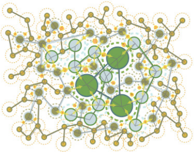
Графи

Трифон Трифонов

Структури от данни и програмиране, спец. Компютърни науки, 2 поток, 2024/25 г.

19 януари 2023 г.



Базирано на "Socialized Energy (AKA Smart Grid)" от Zachary Veach (`https://flic.kr/p/6rTacN`), CC BY-NC-SA 2.0

Дефиниция на граф

Дефиниция (Граф)

(Ориентиран) граф е наредена двойка (V, E), където

- $V \neq \emptyset$ е произволно множество от **върхове**
- $E \subseteq V^2$ е множество от наредени двойки върхове — **ребра**

Дефиниция на граф

Дефиниция (Граф)

(Ориентиран) граф е наредена двойка (V, E), където

- $V \neq \emptyset$ е произволно множество от **върхове**
- $E \subseteq V^2$ е множество от наредени двойки върхове — **ребра**

Ако пренебрегнем реда на компонентите в двойките в E, получаваме **неориентиран** граф.

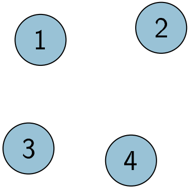
Дефиниция на граф

Дефиниция (Граф)

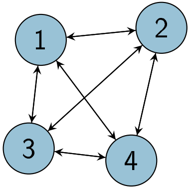
(Ориентиран) граф е наредена двойка (V, E), където

- $V \neq \emptyset$ е произволно множество от **върхове**
- $E \subseteq V^2$ е множество от наредени двойки върхове — **ребра**

Ако пренебрегнем реда на компонентите в двойките в E, получаваме **неориентиран** граф.



$E = \emptyset$ — празен граф



$E = V^2$ — пълен граф

АТД: граф

Нелинейна структура, описваща обекти и връзките между тях.

Операции:

- `vertices()` — списък на върховете
- `successors(u)` — списък на наследниците на даден връх
- `isEdge(u, v)` — проверка за съществуване на ребро
- `addVertex(u)` — включване на връх
- `removeVertex(u)` — изключване на връх
- `addEdge(u, v)` — включване на ребро
- `removeEdge(u, v)` — изключване на ребро

Етикети

Обектите и връзките в графа могат да бъдат свързани с етикети.

Етикети

Обектите и връзките в графа могат да бъдат свързани с етикети.

Нека е дадено множество L от етикети.

- $v : V \to L$ — етикети на върховете
- $e : E \to L$ — етикети на ребрата

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \ldots, v_n$, където $(v_i, v_{i+1}) \in E$

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \ldots, v_n$, където $(v_i, v_{i+1}) \in E$
  - ако $v_1 = v_n$, пътят е **цикъл**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \dots, v_n$, където $(v_i, v_{i+1}) \in E$
    - ако $v_1 = v_n$, пътят е **цикъл**
    - ако $v_i \neq v_j$ за $1 \leq i < j \leq n$, пътят е **ацикличен**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \ldots, v_n$, където $(v_i, v_{i+1}) \in E$
    - ако $v_1 = v_n$, пътят е **цикъл**
    - ако $v_i \neq v_j$ за $1 \leq i < j \leq n$, пътят е **ацикличен**
    - ако $E = \{(v_i, v_{i+1}) \mid 1 \leq i < j \leq n\}$ и $|E| = n - 1$, пътят е **Ойлеров**

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \ldots, v_n$, където $(v_i, v_{i+1}) \in E$
  - ако $v_1 = v_n$, пътят е **цикъл**
  - ако $v_i \neq v_j$ за $1 \leq i < j \leq n$, пътят е **ацикличен**
  - ако $E = \{(v_i, v_{i+1}) \mid 1 \leq i < j \leq n\}$ и $|E| = n - 1$, пътят е **Ойлеров**
  - ако $V = \{v_i \mid 1 \leq i \leq n\}$ и $|V| = n$, пътят е **Хамилтонов**

Допълнителни дефиниции

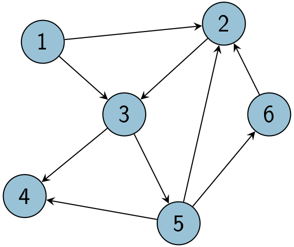
- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \ldots, v_n$, където $(v_i, v_{i+1}) \in E$
  - ако $v_1 = v_n$, пътят е **цикъл**
  - ако $v_i \neq v_j$ за $1 \leq i < j \leq n$, пътят е **ацикличен**
  - ако $E = \{(v_i, v_{i+1}) \mid 1 \leq i < j \leq n\}$ и $|E| = n - 1$, пътят е **Ойлеров**
  - ако $V = \{v_i \mid 1 \leq i \leq n\}$ и $|V| = n$, пътят е **Хамилтонов**
- граф е **цикличен**, ако в него има поне един цикъл

Допълнителни дефиниции

- за $(u, v) \in E$, u наричаме **предшественик**, а v — **наследник**
- (u, u) наричаме **примка**
- $d^+(u) = |\{v \mid (u, v) \in E\}|$ — **положителна полустепен**
- $d^-(v) = |\{u \mid (u, v) \in E\}|$ — **отрицателна полустепен**
- $d(u) = d^+(u) + d^-(u)$ — **степен на връх**
- **път** в граф наричаме редица $v_1, v_2, \ldots, v_n$, където $(v_i, v_{i+1}) \in E$
    - ако $v_1 = v_n$, пътят е **цикъл**
    - ако $v_i \neq v_j$ за $1 \leq i < j \leq n$, пътят е **ацикличен**
    - ако $E = \{(v_i, v_{i+1}) \mid 1 \leq i < j \leq n\}$ и $|E| = n - 1$, пътят е **Ойлеров**
    - ако $V = \{v_i \mid 1 \leq i \leq n\}$ и $|V| = n$, пътят е **Хамилтонов**
- граф е **цикличен**, ако в него има поне един цикъл
- граф е **(слабо) свързан**, ако $\forall a, b \in V$ има път от a до b (или от b до a)

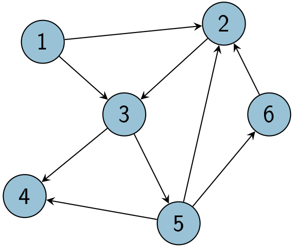
Матрица на съседство

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Матрица на съседство

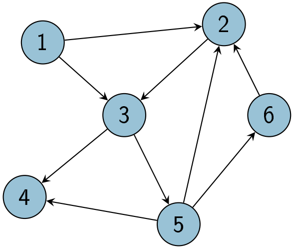
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$

Матрица на съседство

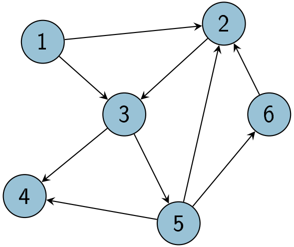
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — ?

