Problem 7 mostova Kenigsberga

Problem sedam mostova Kenigsberga je klasičan problem u teoriji grafova. Grad Kenigsberg (današnji Kalinjingrad) imao je sedam mostova koji su povezivali četiri kopnene mase. Zadatak je bio osmisliti šetnju koja bi prešla svaki most tačno jednom.

Leonhard Ojler je 1736. dokazao da takva šetnja nije moguća. Utemeljio je teoriju grafova tako što je kopnene mase predstavio kao čvorove, a mostove kao grane. Ojler je pokazao da je za postojanje takve šetnje potrebno da **najviše dva čvora imaju neparan stepen** (broj grana koje izlaze iz čvora). Pošto su u ovom grafu svi čvorovi imali neparan stepen, problem nije imao rješenje.

Ojlerov graf: definicija

Graf je **Ojlerov** ako sadrži **Ojlerovu kružnicu**, tj. kružnicu koja prolazi kroz svaku granu grafa tačno jednom.

1. Potrebni uslovi:

Ako graf G ima Ojlerov ciklus, mora ispuniti oba uslova:

- **Povezanost:**
  Ako graf nije povezan, ne može se iz jednog dijela grafa preći u drugi, što onemogućava prolazak kroz sve grane.
- **Parni stepen čvorova:**
  U Ojlerovom ciklusu svaki put koji prolazi kroz čvor ulazi i izlazi iz njega. Dakle, broj ulazaka i izlazaka mora biti paran, što znači da svaki čvor ima paran stepen.

2. Dovoljni uslovi:

Ako su oba uslova ispunjena, konstruisaćemo Ojlerov ciklus:

- Počinjemo od bilo kog čvora i pratimo grane tako da svaku koristimo tačno jednom. Na osnovu parnog stepena, iz svakog čvora možemo izaći svaki put kada u njega uđemo.
- Ako naiđemo na čvor iz kog više nema neposećenih grana, ali još uvek postoje grane u drugim delovima grafa, koristimo povezanost da se vratimo na te delove i proširimo ciklus.
- Proces završavamo kada su sve grane iskorišćene, čime dobijamo Ojlerov ciklus.

Dokaz:

Dokaz. ($\Rightarrow$) Graf je povezan po definiciji Ojlerovog grafa. Neka je

$$u_1 e_1 u_2 e_2 \dots u_n e_n u_1$$

Ojlerova tura u G. Posmatrajmo sada proizvoljan čvor $v \in V(G)$. Ako se čvor v pojavljuje l puta u konturi u slučaju kada je $v \neq u_1$, odnosno $l+1$ ako je $v = u_1$, onda je stepen tog čvora $d_G(v) = 2l$.
($\Leftarrow$) Posmatrajmo u G stazu najveće dužine:

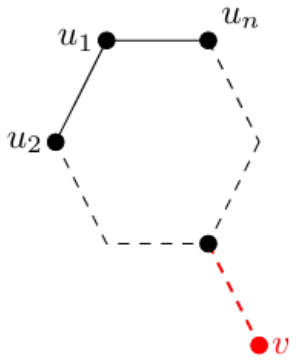
$$u_1 e_1 u_2 e_2 \dots u_n e_n u_{n+1}$$

Pokazaćemo da su prvi i poslednji čvor isti, kao i da se svi čvorovi i grane grafa pojavljuju u toj stazi.

(i) $u_1 = u_{n+1}$: Ako pretpostavimo suprotno, da je $u_1 \neq u_{n+1}$, onda je u toj konturi neparan broj grana incidentan sa u_1 u toj stazi. Kako je stepen čvora u_1 paran, postoji grana $e \in E(G)$ koja nije sadržana u posmatranoj stazi. U tom slučaju bismo mogli kreirati dužu stazu od posmatrane, što dovodi do kontradikcije.

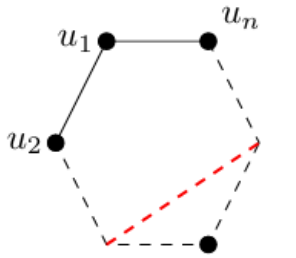
(ii) $\{u_1, \ldots, u_n\} = V(G)$:

Pretpostavimo suprotno, da postoje čvorovi koji nisu na posmatranoj stazi. Kako je G povezan, postoji grana $\{u_i, v\}$, $i \in \{1, \ldots, n\}$, sa osobinom $v \notin \{u_1, \ldots, u_n\}$. U tom slučaju možemo konstruisati stazu veće dužine od posmatrane.



