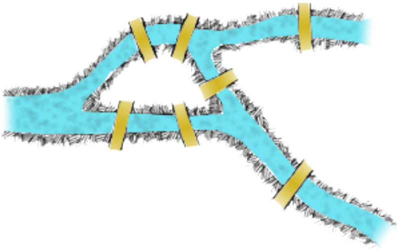
Grafi in algoritmi

Nino Bašić

28. junij 2011

Königsberški mostovi:



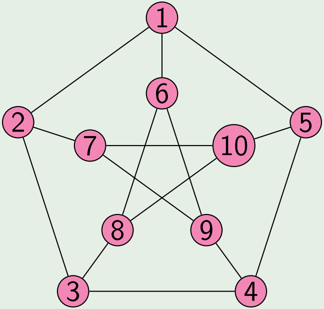
- Ali lahko naredimo tak sprehod (oz. obhod), da gremo čez vsak most natanko enkrat?
- Leonhard Euler, 1735 (začetek teorije grafov)

Definicija

Graf $G = (V, E)$ je množica vozlišč V in množica povezav E. Povezave so (neurejeni) pari vozlišč.

Example

$V = \{1, 2, \ldots, 10\}$
$E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 1\}, \{1, 6\}, \{2, 7\}, \ldots\}$

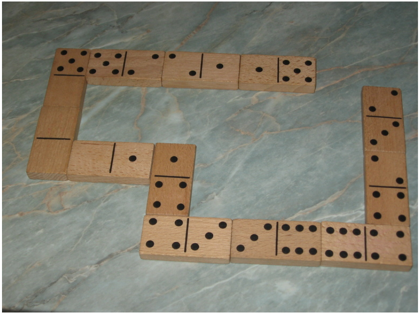


- Število vozlišč: $n = |V|$, število povezav: $m = |E|$
- Vozlišči u in v sta sosednji, če obstaja povezava med njima (tj. če je $\{u, v\} \in E$).
- Zaporedje točk $v_0, v_1, v_2, \ldots, v_k$ je sprehod dolžine k, če sta vsaki dve zaporedni točki povezani ($\{v_i, v_{i+1}\} \in E$).
- Sprehod $v_0, v_1, v_2, \ldots, v_k$ je pot, če so vse točke različne.
- Če je $v_0 = v_k$, tak sprehod imenujemo obhod.
- Obhod, kjer so vsa vozlišča (razen prvega in zadnjega) med seboj različna, je cikel.
- Graf je povezan, če obstaja pod med vsakima dvema točkama.
- Stopnja vozlišča u, $\deg(u)$, je število tistih povezav, ki imajo u za eno od krajišč.
- Lema o rokovanju: $\sum_{u \in V} \deg(u) = 2|E|$

- vozlišča: mesta; povezava: avtocesta (med dvema mestoma)

- vozlišča: mesta; povezava: avtocesta (med dvema mestoma)
- vozlišča: države; povezava: državi imata skupno mejo

- vozlišča: mesta; povezava: avtocesta (med dvema mestoma)
- vozlišča: države; povezava: državi imata skupno mejo
- socialna omrežja (Facebook)
  vozlišča: uporabniki; povezava: prijateljstvo

- vozlišča: mesta; povezava: avtocesta (med dvema mestoma)
- vozlišča: države; povezava: državi imata skupno mejo
- socialna omrežja (Facebook)
  vozlišča: uporabniki; povezava: prijateljstvo
- 

  vozlišča: $\{0, 1, 2, \ldots, 6\}$; povezava $\{u, v\}$: domina, ki ima na eni strani u pik, na drugi pa v pik

Uporaba grafov

- vozlišča: atomi; povezave: kemijske vezi

Uporaba grafov

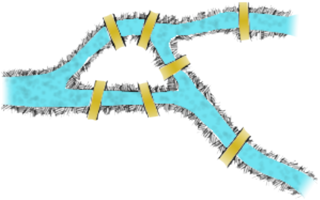
- vozlišča: atomi; povezave: kemijske vezi
- vozlišča: telefonske številke; povezave: vsi vzpostavljeni klici

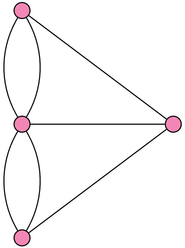
Uporaba grafov

- vozlišča: atomi; povezave: kemijske vezi
- vozlišča: telefonske številke; povezave: vsi vzpostavljeni klici
- vozlišča: besede iz SSKJ-ja; povezava: urejevalniška razdalja (edit distance) med besedama je 1

- vozlišča: atomi; povezave: kemijske vezi
- vozlišča: telefonske številke; povezave: vsi vzpostavljeni klici
- vozlišča: besede iz SSKJ-ja; povezava: urejevalniška razdalja (edit distance) med besedama je 1

Nazaj k Königsberškim mostovom:



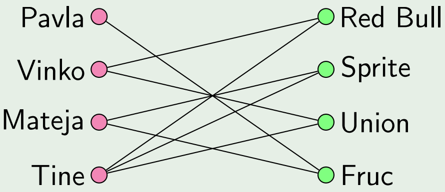


Opomba: graf ima vzporedne povezave.