Матрица на съседство

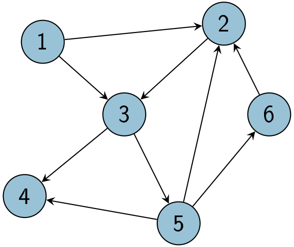
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$

Матрица на съседство

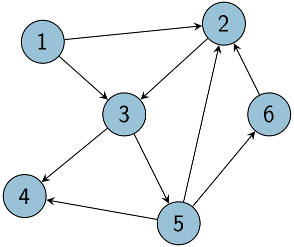
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — ?

Матрица на съседство

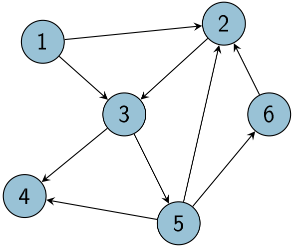
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$

Матрица на съседство

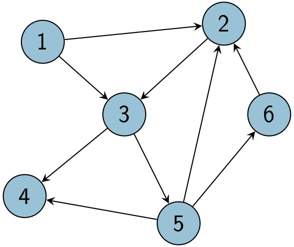
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — ?

Матрица на съседство

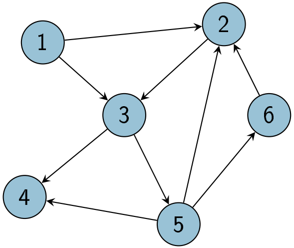
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$

Матрица на съседство

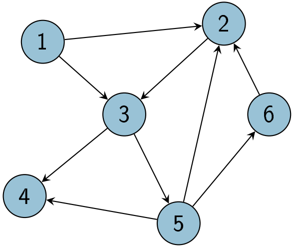
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$
- `addVertex` — ?

Матрица на съседство

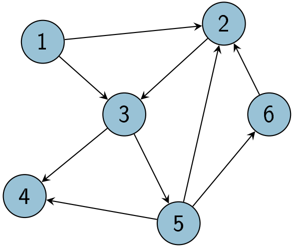
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$
- `addVertex` — $O(|V|)$

Матрица на съседство

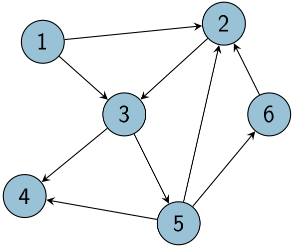
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$
- `addVertex` — $O(|V|)$
- `removeVertex` — ?

Матрица на съседство

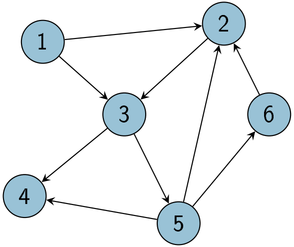
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$
- `addVertex` — $O(|V|)$
- `removeVertex` — $O(|V|^2)$, ако се налага разместване

Матрица на съседство

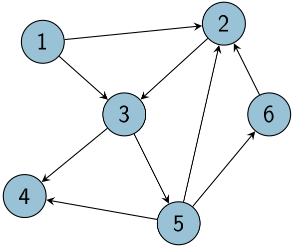
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$
- `addVertex` — $O(|V|)$
- `removeVertex` — $O(|V|^2)$, ако се налага разместване
- `addEdge`, `removeEdge` — ?

Матрица на съседство

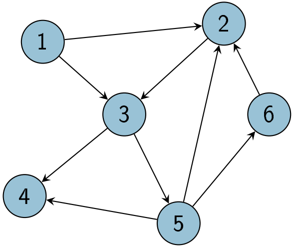
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- памет — $O(|V|^2)$
- `successors` — $O(|V|)$
- `isEdge` — $O(1)$
- `addVertex` — $O(|V|)$
- `removeVertex` — $O(|V|^2)$, ако се налага разместване
- `addEdge`, `removeEdge` — $O(1)$

Матрица на съседство

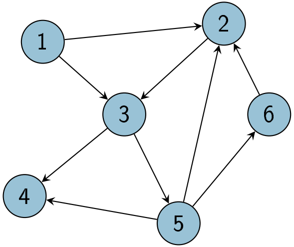
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$

Матрица на съседство

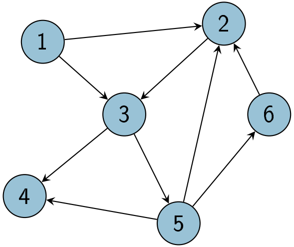
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- $A^2_{i,j} > 0 \leftrightarrow \sum_{k=1}^{n} A_{i,k} A_{k,j} > 0 \leftrightarrow$ има път с дължина 2 от v_i до v_j

Матрица на съседство

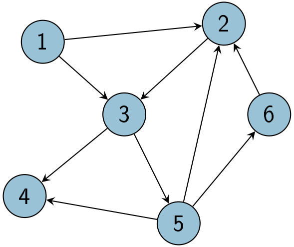
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- $A^2_{i,j} > 0 \leftrightarrow \sum_{k=1}^{n} A_{i,k} A_{k,j} > 0 \leftrightarrow$ има път с дължина 2 от v_i до v_j
- по индукция: $A^n_{i,j} > 0 \leftrightarrow$ има път с дължина n от v_i до v_j

Матрица на съседство

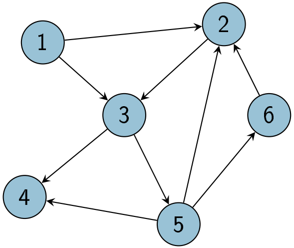
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- $A^2_{i,j} > 0 \leftrightarrow \sum_{k=1}^{n} A_{i,k} A_{k,j} > 0 \leftrightarrow$ има път с дължина 2 от v_i до v_j
- по индукция: $A^n_{i,j} > 0 \leftrightarrow$ има път с дължина n от v_i до v_j
- нещо повече: $A^n_{i,j} =$ броят на пътищата с дължина n от v_i до v_j