(iii) $\{e_1, \ldots, e_n\} = E(G)$:

Sada kada znamo da je posmatrana staza u stvari kontura koja sadrži sve čvorove grafa, pokazaćemo da su sve grane grafa na toj konturi. Ako pretpostavimo suprotno, da postoji grana koja nije na toj konturi, onda bismo ponovo mogli konstruisati dužu stazu od posmatrane.



□

Usmereni graf

Za povezani **usmereni graf** G, potreban i dovoljan uslov da bude Ojlerov jeste:

1. Za svaki čvor, broj ulaznih grana mora biti jednak broju izlaznih grana.
2. Graf mora biti jakopovezan (tj. postoji put između bilo koja dva čvora kada se zanemari orijentacija grana).

Primer neusmerenog Ojlerovog grafa

Graf sa sledećim stepenima čvorova je Ojlerov:

- Čvorovi: A,B,C,D
- Stepeni: deg(A)=2,deg(B)=4,deg(C)=2,deg(D)=2

Ilustracija:

```
A — B
|   |
C — D
```

Primer neusmerenog grafa koji nije Ojlerov

Graf sa sledećim stepenima čvorova nije Ojlerov:

- Čvorovi: A,B,C,D
- Stepeni: deg(A)=3,deg(B)=4,deg(C)=3,deg(D)=2

Zaključak

Neusmereni graf je Ojlerov ako i samo ako:

1. Graf je povezan.
2. Svi čvorovi imaju paran stepen.

Usmereni graf je Ojlerov ako i samo ako:

1. Broj ulaznih grana jednak je broju izlaznih grana za svaki čvor.
2. Graf je jakopovezan.

Potrebni i dovoljni uslovi da graf bude polu Ojlerov

Definicija polu-Ojlerovog grafa:

Graf je **polu-Ojlerov** ako postoji **Otvorena Ojlerova staza**. To je staza koja prolazi kroz svaku granu grafa tačno jednom, ali ne mora biti zatvorena (počinje i završava se u različitim čvorovima).

Potreban i dovoljan uslov:

Graf GGG je **polu-Ojlerov** ako i samo ako:

1. Graf GGG je povezan kada se zanemare čvorovi stepena nula.
2. Tačno dva čvora imaju neparan stepen.

Dokaz

Potreban uslov:

1. **Graf je povezan kada se zanemare čvorovi stepena nula:** Ako graf nije povezan, ne može postojati staza koja prolazi kroz sve grane jer bi staza morala "preskočiti" nepovezane komponente. Dakle, povezanost grafa (zanemarujući čvorove stepena nula) je neophodan uslov.
2. **Tačno dva čvora imaju neparan stepen:** Prema teoremi o rukovanju, u svakom grafu zbir stepena svih čvorova je paran broj. Ako postoji Ojlerova staza koja nije zatvorena, ona mora početi u jednom čvoru (ulaz) i završiti u drugom (izlaz). To implicira da samo ta dva čvora mogu imati neparan stepen.

Dovoljan uslov:

1. **Graf je povezan kada se zanemare čvorove stepena nula:** Ako je graf povezan, svaka grana je dostupna iz bilo kog čvora u kojem se staza trenutno nalazi. To omogućava prelazak kroz sve grane bez prekida.
2. **Tačno dva čvora imaju neparan stepen:** Ako tačno dva čvora imaju neparan stepen, možemo započeti Ojlerovu stazu u jednom od tih čvorova i završiti je u drugom, prolazeći tačno jednom kroz svaku granu. Ovo je zato što ulaz i izlaz u čvorove parnog stepena balansiraju, dok ulaz i izlaz u čvorove neparnog stepena ostaju neuravnoteženi, omogućavajući početak i kraj staze.

Dokaz. ($\Rightarrow$) Sličnim rezonovanjem kao u prethodnom dokazu, za svaki čvor grafa koji nije na krajevim Ojlerovog puta, na tom putu se pojavljuje paran broj grana koje su incidentne sa tim čvorom. Za dva čvora na kraju puta imamo, osim eventualnog parnog broja grana unutar staze, još po jednu granu koja im je incidentna, odakle dobijamo da ta dva čvora imaju neparne stepene.
($\Leftarrow$) Neka su u i v jedini čvorovi grafa neparnog stepena. Posmatrajmo sada graf G' koji dobijamo od grafa G dodavanjem nove grane e koja je incidentna sa čvorovima u i v (ona može biti paralelna nekim već postojećime granama). U grafu G' su sada svi čvorovi parnog stepena. Prema Teoremi 90, G' sadrži Ojleorvu turu. Po definiciji, ta tura sadrži granu G. Oduzimanjem grane e iz Ojlerove ture grafa G' dobijamo Ojlerov put u grafu G.

