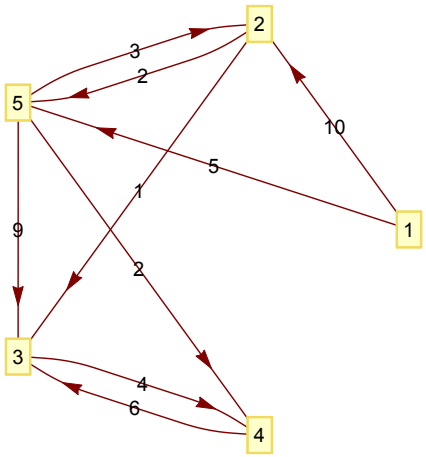
Diskretno modeliranje 2013/2014

10. & 11. vaje

Algoritmi na grafih

1. Implementiraj Dijkstrov algoritem v Pythonu. Preveri delovanje na spodnjem primeru.

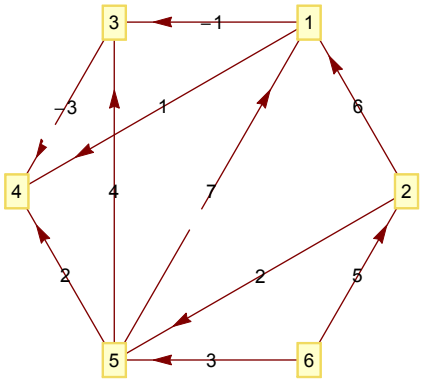


Vhodni podatki: Z n podamo št. vozlišč, v seznamu e podamo usmerjene povezave, v seznamu c njihove pripadajoče vrednosti, s pa je izbrano vozlišče, od kjer nas zanimajo vse razdalje do ostalih vozlišč.

Izhodni podatki: Seznam očetov vozlišč ter seznam pripadajočih minimalnih razdalj vozlišč do s.

Ideja: Najprej poiščemo vozlišče v, ki je najbližje s. Nato popravimo razdaljo vseh njegovih sosedov $(v, w) \in e$ do s, če je le-ta manjša v primeru, da gremo do njih prek vozlišča v. Če je to res, njihovega očeta nastavimo na v. Nato v odstranimo iz množice možnih vozlišč in nadaljujemo, dokler množica ne postane prazna.

2. Implementiraj algoritem za topološko urejanje acikličnih grafov v Pythonu. Preveri delovanje na spodnjem primeru.



VHODNI PODATKI: Z n podamo št. vozlišč, v seznamu pa e podamo usmerjene povezave.

IZHODNI PODATKI: Topološko urejen seznam vozlišč, tj. če je $(u, v) \in e$, je v seznamu u pred v.

IDEJA: V acikličnem grafu obstaja vozlišče brez izhodnih povezav. To vozlišče postavimo na začetek seznama ter ga odstranimo z grafa. Nato ponovimo postopek na manjšem grafu, dokler ne odstranimo vseh vozlišč.

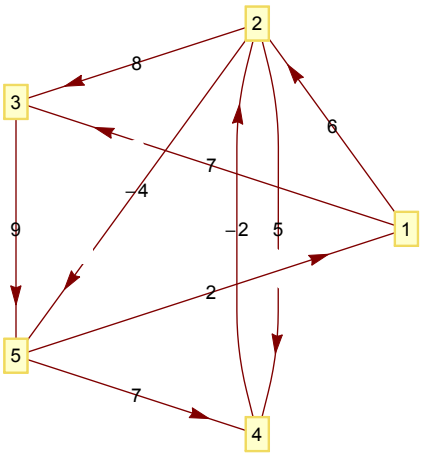
3. Implementiraj algoritem za iskanje najkrajših poti v acikličnih grafih v Pythonu s pomočjo uporabe topološkega urejanja. Preveri delovanje na primeru iz naloge 2.

   VHODNI PODATKI: Z n podamo št. vozlišč, v seznamu e podamo usmerjene povezave, v seznamu c njihove pripadajoče vrednosti, s pa je izbrano vozlišče, od kjer nas zanimajo vse razdalje do ostalih vozlišč.

   IZHODNI PODATKI: Seznam očetov vozlišč ter seznam pripadajočih minimalnih razdalj vozlišč do s.

   IDEJA: Najprej vozlišča topološko uredimo. Nato poiščemo, kje se nahaja vozlišče s, ter pregledamo vse sosede od s dalje na način kot pri Dijkstrovem algoritmu. Topološka urejenost zagotavlja, da so sosedi danega vozlišča vedno desno od njega.

4. Implementiraj Bellman-Fordov algoritem za iskanje najkrajših poti v Pythonu. Preveri delovanje na spodnjem primeru.

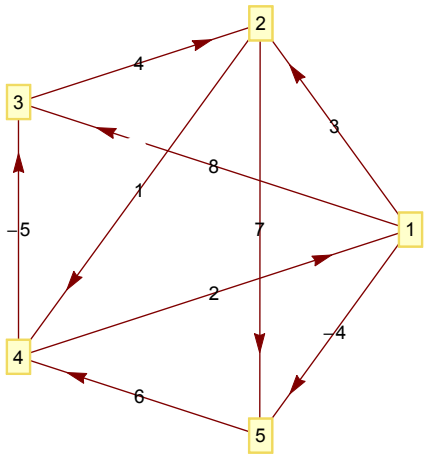


VHODNI PODATKI: Z n podamo št. vozlišč, v seznamu e podamo usmerjene povezave, v seznamu c njihove pripadajoče vrednosti, s pa je izbrano vozlišče, od kjer nas zanimajo vse razdalje do ostalih vozlišč.

IZHODNI PODATKI: Seznam očetov vozlišč ter seznam pripadajočih minimalnih razdalj vozlišč do s.

IDEJA: Algoritem gre $(n-1)$-krat po vseh povezavah ter ugotavlja, če je možno od s do desnega krajišča povezave priti hitreje prek levega krajišča. Če s tem postopkom ne dobimo optimalne rešitve, graf vsebuje negativen cikel.

5. Implementiraj Floyd-Warshalov algoritem za iskanje VSEH najkrajših poti v Pythonu. Preveri delovanje na spodnjem primeru.



VHODNI PODATKI: Z W podamo matrikov vseh povezav med pari vozlišč v grafu. Če povezava ne obstaja, je na ustreznem mestu ∞.

IZHODNI PODATKI: Seznam minimalnih razdalj med vsemi vozlišči (ter opcijsko seznam vozlišč z najvišjim indeksom na poti med izbranima vozliščema).

IDEJA: Algoritem na k-tem koraku preveri ali je moč ceneje priti iz i do j prek k, sicer ohrani prejšnjo rešitev.