Матрица на съседство

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

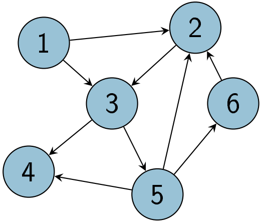


- $A_{i,j} = 1 \leftrightarrow (v_i, v_j) \in E$
- $A^2_{i,j} > 0 \leftrightarrow \sum_{k=1}^{n} A_{i,k}A_{k,j} > 0 \leftrightarrow$ има път с дължина 2 от v_i до v_j
- по индукция: $A^n_{i,j} > 0 \leftrightarrow$ има път с дължина n от v_i до v_j
- нещо повече: $A^n_{i,j} =$ броят на пътищата с дължина n от v_i до v_j
- ако $B = \sum_{k=1}^{|V|} A^k$, то $B_{i,j} > 0 \leftrightarrow$ има път от v_i до v_j

Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

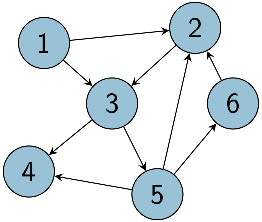
- $A_{i,j} = 1 \leftrightarrow \exists v \in V\ e_j = (v_i, v) \leftrightarrow$ e_j е **изходящо** ребро за v_i
- $A_{i,j} = -1 \leftrightarrow \exists v \in V\ e_j = (v, v_i) \leftrightarrow$ e_j е **входящо** ребро за v_i



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

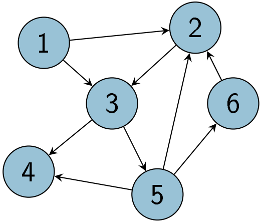
- памет — $O(|V||E|)$



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

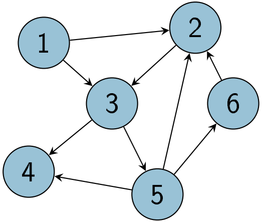
- памет — $O(|V||E|)$
- `successors` — ?



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

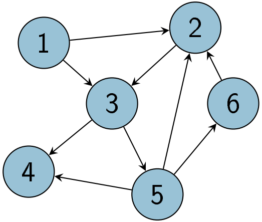
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

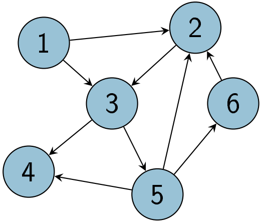
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — ?



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

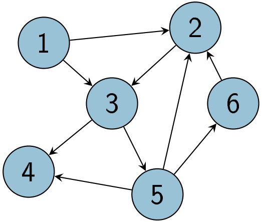
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

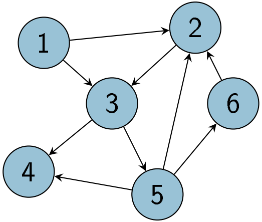
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$
- `addVertex` — ?



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

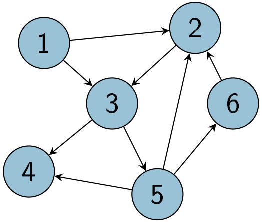
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$
- `addVertex` — $O(|E|)$



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

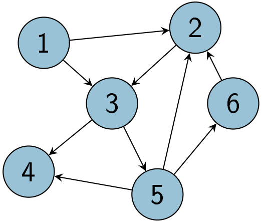
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$
- `addVertex` — $O(|E|)$
- `addEdge` — ?



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

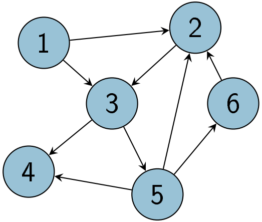
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$
- `addVertex` — $O(|E|)$
- `addEdge` — $O(|V|)$



Матрица на инцидентност

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

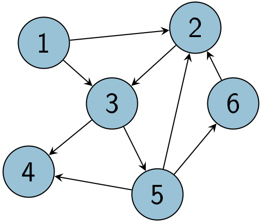
- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$
- `addVertex` — $O(|E|)$
- `addEdge` — $O(|V|)$
- `removeVertex`, `removeEdge` — ?



Матрица на инцидентност

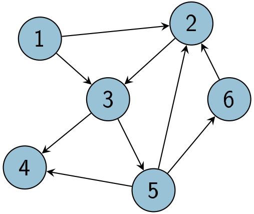
$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

- памет — $O(|V||E|)$
- `successors` — $O(|V||E|)$
- `isEdge` — $O(|E|)$
- `addVertex` — $O(|E|)$
- `addEdge` — $O(|V|)$
- `removeVertex`, `removeEdge` — $O(|V||E|)$, ако се налага разместване



Списък на наследници

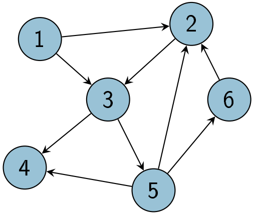
$$D = \left\{ \begin{array}{lcl} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък

Списък на наследници

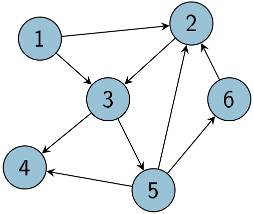
$$D = \left\{ \begin{array}{lll} 1 & \rightarrow & (2,3) \\ 2 & \rightarrow & (3) \\ 3 & \rightarrow & (4,5) \\ 4 & \rightarrow & () \\ 5 & \rightarrow & (2,4,6) \\ 6 & \rightarrow & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — ?

Списък на наследници

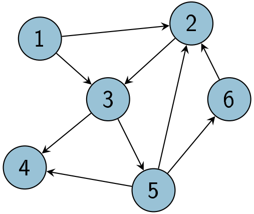
$$D = \left\{ \begin{array}{lll} 1 & \rightarrow & (2,3) \\ 2 & \rightarrow & (3) \\ 3 & \rightarrow & (4,5) \\ 4 & \rightarrow & () \\ 5 & \rightarrow & (2,4,6) \\ 6 & \rightarrow & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$

Списък на наследници

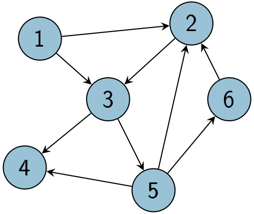
$$D = \left\{ \begin{array}{ccc} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — ?

Списък на наследници

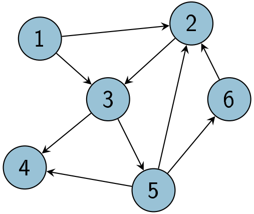
$$D = \left\{ \begin{array}{lll} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$

Списък на наследници

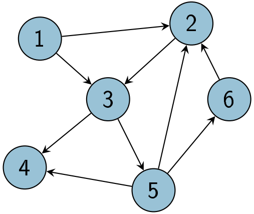
$$D = \left\{ \begin{array}{lcl} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — ?

