Structuri de Date și Algoritmi
Grafuri orientate

Mihai Nan

Departamentul de Calculatoare
Facultatea de Automatică și Calculatoare
Universitatea POLITEHNICA din București

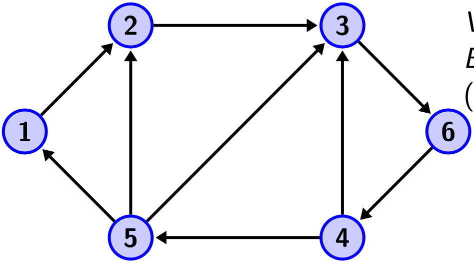
Anul Universitar 2022–2023

Conținutul cursului

Definiție

Un **graf orientat** este definit printr-o mulțime V de noduri și o mulțime E de legături între aceste noduri numite arce.
Din punct de vedere matematic, avem următoarea definiție:
$G = (V, E), E \subseteq V \times V, E = \{(u, v) | u, v \in V\}$

De data aceasta, perechea (u, v) este diferită de perechea (v, u).



$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{(1, 2), (2, 3), (3, 6), (4, 3)$
$(4, 5), (5, 1), (5, 2), (5, 3), (6, 4)\}$

Noțiuni elementare

- Gradul unui nod v este numărul de arce incidente cu acesta și îl notăm cu $grad(v)$.
- Gradul de intrare al unui nod dintr-un graf orientat este notat $ingrad(v)$ și reprezintă numărul arcelor care intră în acel nod.
- Gradul de ieșire al unui nod dintr-un graf orientat este notat $outgrad(v)$ și reprezintă numărul arcelor care ies din acel nod.
- Un nod dintr-un graf orientat este considerat izolat dacă $ingrad(v) = outgrad(v) = 0$.
- Fie un graf G simplu cu n noduri și m muchii / arce.
    1. Dacă G este neorientat, atunci

    $$m \leq \frac{n(n-1)}{2}$$

    2. Dacă G este orientat, atunci

    $$m \leq n(n-1)$$

Noțiuni elementare

- În graful orientat $G = (V, E)$, nodurile u și v sunt **adiacente** dacă există cel puțin un arc care le unește.
- Astfel, apar 3 cazuri posibile:
  1. Există numai arcul $(u, v) \in E$ – în acest caz spunem că arcul (u, v) este incident spre exterior cu u și spre interior cu v;
  2. Există numai arcul $(v, u) \in E$ – în acest caz spunem că arcul (v, u) este incident spre interior cu u și spre exterior cu v;
  3. Există arcul $(u, v) \in E$ dar și $(v, u) \in E$.

Fie un graf orientat G cu n vârfuri și m arce. Avem îndeplinită relația:

$$outgrad(v_1) + outgrad(v_2) + \cdots + outgrad(v_n) = m$$

$$ingrad(v_1) + ingrad(V_2) + \cdots + ingrad(v_n) = m$$

Noțiuni elementare

- Un **graf parțial** al unui graf orientat $G = (V, E)$ este un graf $G_1 = (V, E_1)$, unde $E_1 \subseteq E$.
- Un **subgraf** al unui graf orientat $G = (V, E)$ este un graf $G_1 = (V_1, E_1)$, unde $V_1 \subset V$ și $E_1 \subset E$, iar arcele din E_1 sunt **toate** arcele din E care sunt incidente numai la noduri din mulțimea V_1.
- Un graf orientat este **complet** dacă oricare două vârfuri, $i \neq j$, sunt adiacente.

Avem $3^{\frac{n(n-1)}{2}}$ grafuri complete cu n noduri.

- Un graf orientat este **turneu** dacă oricare ar fi două vârfuri i și j, $i \neq j$, între ele există un singur arc: arcul (i, j) sau arcul (j, i).

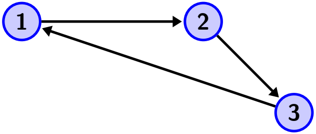
Noțiuni elementare

Proprietăți

1. Orice graf **turneu** este graf **complet**.
2. Avem $2^{\frac{n(n-1)}{2}}$ grafuri turneu cu n noduri.
3. În orice graf turneu există un drum elementar care trece prin toate nodurile grafului.