Primjer: **Priferov Kod → Stablo**

Dati Priferov kod: {2, 2, 4, 5}.

Koraci za rekonstrukciju stabla:

1. Napravimo listu svih čvorova: {1, 2, 3, 4, 5, 6}.
2. Brojimo pojavljivanja svakog čvora u Priferovom kodu:
    - 2: 2, 4: 1, 5: 1, ostali (1, 3, 6): 0.
3. Pronalazimo najmanji čvor koji nije u Priferovom kodu (s 0 pojavljivanja) i povezujemo ga sa prvim čvorom iz Priferovog koda:
    - Povezujemo 1 sa 2.
    - Uklanjamo 2 iz koda i smanjujemo broj pojavljivanja 2.
4. Ažuriramo Priferov kod i pojavljivanja:
    - Kod: {2, 4, 5}.
    - Pojavljivanja: 2: 1, 4: 1, 5: 1, 3: 0, 6: 0.
5. Nastavljamo proces:
    - Povezujemo 3 sa 2.
        - Uklanjamo 2 iz koda i smanjujemo broj pojavljivanja 2.
    - Kod: {4, 5}.
    - Pojavljivanja: 4: 1, 5: 1, 6: 0.
    - Povezujemo 6 sa 4.
        - Uklanjamo 4 iz koda i smanjujemo broj pojavljivanja 4.
    - Kod: {5}.
    - Pojavljivanja: 5: 1.
    - Povezujemo 5 sa 6.
        - Uklanjamo 5 iz koda.
6. Preostali čvorovi:
    - Ostaju čvorovi 4 i 5.
    - Povezujemo 4 sa 5.

Rekonstruisano stablo: {(1, 2), (3, 2), (6, 4), (5, 6), (4, 5)}.

Definisati težinski graf

Težinski graf je poširenje klasičnog grafa u kojem je realan broj pridruzen svakoj grani, koji predstavljaju određene karakteristike, kao što su trošak, udaljenost, vrijeme, kapacitet ili bilo koja druga mjerljiva vrijednost.

Odnosno, težinski graf je uređena trojka: $G = (V, E, w)$

gdje su:

- V – skup **čvorova** (tjemena),
- E – skup **grana** (ivica), gde je svaka grana uređeni ili neuređeni par čvorova,
- $w: E \to R$ – funkcija težine, koja svakoj grani pridružuje numeričku vrednost $w(e)$, pri čemu $e \in E$.

U slučaju kada težine predstavljaju **udaljenost ili trošak**, obično su **nenegativne** $(w: E \to R\ +)$.

Vrste težinskih grafova

Težinski grafovi mogu biti **orijentisani** ili **neorijentisani**:

1. **Neorijentisani težinski graf** – ako je $(u, v) \in E$, tada je $(v, u) \in E$ i težina $w(u, v) = w(v, u)$.
    - Primjer: Putna mreža gdje su putevi dvosmjerni i imaju istu udaljenost u oba smijera.
2. **Orijentisani težinski graf** – ako postoji grana (u, v), ne mora postojati grana (v, u), i težine mogu biti različite $w(u, v) \neq w(v, u)$.
    - Primjer: Saobraćajna mreža gdje putevi imaju različitu dužinu u različitim smjerovima (npr. zbog jednosmjernih ulica).

Svojstva težinskih grafova

1.Pozitivne ili negativne težine

- U većini slučajeva težine su **pozitivne** (npr. udaljenosti ili troškovi).
- Ako graf sadrži **negativne težine**, tada algoritmi poput Dijkstrinog ne rade pravilno, ali Bellman-Fordov algoritam može riješiti problem.

2.Povezanost

- Težinski graf može biti **povezan** (iz svakog čvora postoji put do bilo kog drugog) ili **nepovezan** (postoje izolovane komponente).

3.Minimalni i maksimalni putevi

- Kod težinskih grafova, posebno su značajni **najkraći putevi** (algoritmi Dijkstra, Bellman-Ford) i **minimalna razapinjuća stabla** (Kruskalov i Primov algoritam).