Списък на наследници

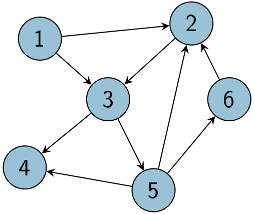
$$D = \left\{ \begin{array}{ccl} 1 & \rightarrow & (2,3) \\ 2 & \rightarrow & (3) \\ 3 & \rightarrow & (4,5) \\ 4 & \rightarrow & () \\ 5 & \rightarrow & (2,4,6) \\ 6 & \rightarrow & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)

Списък на наследници

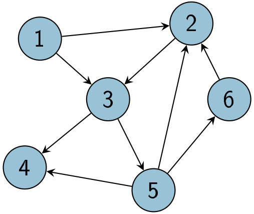
$$D = \left\{ \begin{array}{ccc} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — ?

Списък на наследници

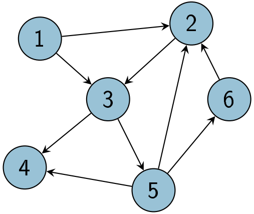
$$D = \left\{ \begin{array}{ccc} 1 & \rightarrow & (2,3) \\ 2 & \rightarrow & (3) \\ 3 & \rightarrow & (4,5) \\ 4 & \rightarrow & () \\ 5 & \rightarrow & (2,4,6) \\ 6 & \rightarrow & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$

Списък на наследници

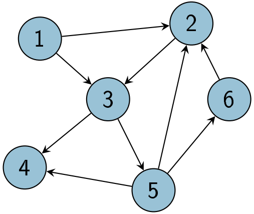
$$D = \left\{ \begin{array}{lcl} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$
- `removeVertex` — ?

Списък на наследници

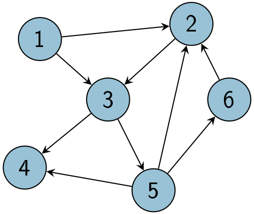
$$D = \left\{ \begin{array}{lcl} 1 & \rightarrow & (2,3) \\ 2 & \rightarrow & (3) \\ 3 & \rightarrow & (4,5) \\ 4 & \rightarrow & () \\ 5 & \rightarrow & (2,4,6) \\ 6 & \rightarrow & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$
- `removeVertex` — $O(|E|)$, трябва да се премахнат входящите ребра

Списък на наследници

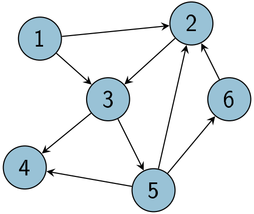
$$D = \left\{ \begin{array}{lcl} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$
- `removeVertex` — $O(|E|)$, трябва да се премахнат входящите ребра
- `addEdge` — ?

Списък на наследници

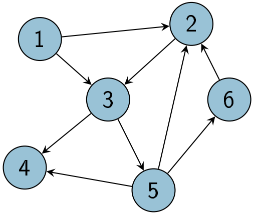
$$D = \left\{ \begin{array}{lcl} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$
- `removeVertex` — $O(|E|)$, трябва да се премахнат входящите ребра
- `addEdge` — $O(1)$

Списък на наследници

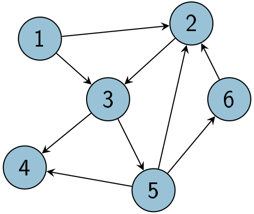
$$D = \left\{ \begin{array}{lll} 1 & \to & (2,3) \\ 2 & \to & (3) \\ 3 & \to & (4,5) \\ 4 & \to & () \\ 5 & \to & (2,4,6) \\ 6 & \to & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$
- `removeVertex` — $O(|E|)$, трябва да се премахнат входящите ребра
- `addEdge` — $O(1)$
- `removeEdge` — ?

Списък на наследници

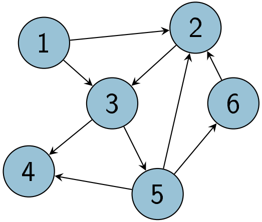
$$D = \left\{ \begin{array}{ccc} 1 & \rightarrow & (2,3) \\ 2 & \rightarrow & (3) \\ 3 & \rightarrow & (4,5) \\ 4 & \rightarrow & () \\ 5 & \rightarrow & (2,4,6) \\ 6 & \rightarrow & (2) \end{array} \right\}$$



- $D_i = \{v \mid (v_i, v) \in E\}$, може да е множество или списък
- памет — $O(|V| + |E|)$
- `successors` — $O(1)$
- `isEdge` — $O(|V|)$ (ако е множество — $O(1)$)
- `addVertex` — $O(1)$
- `removeVertex` — $O(|E|)$, трябва да се премахнат входящите ребра
- `addEdge` — $O(1)$
- `removeEdge` — $O(|V|)$ (ако е множество — $O(1)$)

Списък на инцидентност

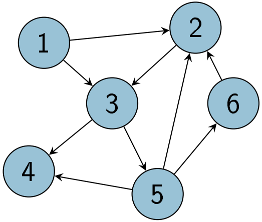
$$E = ((1,2),\quad (2,3),\quad (5,6),\quad (1,3),\quad (3,5),\quad (3,4),\quad (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество

Списък на инцидентност

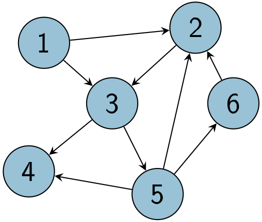
$$E = ((1,2),\quad (2,3),\quad (5,6),\\ (1,3),\quad (3,5),\quad (3,4),\\ (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество
- памет — ?

Списък на инцидентност

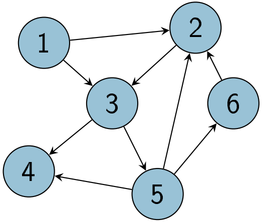
$$
\begin{aligned}
E = ((1,2),\quad & (2,3),\quad (5,6),\\
(1,3),\quad & (3,5),\quad (3,4),\\
(5,2),\quad & (5,4),\quad (6,2))
\end{aligned}
$$



- може да е представено като списък или множество
- памет — $O(|E|)$

Списък на инцидентност

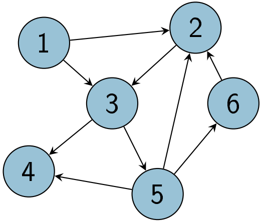
$$E = ((1,2),\quad (2,3),\quad (5,6),\\ (1,3),\quad (3,5),\quad (3,4),\\ (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — ?