Graf tare conex

Graful orientat $G = (V, E)$ este **tare conex** dacă $\forall x, y \in V, \exists$ drum de la x la y și drum de la y la x.

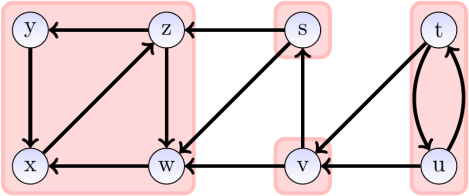


Noțiuni elementare

Componentă tare conexă

Subgraful $G_1 = (V_1, E_1)$ al grafului $G = (V, E)$ reprezintă o **componentă tare conexă** dacă:

1. $\forall x, y \in V_1, \exists$ un drum de la x la y și un drum de la y la x.
2. Nu există un alt subgraf al lui G, $G_2 = (V_2, E_2)$, cu $V_1 \subset V_2$ care îndeplinește condiția 1.



Modalități de reprezentare

Ce structură de date putem folosi pentru a reprezenta un graf orientat?

Modalități de reprezentare

Ce structură de date putem folosi pentru a reprezenta un graf orientat?

Putem folosi o matrice de adiacență!

Modalități de reprezentare

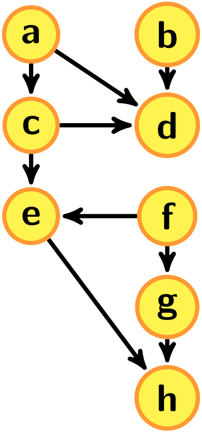
Ce structură de date putem folosi pentru a reprezenta un graf orientat?

Putem folosi o matrice de adiacență!

Putem folosi o reprezentare sub formă de liste de adiacență!

Algoritmul de sortare topologică

- O sortare topologică a vârfurilor unui **graf orientat aciclic** este o operație de ordonare liniară a vârfurilor, astfel încât, dacă există un arc (i, j), atunci i apare înaintea lui j în această ordonare.



Algoritmul de sortare topologică

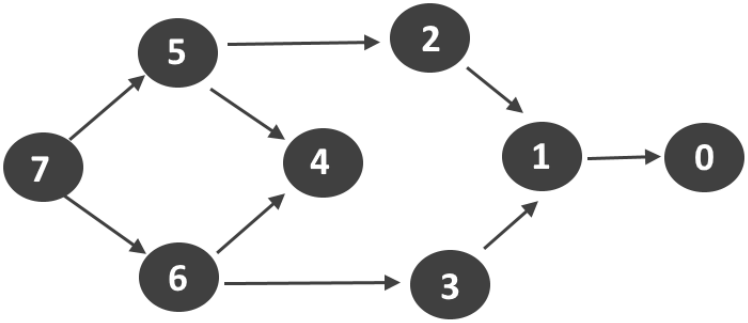
- Putem porni de la implementarea algoritmului de parcurgere DFS.
- Vom folosi varianta recursivă pentru care adăugăm un parametru suplimentar de tip stivă.
- În această stivă adăugăm nodurile în ordinea în care ele au fost expandate complet.

```
void recursiveDFS(Graph g, int start, Stack *topSort) {
    g->visited[start] = 1;
    List tmp = g->adjLists[start];
    while (tmp != NULL) {
        if (!g->visited[tmp->data])
            recursiveDFS(g, tmp->data, topSort);
            tmp = tmp->next;
    }
    *topSort = push(*topSort, start);
}
```

Algoritmul de sortare topologică

```c
void topologicalSort(Graph g) {
    Stack topSort = NULL;
    int i, current;
    for (i = 0; i < g->V; i++)
        g->visited[i] = 0;
    for (i = 0; i < g->V; i++)
        if (g->visited[i] == 0)
            recursiveDFS(g, i, &topSort);
    while(!isEmptyStack(topSort)) {
        current = top(topSort);
        topSort = pop(topSort);
        printf("%d ", current);
    }
    printf("\n");
}
```

Algoritmul de sortare topologică



Topological Sort : 7 6 5 4 3 2 1 0

Închiderea tranzitivă

- Ne interesează pentru anumite probleme închiderea tranzitivă a unui graf orientat.