Primeri težinskih grafova u stvarnom svetu

Mreža puteva i saobraćaja

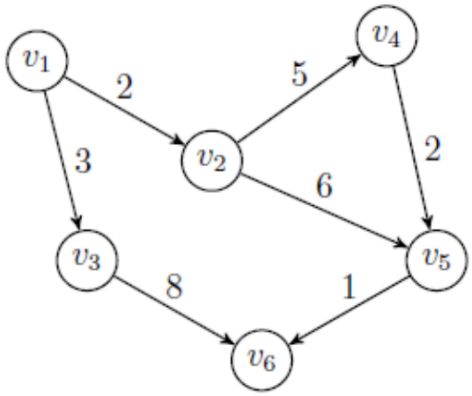
- Čvorovi predstavljaju gradove, a grane puteve sa težinama koje označavaju razdaljinu ili vrijeme putovanja.

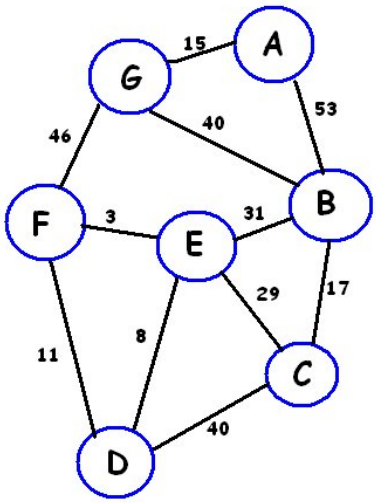
Telekomunikacione mreže

- Čvorovi su ruteri, a grane predstavljaju kablove sa težinama koje označavaju brzinu prenosa podataka ili kašnjenje.

Finansijski modeli

- U berzanskim modelima, čvorovi mogu predstavljati kompanije, a grane troškove transakcija između njih.





Ojlerova tura – definicija

Ojlerova tura u neusmerenom grafu je **put koji prolazi kroz svaku granu tačno jednom** i završava u **početnom čvoru** (ciklus). Ako se ne mora završiti u početnom čvoru, tada govorimo o **Ojlerovom putu**, a ne turi.

Uslov za postojanje Ojlerove ture u neusmerenom grafu:

1. Graf mora biti **povezan** (osim izolovanih temena).
2. Sva temena moraju imati **paran stepen** (paran broj izlaznih grana).

Algoritam za pronalaženje Ojlerove ture (Flerov algoritam – Fleury's algorithm):

Koraci:

1. **Proveri da li graf ispunjava uslove za Ojlerovu turu:**
    - Ako neki čvor ima neparan stepen → nema Ojlerove ture.
2. **Izaberi početni čvor:**
    - Bilo koji čvor (s obzirom da svi imaju paran stepen).
3. **Kreći se rekurzivno ili iterativno po grafu:**
    - Na svakom koraku idi na susedni čvor **preko grane koja nije most**, osim ako ne postoji druga opcija.
    - Kada uzmeš granu, **obriši je** iz grafa.
4. **Zabeleži redosled prelaska grana i temena.**

Pseudokod algoritma:

```python
def is_valid_next_edge(u, v, graph):
    # Ako postoji samo jedna susedna grana, koristi je
    if len(graph[u]) == 1:
        return True

    # Inače, proveri da li je (u,v) most (uklanjanjem bi graf postao nepovezan)
    visited = set()
    count1 = dfs_count(u, visited, graph)

    # Privremeno uklonimo granu (u,v)
    graph[u].remove(v)
    graph[v].remove(u)

    visited = set()
    count2 = dfs_count(u, visited, graph)

    # Vratimo granu nazad
    graph[u].append(v)
    graph[v].append(u)

    return count1 <= count2  # ako se broj čvorova smanji, (u,v) je most

def dfs_count(u, visited, graph):
    visited.add(u)
    count = 1
    for v in graph[u]:
        if v not in visited:
            count += dfs_count(v, visited, graph)
    return count

def print_euler_util(u, graph):
    for v in list(graph[u]):
        if is_valid_next_edge(u, v, graph):
            print(f"{u} -> {v}")
            graph[u].remove(v)
            graph[v].remove(u)
            print_euler_util(v, graph)

def print_euler_tour(graph):
    # Nađi čvor sa nenultim stepenom
    u = next((v for v in graph if len(graph[v]) > 0), None)
    if u is None:
        return
    print_euler_util(u, graph)
```