Списък на инцидентност

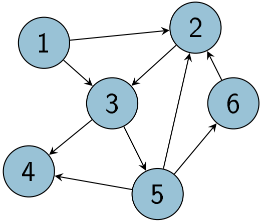
$$E = ((1,2),\quad (2,3),\quad (5,6),\\ (1,3),\quad (3,5),\quad (3,4),\\ (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$

Списък на инцидентност

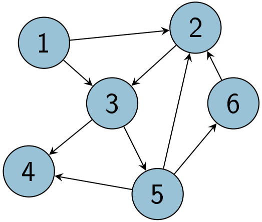
$$
\begin{aligned}
E = ((1,2), \quad & (2,3), \quad (5,6), \\
(1,3), \quad & (3,5), \quad (3,4), \\
(5,2), \quad & (5,4), \quad (6,2))
\end{aligned}
$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — ?

Списък на инцидентност

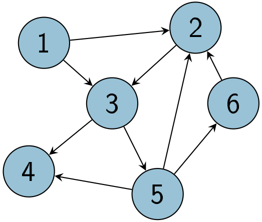
$$E = ((1,2), \quad (2,3), \quad (5,6), \\ (1,3), \quad (3,5), \quad (3,4), \\ (5,2), \quad (5,4), \quad (6,2))$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)

Списък на инцидентност

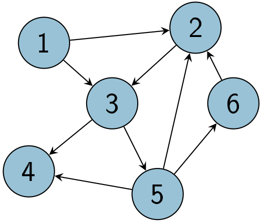
$$E = ((1,2),\quad (2,3),\quad (5,6),\\ (1,3),\quad (3,5),\quad (3,4),\\ (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)
- `addVertex`, `addEdge` — ?

Списък на инцидентност

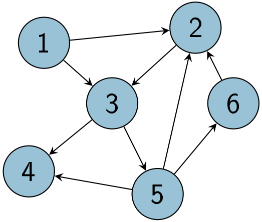
$$E = \begin{array}{lll}((1,2), & (2,3), & (5,6), \\ (1,3), & (3,5), & (3,4), \\ (5,2), & (5,4), & (6,2))\end{array}$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)
- `addVertex`, `addEdge` — $O(1)$

Списък на инцидентност

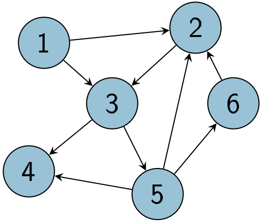
$$E = ((1,2),\quad (2,3),\quad (5,6),\\ (1,3),\quad (3,5),\quad (3,4),\\ (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)
- `addVertex`, `addEdge` — $O(1)$
- `removeVertex` — ?

Списък на инцидентност

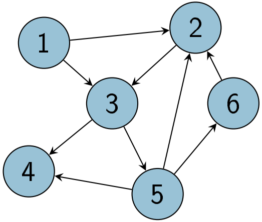
$$E = \begin{aligned}((1,2), \quad (2,3), \quad (5,6),\\ (1,3), \quad (3,5), \quad (3,4),\\ (5,2), \quad (5,4), \quad (6,2))\end{aligned}$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)
- `addVertex`, `addEdge` — $O(1)$
- `removeVertex` — $O(|E|)$

Списък на инцидентност

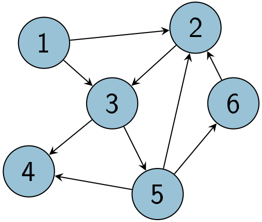
$$E = ((1,2),\quad (2,3),\quad (5,6),\\ (1,3),\quad (3,5),\quad (3,4),\\ (5,2),\quad (5,4),\quad (6,2))$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)
- `addVertex`, `addEdge` — $O(1)$
- `removeVertex` — $O(|E|)$
- `removeEdge` — ?

Списък на инцидентност

$$\begin{aligned} E = ((1,2), &\quad (2,3), \quad (5,6), \\ (1,3), &\quad (3,5), \quad (3,4), \\ (5,2), &\quad (5,4), \quad (6,2)) \end{aligned}$$



- може да е представено като списък или множество
- памет — $O(|E|)$
- `successors` — $O(|E|)$
- `isEdge` — $O(|E|)$ (ако е множество — $O(1)$)
- `addVertex`, `addEdge` — $O(1)$
- `removeVertex` — $O(|E|)$
- `removeEdge` — $O(|E|)$ (ако е множество — $O(1)$)

Локални задачи

Задача. Да се намерят върховете, които нямат наследници.

Локални задачи

Задача. Да се намерят върховете, които нямат наследници.
Решение. $\{u \mid \nexists v\,(u,v) \in E\}$

Локални задачи

Задача. Да се намерят върховете, които нямат наследници.
Решение. $\{u \mid \nexists v\, (u, v) \in E\}$

Задача. Да се намерят предшествениците на даден връх v.

Локални задачи

Задача. Да се намерят върховете, които нямат наследници.
Решение. $\{u \mid \nexists v\,(u, v) \in E\}$

Задача. Да се намерят предшествениците на даден връх v.
Решение. $\{u \mid (u, v) \in E\}$

Локални задачи

Задача. Да се намерят върховете, които нямат наследници.
Решение. $\{u \mid \nexists v\,(u, v) \in E\}$

Задача. Да се намерят предшествениците на даден връх v.
Решение. $\{u \mid (u, v) \in E\}$

Задача. Да се провери дали граф е симетричен.

Локални задачи

Задача. Да се намерят върховете, които нямат наследници.
Решение. $\{u \mid \nexists v\,(u,v) \in E\}$

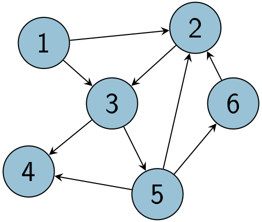
Задача. Да се намерят предшествениците на даден връх v.
Решение. $\{u \mid (u,v) \in E\}$

Задача. Да се провери дали граф е симетричен.
Решение. $\forall u, v \in V[(u,v) \in E \to (v,u) \in E]$

Обхождане в дълбочина

Обхождане на връх v:

- Обходи последователно всички наследници на v

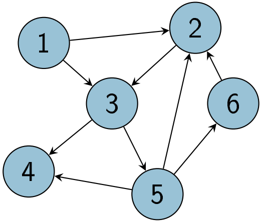


Обхождане в дълбочина

Обхождане на връх v:

- Обходи последователно всички наследници на v

- Имаме ли дъно?

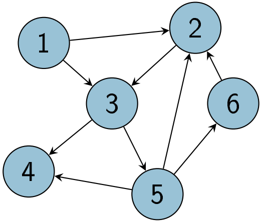


Обхождане в дълбочина

Обхождане на връх `v`:

- Обходи последователно всички наследници на `v`

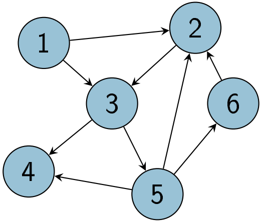
- Имаме ли дъно?
  - Да: при празен списък от наследници!



