăugate în AMA:
- P_1: avem V' = s, E' = ∅. Adaug muchia (u,s), u = nod adiacent sursei aflat cel mai aproape de aceasta → din Propr. 2 → (u,s) ∈ Arb.
- $P_n \rightarrow P_{n+1}$:
  - S = (V',E') mulțimea vârfurilor și muchiilor adăugate deja în arbore înainte de a adăuga (u,p[u]).
  - p[u] ∈ V', u ∉ V'; (u,p[u]) are cost minim dintre muchiile care au un capăt în S (conform extrage minim)
  - din Propr. 2 → (u,p[u]) ∈ Arb

Corectitudine (II)

- 2. arătăm că muchiile ignorate nu fac parte din Arb:
  - d[v] scade tot timpul de-a lungul algoritmului până când v este adăugat în AMA. În momentul adăugării, s-a găsit muchia de cost minim ce conectează nodul v la AMA;
  - Pp. (u,v) a.i. Arb(u) = Arb(v)
    - → (u,v) creează un ciclu în Arb(u) (arborii sunt aciclici) – fie ciclul format din u..x..v și (u,v).
    - w(u,v) = max {w(u',v') | (u',v') ∈ Arb(u)} DE CE?
      - Nodul u i-a fost adiacent nodului v, dar nu a fost ales la niciunul din momentele ulterioare de timp, când au fost parcurse muchiile din u..x..v → (u,v) are costul maxim din ciclu
    - → din Propr. 1 → (u,v) ∉ Arb

Complexitate Prim

- Prim(G,w,s)
  - A = ∅ // AMA
  - **Pentru fiecare** (u ∈ V)
    - d[u] = ∞; p[u] = null // inițializăm distanța și părintele
  - d[s] = 0; // nodul de start are distanța 0
  - Q = constrQ(V, d); // ordonată după costul muchiei
    // care unește nodul de AMA deja creat
  - **Cât timp** (Q != ∅) // cât timp mai sunt noduri neadăugate
    - u = ExtrMin(Q); // extrag nodul aflat cel mai aproape
    - A = A ∪ {(u,p[u])}; // adaug muchia în AMA
    - **Pentru fiecare** (v ∈ succs(u))
      - **Dacă** d[v] > w(u,v) **atunci**
        - d[v] = w(u,v); //+ d[u] // actualizăm distanțele și părinții nodurilor
        - p[v] = u; // adiacente care nu sunt în AMA încă
  - **Întoarce** A - {(s,p(s))} // prima muchie adăugată

Complexitate?

Algoritmul lui Dijkstra (II)

- Dijkstra(G,s)
  - **Pentru fiecare** (u ∈ V)
    - d[u] = ∞; p[u] = null;
  - d[s] = 0;
  - Q = construiește_coada(V) // coadă cu priorități
  - **Cât timp** (Q != ∅)
    - u = ExtrageMin(Q); // extrage din V elementul cu d[u] minim
    - // Q = Q - {u} – se execută în cadrul lui ExtrageMin
    - **Pentru fiecare** (v ∈ Q și v din succesorii lui u)
      - **Dacă** (d[v] > d[u] + w(u,v))
        - d[v] = d[u] + w(u,v) // actualizez distanța
        - p[v] = u // și părintele

Complexitate Prim

- Depinde de implementare (v. Dijkstra)
  - Matrice de adiacență $O(V^2)$
  - Heap binar $O(ElogV)$
  - Heap Fibonacci $O(VlogV+E)$

- Concluzii
  - Grafuri dese
    - Matrice de adiacență preferată
  - Grafuri rare
    - Heap binar sau Fibonacci