Relație tranzitivă

Considerăm o relătie $R : A \times A$. Spunem că R este **tranzitivă** dacă:

- $\forall x, y, z \in A$, dacă xRy și yRz atunci xRz.

- Închiderea tranzitivă a relației R
  $R^k = R \cdot R \cdot \dots \cdot R$ de k ori până când $R^{k+1} = R^k$.
- Pentru un graf orientat se poate realiza prin DFS în:
  1. $O(|V| \cdot (|E| + |V|))$ pentru grafuri rare
  2. $O(V^3)$ pentru grafuri dense

Algoritmul lui Warshall

- Pentru fiecare pereche de noduri (x, y) astfel încât y este accesibil din x, adăugăm (dacă nu există) un arc $x \to y$.
- Pentru algoritmul lui Warshall vom prefera reprezentarea grafurilor prin matrice de adiacență.
- Dacă există un k astfel încât $x \to k$ și $k \to y$ atunci $x \to y$.

Dacă există un drum de la x la y și un drum de la y la j atunci se poate ajunge de la x la j.

- Găsește închiderea tranzitivă în $O(V^3)$.

Algoritmul lui Warshall

Cum putem converti un graf reprezentat prin liste de adiacență în varianta cu matrice de adiacență?

Algoritmul lui Warshall

Cum putem converti un graf reprezentat prin liste de adiacență în varianta cu matrice de adiacență?

```c
void Warshall(Graph g) {
    int i, **mat;
    mat = malloc(g->V * sizeof(int*));
    for (i = 0; i < g->V; i++) {
        mat[i] = calloc(g->V, sizeof(int));
        List tmp = g->adjLists[i];
        while (tmp != NULL) {
            mat[i][tmp->data] = 1;
            tmp = tmp->next;
        }
    }
```

Algoritmul lui Warshall

Dacă există un drum de la x la y și un drum de la y la j atunci se poate ajunge de la x la j.

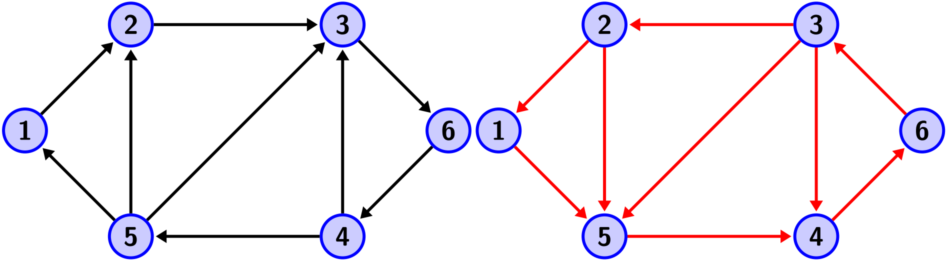
```c
int y, x, j;
for (y = 0; y < g->V; y++) {
    for (x = 0; x < g->V; x++) {
        if (mat[x][y])
            for (j = 0; j < g->V; j++) {
                if (mat[y][j])
                    mat[x][j] = 1;
            }
    }
}
```

Algoritmul lui Warshall

```
    for (x = 0; x < g->V; x++) {
        for (y = 0; y < g->V; y++) {
            printf("%d ", mat[x][y]);
        }
        free(mat[x]);
        printf("\n");
    }
    free(mat);
}
```

Determinarea componentelor tare conexe

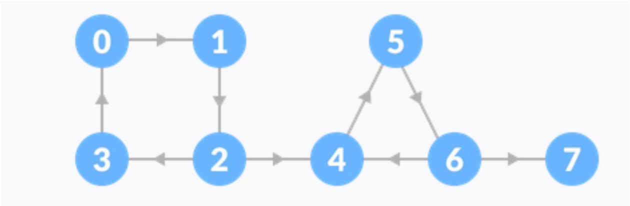
1. Aplicăm o parcurgere DFS pe graf și reținem nodurile într-o stivă pe măsură ce ele sunt expandate complet.
2. Determinăm graful transpus.



3. Extragem nodurile din stivă unul câte unul și aplicăm algoritmul de parcurgere DFS pe graful transpus.

Determinarea componentelor tare conexe

Exemplu prezentat la tablă!



Vă mulțumesc pentru atenție!