Обхождане в дълбочина

Обхождане на връх v:

- Обходи последователно всички наследници на v

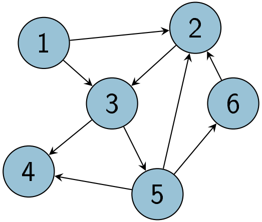


- Имаме ли дъно?
  - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?

Обхождане в дълбочина

Обхождане на връх v:

- Обходи последователно всички наследници на v

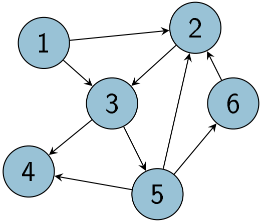


- Имаме ли дъно?
  - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?
  - Програмата също зацикля! Как да се справим с този проблем?

Обхождане в дълбочина

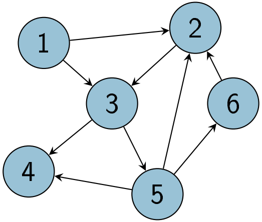
Обхождане на връх v:

- Обходи последователно всички наследници на v



- Имаме ли дъно?
  - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?
  - Програмата също зациклят! Как да се справим с този проблем?
  - Трябва да помним през кои върхове сме минали!

Обхождане в дълбочина



Обхождане на връх v:

- Обходи последователно всички наследници на v

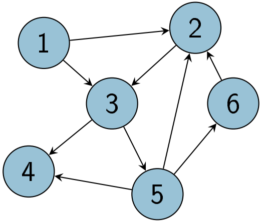
- Имаме ли дъно?
  - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?
  - Програмата също зациклЯ! Как да се справим с този проблем?
  - Трябва да помним през кои върхове сме минали!

1. да помним текущия път

Обхождане в дълбочина

Обхождане на връх v:

- Обходи последователно всички наследници на v

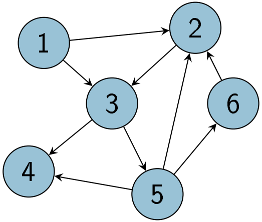


- Имаме ли дъно?
    - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?
    - Програмата също зацикля! Как да се справим с този проблем?
    - Трябва да помним през кои върхове сме минали!

1. да помним текущия път

2. да помним всички обходени до момента върхове

Обхождане в дълбочина



Обхождане на връх v:

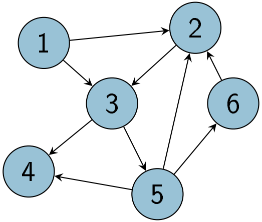
- Обходи последователно всички наследници на v

- Имаме ли дъно?
  - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?
  - Програмата също зацикля! Как да се справим с този проблем?
  - Трябва да помним през кои върхове сме минали!

1. да помним текущия път
   - намираме всички ациклични пътища

2. да помним всички обходени до момента върхове
   - обхождаме всеки връх по един път

Обхождане в дълбочина



Обхождане на връх v:

- Обходи последователно всички наследници на v

- Имаме ли дъно?
  - Да: при празен списък от наследници!
- Какво се случва ако графът е цикличен?
  - Програмата също зацикля! Как да се справим с този проблем?
  - Трябва да помним през кои върхове сме минали!

1. да помним текущия път
   - намираме всички ациклични пътища
   - сложност $O(|V||V|!)$
2. да помним всички обходени до момента върхове
   - обхождаме всеки връх по един път
   - сложност $O(|E|)$

Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
  - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$

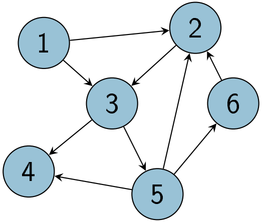


Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
  - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$
- Какво се случва ако графът е цикличен?

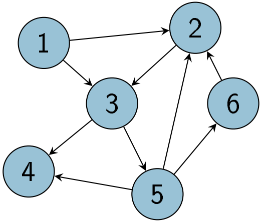


Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
    - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$
- Какво се случва ако графът е цикличен?
    - Ако има път: намира го.

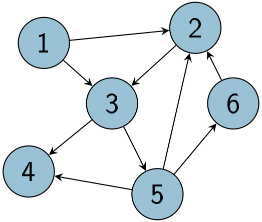


Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
  - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$
- Какво се случва ако графът е цикличен?
  - Ако има път: намира го.
  - Ако няма път: програмата зацикля!

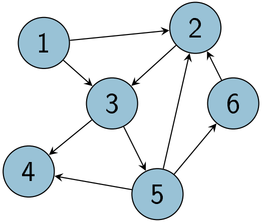


Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
    - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$
- Какво се случва ако графът е цикличен?
    - Ако има път: намира го.
    - Ако няма път: програмата зациклЯ!
    - Трябва да помним през кои върхове сме минали!

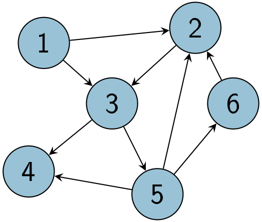


Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
    - Маркират се всички наследници `s` на `v` за обхождане на ниво $n + 1$
- Какво се случва ако графът е цикличен?
    - Ако има път: намира го.
    - Ако няма път: програмата зацикля!
    - Трябва да помним през кои върхове сме минали!

① да помним текущия път за всеки връх в текущото ниво

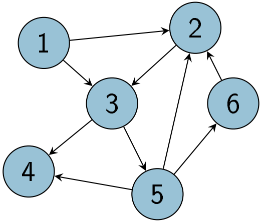
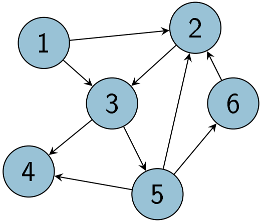


Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
  - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$



- Какво се случва ако графът е цикличен?
  - Ако има път: намира го.
  - Ако няма път: програмата зацикля!
  - Трябва да помним през кои върхове сме минали!

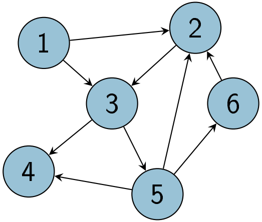
1. да помним текущия път за всеки връх в текущото ниво

2. да помним всички обходени до момента върхове

Схема на обхождане в ширина

Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
  - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$

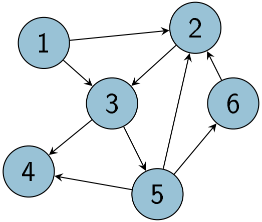


- Какво се случва ако графът е цикличен?
  - Ако има път: намира го.
  - Ако няма път: програмата зациклѝ!
  - Трябва да помним през кои върхове сме минали!

