Structuri de date – Curs 12

Conf. univ. dr. Cristian CIUREA
Departamentul de Informatica si Cibernetica Economica
Academia de Studii Economice din Bucuresti
cristian.ciurea@ie.ase.ro

Agenda

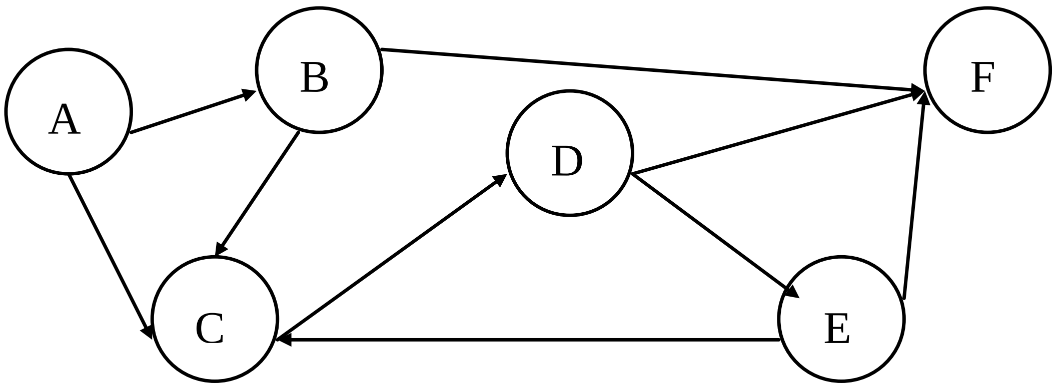
- Grafuri – închidere tranzitivă (algoritmul Roy-Floyd-Warshall)
- Grafuri – drum, lanț, ciclu, conexitate
- Sortarea topologică
- Algoritmul lui Dijkstra
- Algoritmul lui Prim

Grafuri

- Utilizarea grafurilor în rezolvarea problemelor de transport: **închiderea tranzitivă** a matricei de adiacenţă (pentru fiecare nod în parte arată unde se poate ajunge plecând din acesta).
- Modalitatea de creare a închiderii tranzitive: traversarea în adâncime a grafului din fiecare nod al său.
- Se obţin atâtea liste câte noduri sunt, liste care arată în ce noduri se ajunge din nodul de start.

Grafuri

- **închiderea tranzitivă:**



- pornind din A avem: A, C, D, F, E, B;
- pornind din B avem: B, F, C, D, E;
- pornind din C avem: C, D, F, E;
- pornind din D avem: D, F, E, C;
- pornind din E avem: E, F, C, D;
- pornind din F avem: F.

Grafuri

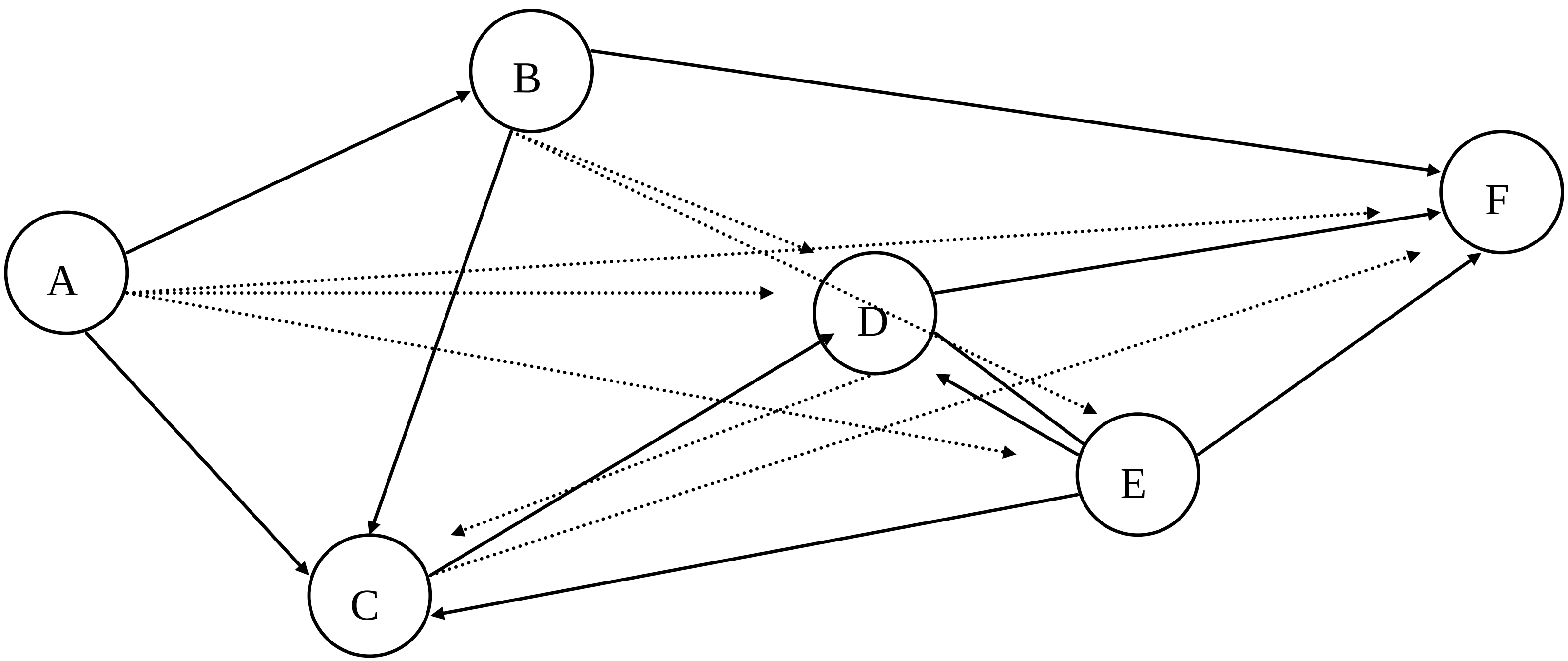
- Matricea închiderii tranzitive:

$$MIT = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Folosind matricea se obtine graful extins numit **închidere tranzitivă.**
- Pe reprezentarea grafului iniţial se desenează săgeţi punctate către nodurile la care se ajunge şi care nu sunt adiacente lui.