Dvodelni grafi

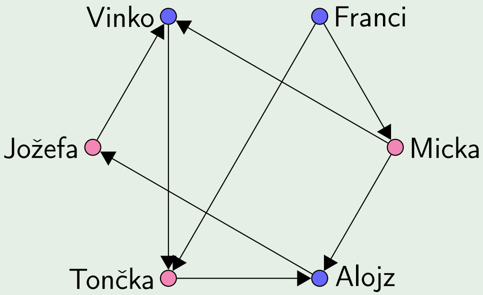
Včasih lahko vozlišča grafa razdelimo na dve množici: $V = A \cup B$, tako da povezava povezuje po eno vozlišče iz množice A in eno iz množice B. Takim grafom rečemo dvodelni grafi.

Example



Včasih so povezave usmerjene: povezava je urejeni par vozlišč.
Takim grafom rečemo usmerjeni grafi ali digrafi.

Example



Še nekaj posebnih grafov

- Polni graf K_n: vsaki dve vozlišči sta med seboj povezani
  Koliko povezav ima polni graf?
- Drevo je povezan graf brez ciklov.
  Koliko povezav ima drevo?
- Pot P_n je graf, ki ga sestavlja ena sama pot na n vozliščih.
- Cikel C_n je graf, ki ga sestavlja en sam cikel na n vozliščih.
- Polni dvodelni graf $K_{r,s}$ je dvodelen graf, kjer je vsaka točka iz prve podmnožice povezana z vsako točko iz druge podmnožice. Graf $K_{1,s}$ imenujemo zvezda.
- Ravninski grafi

Če ima graf malo povezav je redek. Če ima veliko povezav je gost.

Včasih imajo povezave še kakšno dodatno lastnost (dolžina, cena, kapaciteta, . . .).

- V = {0, 1, 2, . . . , n-1}
- ```
  const int N_MAX = 1000;
  int n; // st. vozlisc
  bool matSos[N_MAX][N_MAX];
  ```
- matrika velikosti $n \times n$
- `matSos[i][j] == True` če sta vozlišči i in j sosednji
- ```
  void dodajPovezavo(int u, int v) {
    matSos[u][v] = matSos[v][u] = True;
  }
  ```
- prednosti:
  - dodajanje/brisanje povezave: $O(1)$
  - preverjanje, če obstaja povezava v grafu: $O(1)$

- slabosti:
  - porabi $O(|V|^2)$ pomnilnika
  - pregled vseh sosedov: $O(|V|)$
  - pregled vseh povezav: $O(|V|^2)$

- za vsako točko si shranimo seznam njenih sosedov
- preprosta implementacija:

```
const int N_MAX = 1000;
const int D_MAX = 100;
int n;
int deg[N_MAX]; // stopnja vozlisca
int sezSos[N_MAX][D_MAX];
```

- boljša implementacija: `sezSos` je tabela povezanih seznamov
-
```
void dodajPovezavo(int u, int v) {
    sezSos[u][deg[u]] = v;
    deg[u]++;
    sezSos[v][deg[v]] = u;
    deg[v]++;
}
```

- prednosti:
  - porabi $O(|V| + |E|)$ pomnilnika (boljša implementacija)
  - pregled vseh sosedov vozlišča u: $O(deg(u))$
  - pregled vseh povezav: $O(|V| + |E|)$
- slabosti:
  - preverjanje, če obstaja povezava (u, v) v grafu: $O(deg(u))$
  - brisanje povezave (u, v): $O(deg(u) + deg(v))$

Problemi, za katere poznamo učinkovite algoritme:

- topološko urejanje
- iskanje v globino (DFS) in iskanje v širino (BFS)
- iskanje povezanih komponent
- barvanje grafa z dvema barvama
- iskanje Eulerjevega obhoda
- minimalno vpeto drevo
- iskanje najkrajših poti
- ...

Problemi, za katere ni učinkovitih algoritmov (NP-težki problemi):

- iskanje Hamiltonovega cikla
- barvanje grafa s tremi barvami
- ...