1. да помним текущия път за всеки връх в текущото ниво
   - обхождаме с повторение на върховете (всички ациклични пътища)

2. да помним всички обходени до момента върхове
   - обхождаме всеки връх по един път

Схема на обхождане в ширина



Обхождане, започващо от връх `u`:

- Маркира се `u` за обхождане на ниво 1
- За всеки връх `v` маркиран за ниво n:
  - Маркират се всички наследници `s` на `v` за обхождане на ниво $n+1$
- Какво се случва ако графът е цикличен?
  - Ако има път: намира го.
  - Ако няма път: програмата зациклят!
  - Трябва да помним през кои върхове сме минали!

1. да помним текущия път за всеки връх в текущото ниво
   - обхождаме с повторение на върховете (всички ациклични пътища)
   - сложност по време **и памет** $O(|V||V|!)$
2. да помним всички обходени до момента върхове
   - обхождаме всеки връх по един път
   - сложност $O(|E|)$ по време и $O(|V|)$ по памет

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е “надълбоко”

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е “надълбоко”
- ...може да обхожда избрани пътища преференциално

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е "надълбоко"
- ...може да обхожда избрани пътища преференциално
- ...естествено се реализира с **рекурсия**

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е “надълбоко”
- ...може да обхожда избрани пътища преференциално
- ...естествено се реализира с **рекурсия**

Обхождането в ширина...

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е “надълбоко”
- ...може да обхожда избрани пътища преференциално
- ...естествено се реализира с **рекурсия**

Обхождането в ширина...

- ...използва памет за всички разглеждани нива

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е "надълбоко"
- ...може да обхожда избрани пътища преференциално
- ...естествено се реализира с **рекурсия**

Обхождането в ширина...

- ...използва памет за всички разглеждани нива
- ...е подходящо за задачи, където търсим "най-плитката" цел

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина. . .

- . . . използва памет само за обходените до момента върхове
- . . . е подходящо за задачи, където търсим единична цел, която е “надълбоко”
- . . . може да обхожда избрани пътища преференциално
- . . . естествено се реализира с **рекурсия**

Обхождането в ширина. . .

- . . . използва памет за всички разглеждани нива
- . . . е подходящо за задачи, където търсим “най-плитката” цел
- . . . обхожда графа “равномерно”

Сравнение на обхождане в дълбочина и ширина

Обхождането в дълбочина...

- ...използва памет само за обходените до момента върхове
- ...е подходящо за задачи, където търсим единична цел, която е “надълбоко”
- ...може да обхожда избрани пътища преференциално
- ...естествено се реализира с **рекурсия**

Обхождането в ширина...

- ...използва памет за всички разглеждани нива
- ...е подходящо за задачи, където търсим “най-плитката” цел
- ...обхожда графа “равномерно”
- ...естествено се реализира с **итерация**

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
    - $u = v$ (дъно)

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
    - $u = v$ (дъно)
    - $\exists w\,(u, w) \in E$ & има път от w до v

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
  - $u = v$ (дъно)
  - $\exists w \, (u, w) \in E$ & има път от w до v

Търсене на път **в ширина**

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (проследване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
  - $u = v$ (дъно)
  - $\exists w\,(u, w) \in E$ & има път от w до v

Търсене на път **в ширина**

- удобно е да пазим в **стек** обходените **ребра**

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
    - $u = v$ (дъно)
    - $\exists w\,(u, w) \in E$ & има път от w до v

Търсене на път **в ширина**

- удобно е да пазим в **стек** обходените **ребра**
- винаги намираме най-късия по брой ребра път

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
    - $u = v$ (дъно)
    - $\exists w\,(u, w) \in E$ & има път от w до v

Търсене на път **в ширина**

- удобно е да пазим в **стек** обходените **ребра**
- винаги намираме най-късия по брой ребра път
- конструираме пътя като се връщаме назад по ребрата

Търсене на път

Задача. Да се намери път между върховете u и v, ако такъв има.

Търсене на път **в дълбочина**

- удобно е да пазим текущия път в **стек**
- при стъпка напред (последване на наследник) добавяме в стека
- при стъпка назад (няма повече наследници) махаме от стека
- има път от u до v, ако:
  - $u = v$ (дъно)
  - $\exists w\,(u, w) \in E$ & има път от w до v

Търсене на път **в ширина**

- удобно е да пазим в **стек** обходените **ребра**
- винаги намираме най-късия по брой ребра път
- конструираме пътя като се връщаме назад по ребрата
- има път от u до v, ако при обхождане, стартиращо от u, съществува ниво n, така че на него обхождаме v

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха *u*.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра…
- …което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
    - при стъпка назад макрираме върха като непосетен

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра. . .
- . . . което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра…
- …което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!
- търсене в **ширина**

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!
- търсене в **ширина**
  - ако в нивото пазим само ребрата: връщайки се назад трябва да изпробваме рекурсивно всички възможни комбинации

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха *u*.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!
- търсене в **ширина**
  - ако в нивото пазим само ребрата: връщайки се назад трябва да изпробваме рекурсивно всички възможни комбинации
    - става еквивалентно на търсене в дълбочина!

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!
- търсене в **ширина**
  - ако в нивото пазим само ребрата: връщайки се назад трябва да изпробваме рекурсивно всички възможни комбинации
    - става еквивалентно на търсене в дълбочина!
  - ако в нивото пазим целия път: при завършване на обхождането получаваме списък от всички ациклични пътища

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха *u*.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!
- търсене в **ширина**
  - ако в нивото пазим само ребрата: връщайки се назад трябва да изпробваме рекурсивно всички възможни комбинации
    - става еквивалентно на търсене в дълбочина!
  - ако в нивото пазим целия път: при завършване на обхождането получаваме списък от всички ациклични пътища
  - компромисен вариант: вместо път пазим връх и указател към предшественик

Намиране на всички пътища

Задача. Да се намерят всички пътища, започващи от върха u.

- ако графът е цикличен, пътищата са безкрайно много!
- можем да търсим всички **ациклични** пътища
- трябва да изследваме всички възможни комбинации от ребра...
- ...което означава, че трябва да позволим повтарянето на върхове!
- търсене в **дълбочина**
  - при стъпка назад макрираме върха като непосетен
  - всъщност посетените върхове са точно тези от текущия път!
- търсене в **ширина**
  - ако в нивото пазим само ребрата: връщайки се назад трябва да изпробваме рекурсивно всички възможни комбинации
    - става еквивалентно на търсене в дълбочина!
  - ако в нивото пазим целия път: при завършване на обхождането получаваме списък от всички ациклични пътища
  - компромисен вариант: вместо път пазим връх и указател към предшественик
    - не намалява сложността по памет в най-лошия случай!