Grafuri

- **închiderea tranzitivă** a grafului:



Grafuri

- Algoritmul pentru închiderea tranzitivă este echivalent cu algoritmul **Roy-Floyd**, **Roy-Warshall**, respectiv **Floyd-Warshall** pentru determinarea matricei drumurilor minime dintr-un graf.

Grafuri

- **Drum**: Se numește drum într-un graf o succesiune de muchii adiacente și distincte care conectează două vârfuri din graf. Un drum se numește **simplu** dacă muchiile care îl compun sunt distincte.
- **Lanţ**: Se numeşte lanţ o listă de noduri de forma x_1, x_2, ..., x_n cu proprietatea că există arc între oricare (x_i, x_{i+1}). Daca lanţul respectă şi proprietatea (x_{i+1}, x_i), atunci lanţul este bidirecţional.

Grafuri

- **Ciclu**: Se numeşte ciclu o listă de noduri de forma x_1, x_2, ..., x_n, x_1 cu proprietatea că (x_i, x_{i+1}) şi (x_n, x_1). Un graf care nu conţine cicluri se numeşte graf aciclic.
- Un ciclu se numește **hamiltonian** dacă este simplu și trece prin toate nodurile grafului *G*, exact o dată, și se numește **eulerian** dacă trece prin toate muchiile grafului G, exact o dată.

Grafuri

- Se consideră că *G'* este un **subgraf** al lui *G*, dacă acesta conține o parte din vârfurile lui *G* și numai acele muchii care le conectează.
- **Conex**: Un graf se numeşte conex dacă între oricare două noduri din *G* există un lanţ.
- **Tare conex**: Un graf se numeşte tare conex dacă între oricare două noduri din *G* există un lanţ bidirecţional.

Grafuri

- **Componente conexe**: Un subgraf *G'* se numeşte componentă conexă a unui graf *G* dacă este conex şi nu există nici un lanţ între un nod din *G'* şi un nod din *G* neinclus în *G'*.
- **Componente tare conexe**: Un subgraf *G'* se numeşte componentă tare conexă a unui graf *G* dacă este tare conex maximal şi nu există nici un lanţ bidirecţional între un nod din *G'* şi un nod din *G* neinclus în *G'*.

Grafuri

- Orice vârf izolat este considerat componentă conexă.
- Dacă numărul componentelor conexe dintr-un graf este mai mare decât 1, atunci graful nu este conex.
- Un graf conex are o singură componentă conexă, care cuprinde toate nodurile sale.
- În teoria grafurilor, un graf conex este un graf neorientat în care există un drum între oricare două noduri distincte.

Grafuri

- Componentele tare conexe se pot determina folosind parcurgerea în adâncime şi **sortarea topologică.**
- Algoritmul are trei etape:
  - se sortează topologic graful *G*
  - se calculează graful transpus *G'* (conţine muchiile inversate)
  - se parcurge în adâncime graful *G'* considerând nodurile în ordinea indicată de sortarea topologică
- Arborii parțiali astfel obţinuţi constituie componentele tare conexe ale grafului.

Grafuri

- **Sortarea topologică** a unui graf orientat presupune găsirea unei ordonări a nodurilor astfel încât dacă graful conţine o muchie *(i, j)*, atunci *i* se va afla înaintea lui *j* în listă.
- Soluţia nu este unică; pentru aceasta trebuie impuse condiţii suplimentare.
- Sortarea topologică are multe aplicaţii practice în:
  - planificarea activităţilor;
  - proiectarea circuitelor logice;
  - proiectarea compilatoarelor.

Grafuri

- Se doreste sortarea numerelor *1, 2,…, n* numere ce se găsesc într-o anumită relaţie de ordine.
- Pentru a afla relaţia în care se găsesc elementele, se introduce un număr finit de perechi *(i, j)*.
- O astfel de pereche indică faptul că în relaţia de ordine considerată, *i* se află înaintea lui *j*.
- În funcţie de configuraţia perechilor prin care se dă relaţia, aceasta poate fi corectă sau nu, deci problema poate avea sau nu soluţie, iar dacă soluţia există, ea nu este neapărat unică.

Grafuri

- Pentru fiecare număr între *1* şi *n* trebuie să existe următoarele informaţii:
  - numărul predecesorilor săi
  - succesorii săi
- Pentru aceasta se folosesc doi vectori:
  - un vector contor, care reţine pentru fiecare *k*, *k=1,n* numărul predecesorilor săi
  - un vector ce reţine adresele de început ale listelor de succesori ai fiecărui element
- Iniţial, elementele vectorilor vor fi iniţializate cu 0, respectiv NULL, pentru că nu s-a citit nici o pereche

Grafuri

- Citirea unei perechi *(i, j)* determină efectuarea următoarelor operaţii:
  - se incrementează cu 1 contor(j), pentru că *i* este un predecesor în plus pentru *j*
  - se adaugă *j* în lista de succesori ai lui *i*
- Toate elementele din vectorul contor care au valoarea 0 se reţin într-un vector nou, iar valoarea 0 din vectorul contor iniţial se face -1, pentru ca ulterior elementul să nu mai fie luat în considerare

Grafuri

- Pentru fiecare element din vectorul nou se procedează astfel: pentru toţi succesorii săi aflaţi în lista de succesori se scade 1 din elementul corespunzător din vectorul contor, deoarece ulterior considerăm că aceştia au un predecesor mai puţin
- Se reia algoritmul dacă nu este îndeplinită una din următoarele condiţii:
  - au fost prelucrate toate elementele, caz în care s-a ajuns la o soluţie;
  - nu există nici un element 0 în vectorul contor, caz în care perechile de numere citite nu desemnează o relaţie de ordine coerentă, deci nu există soluţie.