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.

Решение:

- започваме от произволен връх u

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.

Решение:

- започваме от произволен връх u
- обхождаме в ширина или дълбочина

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.

Решение:

- започваме от произволен връх u
- обхождаме в ширина или дълбочина
- запомняме обходените върхове и не ги обхождаме повторно

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.

Решение:

- започваме от произволен връх u
- обхождаме в ширина или дълбочина
- запомняме обходените върхове и не ги обхождаме повторно
- запомняме пътя (както при търсене на път)

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.
Решение:

- започваме от произволен връх u
- обхождаме в ширина или дълбочина
- запомняме обходените върхове и не ги обхождаме повторно
- запомняме пътя (както при търсене на път)
- ако достигнем вече обходен връх — има цикъл, връщаме намерения път

Намиране на цикли

Задача. Да се намери цикъл в даден граф, ако такъв има.

Решение:

- започваме от произволен връх u
- обхождаме в ширина или дълбочина
- запомняме обходените върхове и не ги обхождаме повторно
- запомняме пътя (както при търсене на път)
- ако достигнем вече обходен връх — има цикъл, връщаме намерения път
- ако успеем да обходим всички ребра — няма цикъл

Покриващо дърво

Дефиниция

Дърво наричаме ацикличен граф, в който има единствен връх r, така че:

- r няма предшественици $(d^-(r) = 0)$
- другите върхове имат точно един предшественик $(\forall v \neq r\ d^-(v) = 1)$

Покриващо дърво

Дефиниция

Дърво наричаме ацикличен граф, в който има единствен връх r, така че:

- r няма предшественици ($d^-(r) = 0$)
- другите върхове имат точно един предшественик ($\forall v \neq r\ d^-(v) = 1$)

Задача. По даден (свързан) граф (V, E) да се намери дърво (V, E') за $E' \subseteq E$.

Покриващо дърво

Дефиниция

Дърво наричаме ацикличен граф, в който има единствен връх r, така че:

- r няма предшественици $(d^-(r) = 0)$
- другите върхове имат точно един предшественик $(\forall v \neq r\ d^-(v) = 1)$

Задача. По даден (свързан) граф (V, E) да се намери дърво (V, E') за $E' \subseteq E$.

Решение:

- обхождаме в дълбочина или ширина

Покриващо дърво

Дефиниция

Дърво наричаме ацикличен граф, в който има единствен връх r, така че:

- r няма предшественици ($d^-(r) = 0$)
- другите върхове имат точно един предшественик ($\forall v \neq r\ d^-(v) = 1$)

Задача. По даден (свързан) граф (V, E) да се намери дърво (V, E') за $E' \subseteq E$.

Решение:

- обхождаме в дълбочина или ширина
- запомняме обходените върхове и не ги обхождаме повторно

Покриващо дърво

Дефиниция

Дърво наричаме ацикличен граф, в който има единствен връх r, така че:

- r няма предшественици $(d^-(r) = 0)$
- другите върхове имат точно един предшественик $(\forall v \neq r\ d^-(v) = 1)$

Задача. По даден (свързан) граф (V, E) да се намери дърво (V, E') за $E' \subseteq E$.

Решение:

- обхождаме в дълбочина или ширина
- запомняме обходените върхове и не ги обхождаме повторно
- добавяме в дървото всеки нов връх и реброто, по което сме дошли до него

Покриващо дърво

Дефиниция

Дърво наричаме ацикличен граф, в който има единствен връх r, така че:

- r няма предшественици ($d^-(r) = 0$)
- другите върхове имат точно един предшественик ($\forall v \neq r\ d^-(v) = 1$)

Задача. По даден (свързан) граф (V, E) да се намери дърво (V, E') за $E' \subseteq E$.

Решение:

- обхождаме в дълбочина или ширина
- запомняме обходените върхове и не ги обхождаме повторно
- добавяме в дървото всеки нов връх и реброто, по което сме дошли до него
- приключваме при обхождане на всички върхове

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \dots, v_{i_n}$, така че $\forall (u, w) \in E\, \exists j < k\, (u = v_{i_j} \,\&\, w = v_{i_k})$.

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E\, \exists j < k\, (u = v_{i_j} \,\&\, w = v_{i_k})$.

Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^-(u) = 0$

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E \, \exists j < k \, (u = v_{i_j} \,\&\, w = v_{i_k})$.
Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^{-}(u) = 0$
- обхождаме l от началото, като за всеки обходен връх u:

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E\, \exists j < k\, (u = v_{i_j} \,\&\, w = v_{i_k})$.

Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^-(u) = 0$
- обхождаме l от началото, като за всеки обходен връх u:
  - за всеки наследник v на u

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E\, \exists j < k\, (u = v_{i_j} \,\&\, w = v_{i_k})$.

Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^-(u) = 0$
- обхождаме l от началото, като за всеки обходен връх u:
    - за всеки наследник v на u
        - премахваме реброто (u, v) от графа

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E \, \exists j < k \, (u = v_{i_j} \,\&\, w = v_{i_k})$.

Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^-(u) = 0$
- обхождаме l от началото, като за всеки обходен връх u:
    - за всеки наследник v на u
        - премахваме реброто (u, v) от графа
        - ако $d^-(v) = 0$ добавяме го в края на l

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E\, \exists j < k\, (u = v_{i_j} \,\&\, w = v_{i_k})$.
Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^-(u) = 0$
- обхождаме l от началото, като за всеки обходен връх u:
    - за всеки наследник v на u
        - премахваме реброто (u, v) от графа
        - ако $d^-(v) = 0$ добавяме го в края на l
- ако изчерпим l, но в графа останат още ребра — грешка, имало е цикъл

Топологично сортиране

Задача. По даден граф (V, E) търсим пермутация на върховете $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, така че $\forall (u, w) \in E\, \exists j < k\, (u = v_{i_j} \,\&\, w = v_{i_k})$.

Решение: Събираме списък от върхове l

- първоначално в l поставяме всички $u \in V$, за които $d^-(u) = 0$
- обхождаме l от началото, като за всеки обходен връх u:
  - за всеки наследник v на u
    - премахваме реброто (u, v) от графа
    - ако $d^-(v) = 0$ добавяме го в края на l
- ако изчерпим l, но в графа останат още ребра — грешка, имало е цикъл
- в противен случай, l е решение на задачата