Grafuri

- **Sortare topologica**

- Dati numarul de termeni :7
- Dati termenul 1 :5
- Dati termenul 2 :1
- Dati termenul 3 :3
- Dati termenul 4 :2
- Dati termenul 5 :4
- Dati termenul 6 :7
- Dati termenul 7 :6

- Dati numarul de relatii :4
- Relatia numarul 1 (a<b) :
- a = 2
- b = 3
- Relatia numarul 2 (a<b) :
- a = 4
- b = 5
- Relatia numarul 3 (a<b) :
- a = 6
- b = 7
- Relatia numarul 4 (a<b) :
- a = 1
- b = 2

- **Sortare topologica:**
- **1 2 3 4 5 6 7**

Grafuri

- Algoritmi de sortare:
  - sortarea prin numărare
  - sortarea prin selecţie directă
  - sortarea prin interschimbare (metoda bulelor)
  - sortarea prin inserţie
  - sortarea prin interclasare
  - sortarea rapidă (QuickSort)
  - sortarea prin distribuire
  - sortarea topologică

Grafuri

- Problema drumului de lungime minima în graf – **algoritmul Dijkstra**.
- În cazul grafului cu greutate se determină drumul ce are suma valorilor arcelor minimă, iar în cazul grafului fără greutate drumul care are cele mai puţine arce.
- Algoritmul Dijkstra examinează toate drumurile ce pornesc din nodul curent, actualizând distanțele dintre el si celelalte noduri.

Grafuri

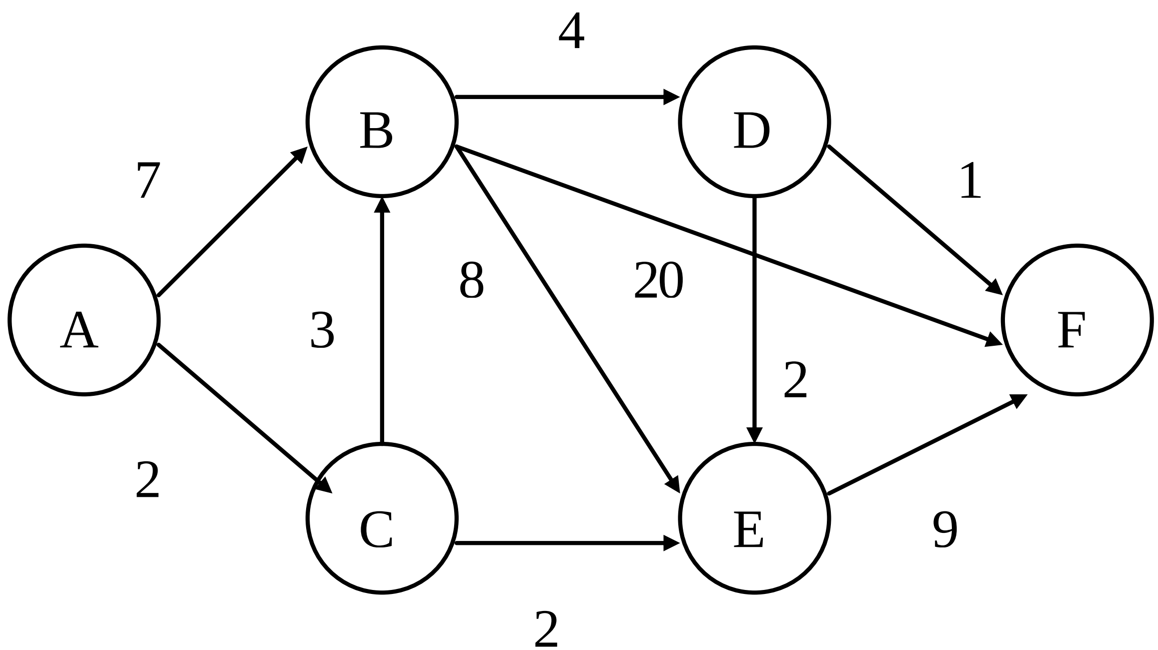
- Pentru a păstra nodurile prin care trece drumul cel mai scurt, programul le reține într-o listă.
- În final lista contine mulțimea minimă de noduri care să le conțină pe toate cele care vor forma efectiv drumul optim.
- Nodurile care se adaugă în listă sunt acele noduri ale grafului la care se ajunge prin arce directe doar de la nodurile din listă și care au lungimea cumulată până în acel moment minimă.
- Drumul minim este găsit în momentul în care în listă se află nodul destinaţie.

Grafuri

- Algoritmul lui Dijkstra este un algoritm de tip greedy care porneşte de la un graf conex orientat aciclic şi de la un nod sursă şi determină drumurile şi distanţele minime până la toate celelalte noduri.

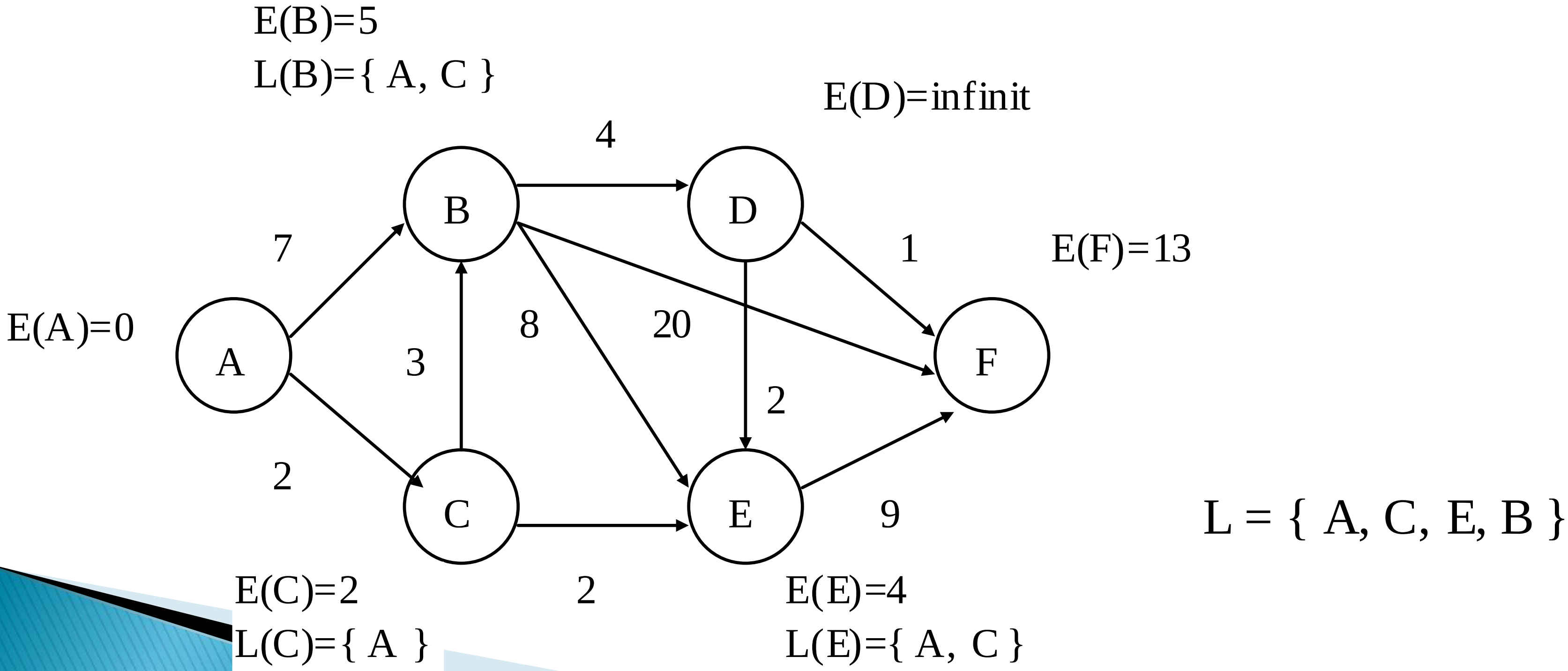
Grafuri

- algoritmul lui Dijkstra pentru calculul drumului minim de la nodul A la nodul F:



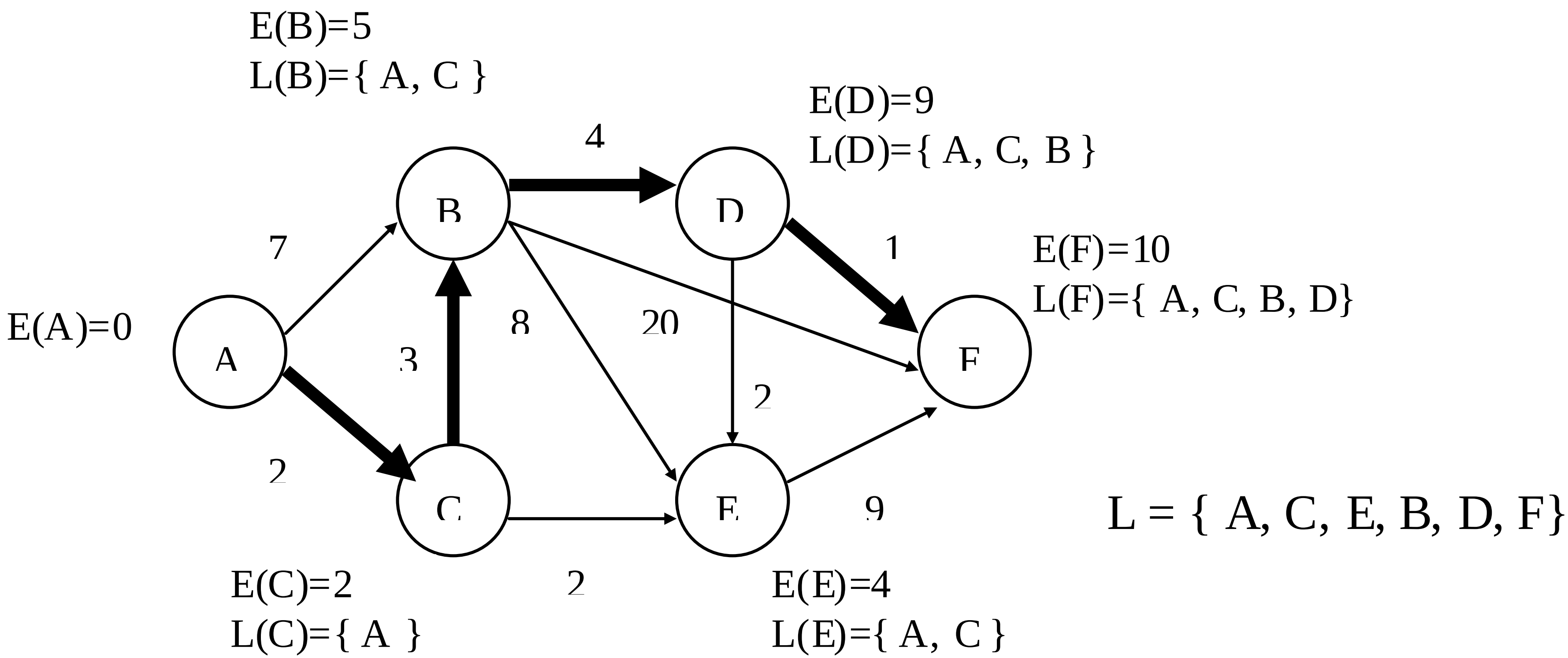
Grafuri

- Se fac notaţiile:
  - $E(N_i)$ – valoarea etichetei nodului N_i;
  - $L(N_i)$ – lista nodurilor prin care s-a ajuns la nodul N_i;
  - L – lista nodurilor care au fost luate în considerare.



Grafuri

- algoritmul lui Dijkstra:

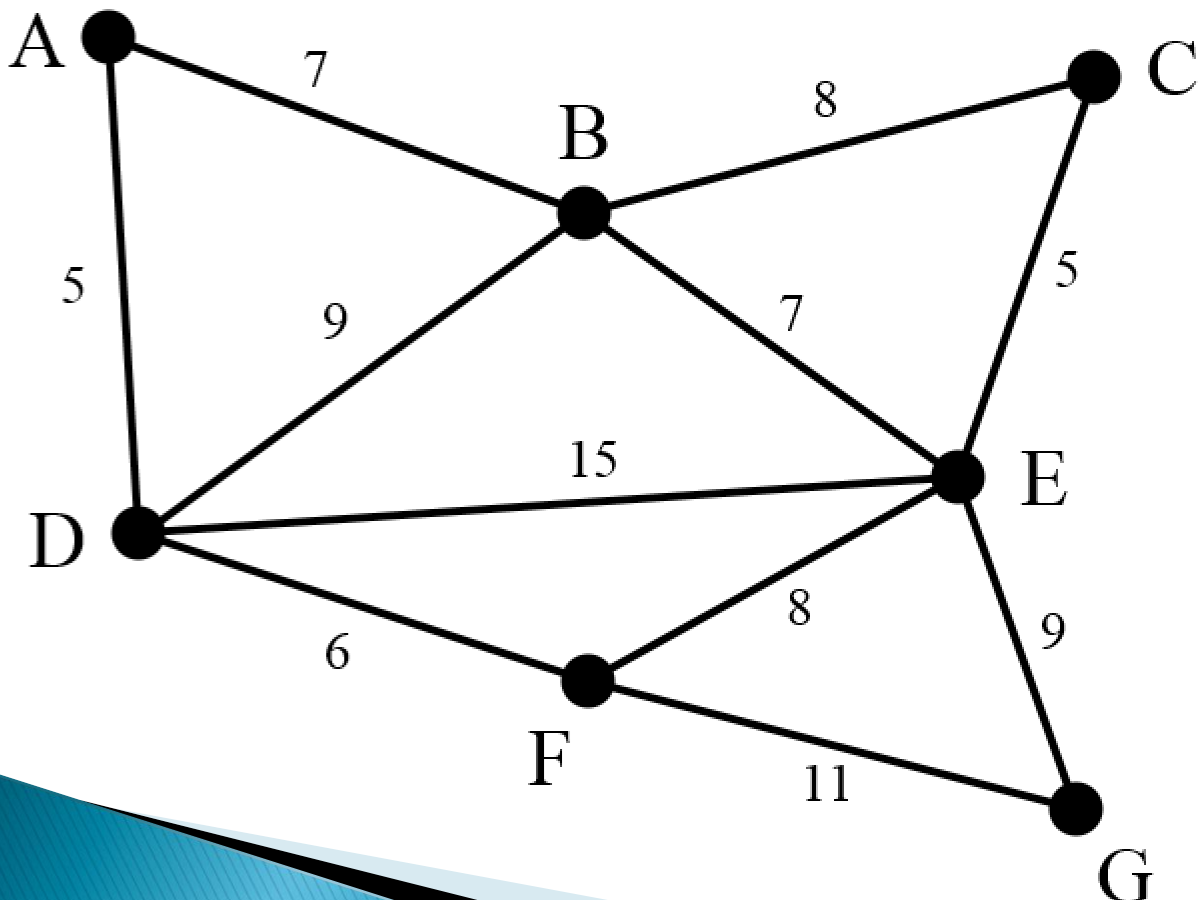


Grafuri

- **Algoritmul lui Prim** identifică arborele parțial de cost minim al unui graf conex ponderat.
- El găsește submulțimea muchiilor care formează un arbore ce include toate vârfurile și al cărui cost este minimizat.
- Mai este numit **algoritmul DJP**, **algoritmul Jarník** sau **algoritmul Prim-Jarník**.

Grafuri

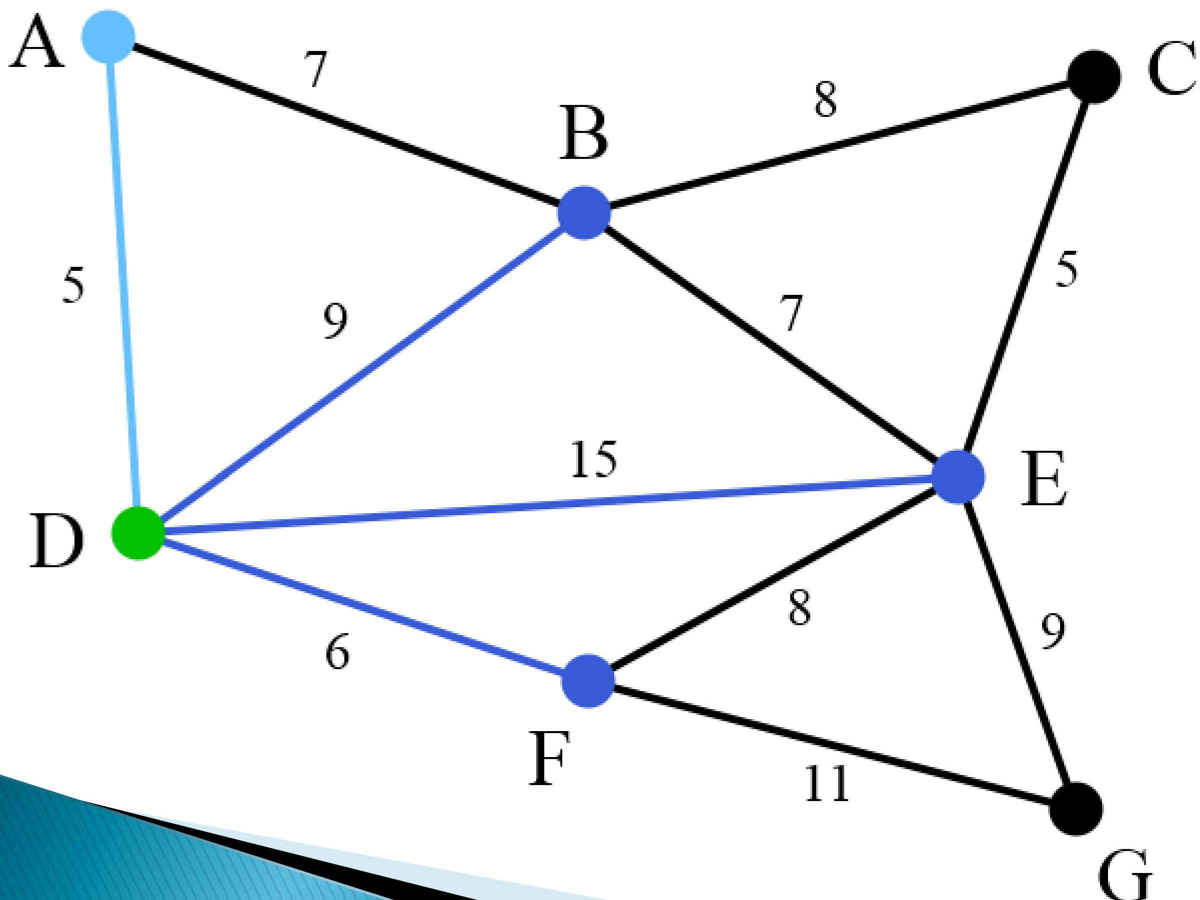
- algoritmul lui Prim:



- Soluția: D
- Vecini: A, B, E, F
- Nevizitați: C, G

Grafuri

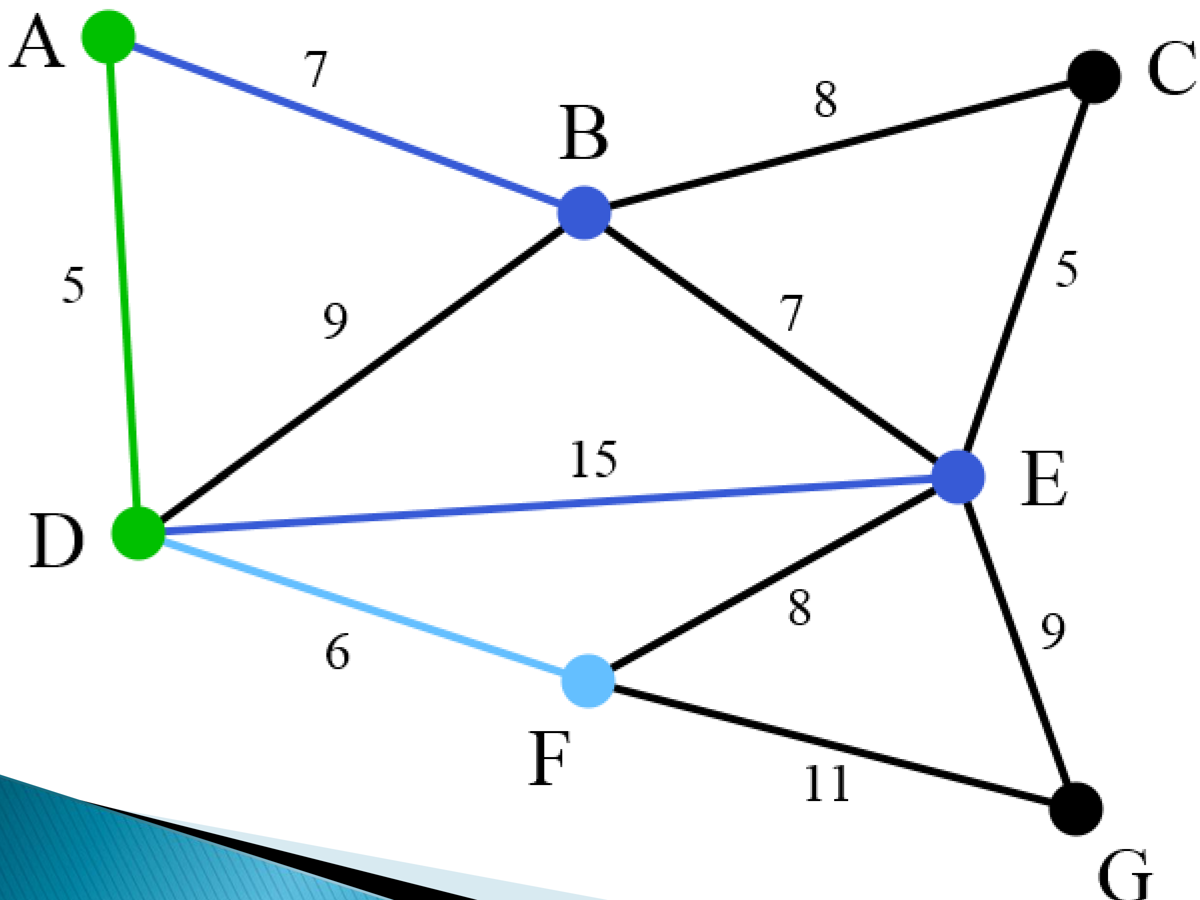
- algoritmul lui Prim:



- Soluția: A, D
- Vecini: B, E, F
- Nevizitați: C, G

Grafuri

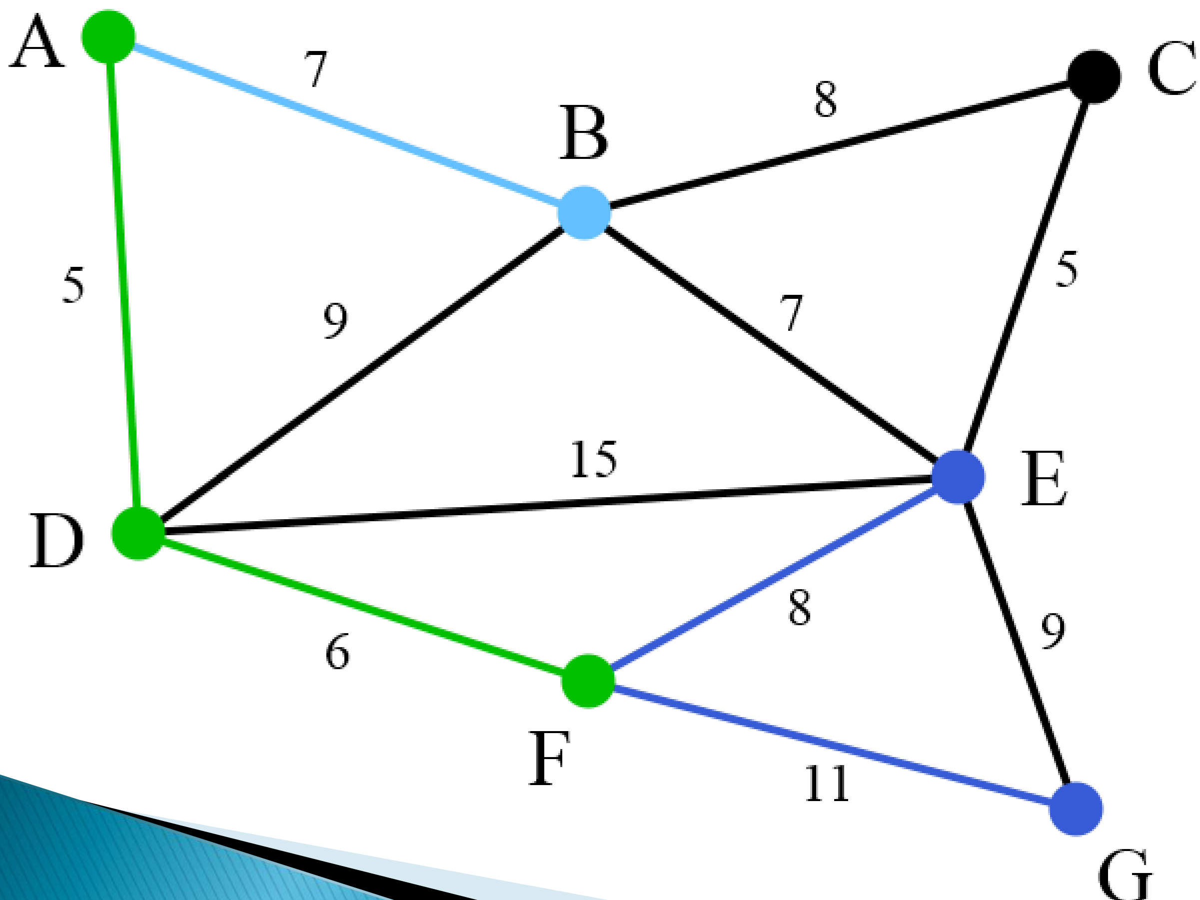
- algoritmul lui Prim:



- Soluția: A, D, F
- Vecini: B, E, G
- Nevizitați: C

Grafuri

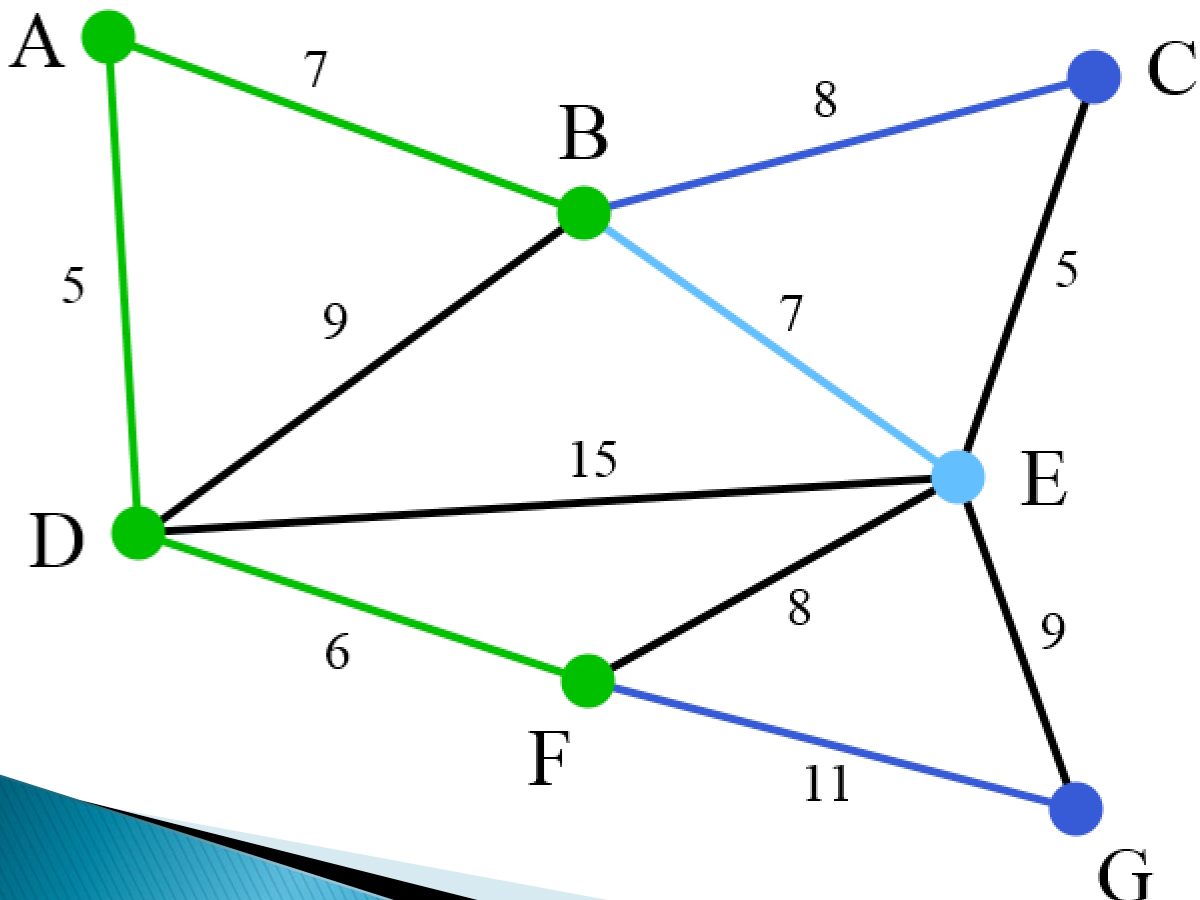
- algoritmul lui Prim:



- Soluția: A, D, F, B
- Vecini: C, E, G
- Nevizitați: NULL

Grafuri

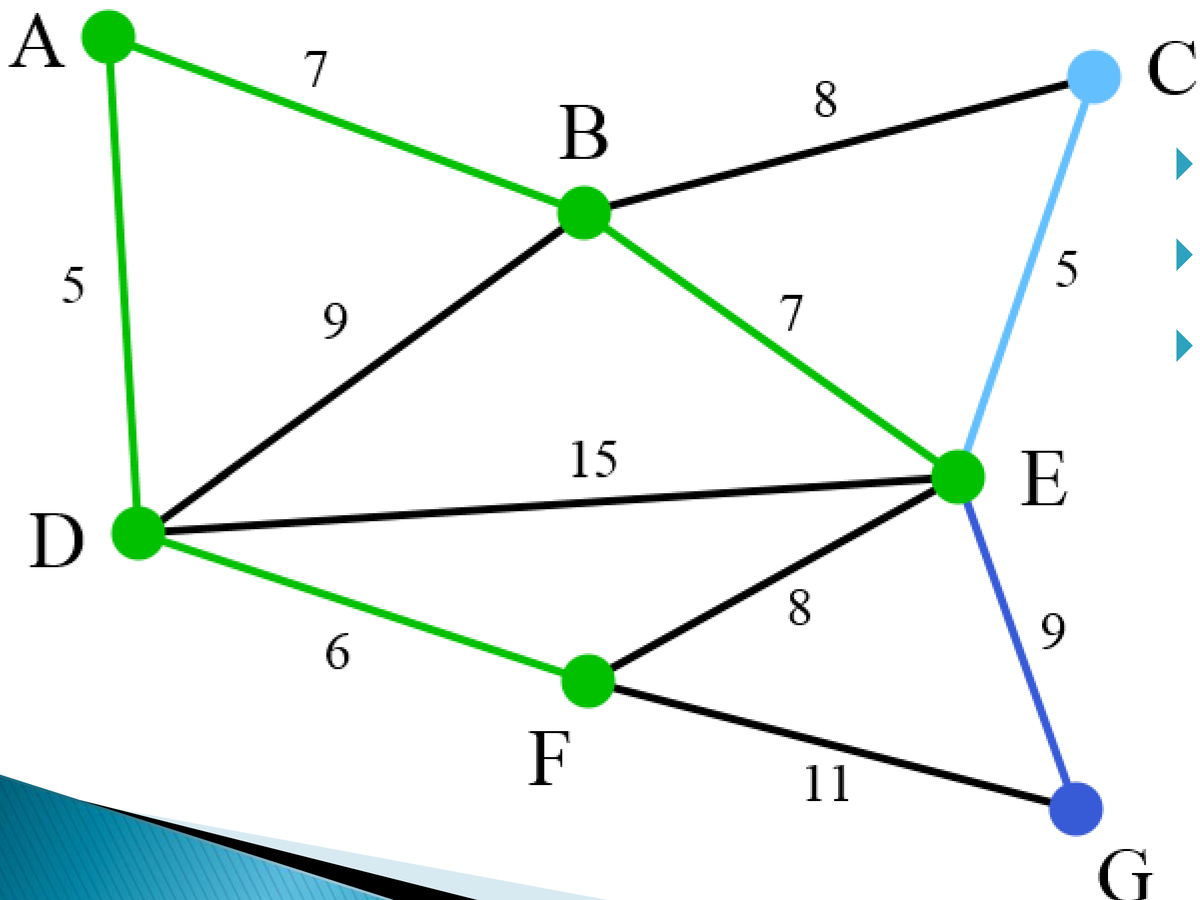
- algoritmul lui Prim:



- Soluția: A,D,F,B,E
- Vecini: C, G
- Nevizitați: NULL

Grafuri

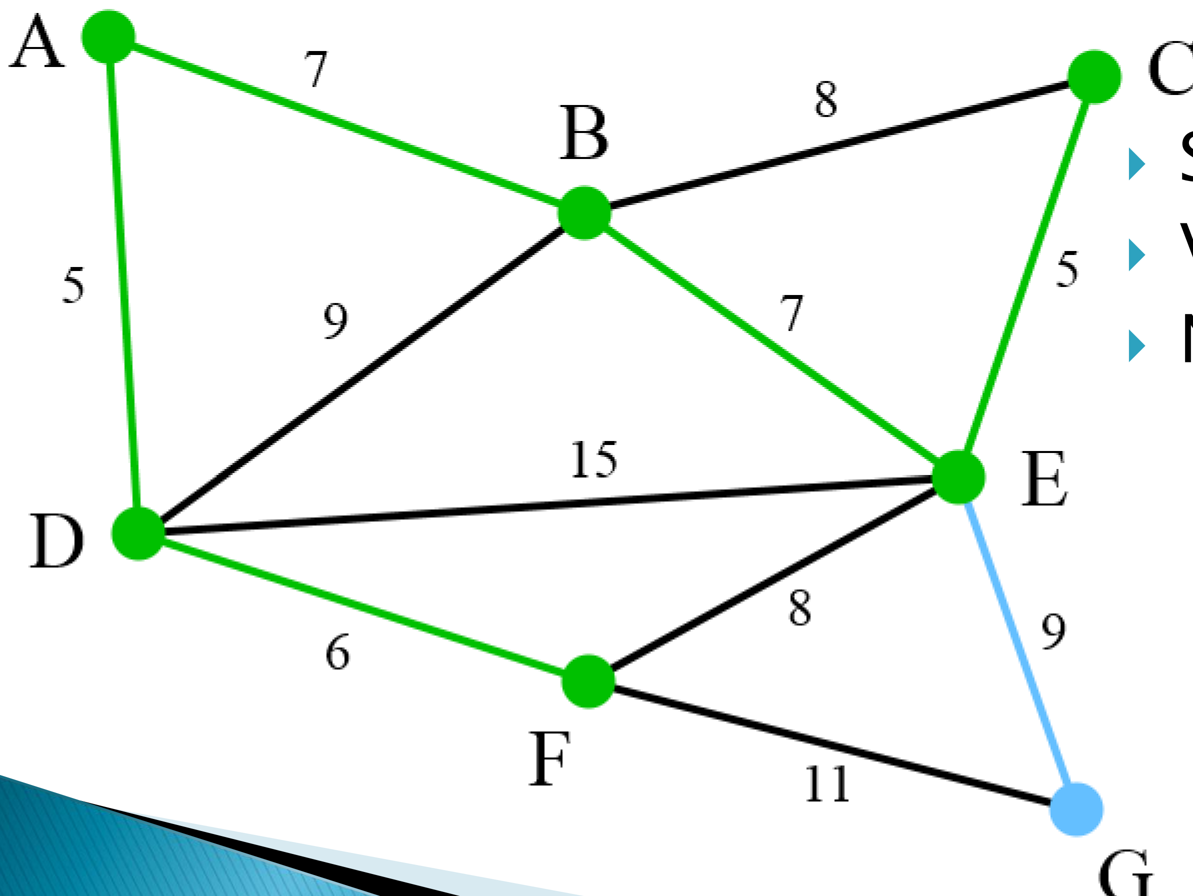
- algoritmul lui Prim:



- Soluția: A,D,F,B,E,C
- Vecini: G
- Nevizitați: NULL

Grafuri

- algoritmul lui Prim:



- Soluția: A,D,F,B,E,C,G
- Vecini: NULL
- Nevizitați: NULL

Bibliografie

- Ion Ivan, Marius Popa, Paul Pocatilu (coordonatori) – *Structuri de date*, Editura ASE, Bucureşti, 2008.
  - Cap. 16. Grafuri