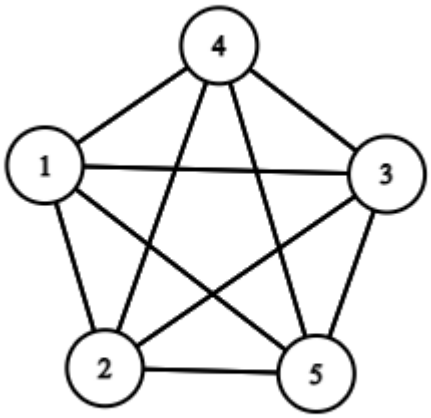
Problem:
Denimo, da je n mest povezanih med seboj s poleti po principu vsak z vsakim. Iz prvega mesta želimo obiskati vsa ostala mesta tako, da nobenega mesta ne obiščemo več kot enkrat (pri tem je vseeno, v katerem mestu končamo). Dolžine poletov med posameznimi mesti so znane, iščemo pa najkrajšo pot (skupno dolžino poletov) iz prvega mesta, ki obišče vsa ostala mesta.

A)

Model problema, ki vsebuje 5 mest, kjer vozlišče predstavlja mesto, povezava pa letalsko linijo med dvema mestoma.



B)

```
MAX_RAZDALJA = 99999

def poisci_najkrajso_pot(trenutno_mesto, n):
    """ Poiskati moramo najkrajšo pot, ki vsebuje vsa mesta iz seznama
    Mesto je definirano z imenom, koordinatama ter sosednjimi mesti, s katerimi je povezan
    Razdaljo med mesti izračunamo s pomočjo evklidske razdalje (√(Δx**2 + Δy**2) => funkcija razdalja(a, b))
    in jo lahko uporabimo kot zračno razdaljo med dvema mestoma. """
    obiskana_mesta = []
    min_razdalja = MAX_RAZDALJA
    sum_razdalja = 0
    obiskana_mesta.push(trenutno_mesto)
    dokler len(obiskana_mesta) < n:
        za sosednje_mesto iz mesto.sosedi:
            če je razdalja(mesto, sosednje_mesto) < min_razdalja in sosednje_mesto ni v obiskana_mesta:
                min_razdalja = razdalja(mesto, sosednje_mesto)
                najblizje_mesto = sosednje_mesto
        obiskana_mesta.push(najblizje_mesto)
        trenutno_mesto = najblizje_mesto
        sum_razdalja += min_razdalja
        min_razdalja = MAX_RAZDALJA
    return { obiskana_mesta: sum_razdalja } # Vrnemo sled obiskanih mest in razdaljo med obiskanimi mesti

def main():
    mesta = ["Ljubljana", "Zagreb", "Sarajevo", "Beograd", "Skopje"]
    za mesto iz mesta:
        poti.push(poisci_najkrajso_pot(mesto))

    za pot iz poti:
        pot z najkrajšo razdaljo je ta, ki jo iščemo
```

C) Sled programa

```
"""
Sled programa:

razdalje: Lj-Zg: 120km, Lj-Sar: 250km, Lj-Beo: 350km, Lj-Sko: 420km, Zg-Sar: 90km, Zg-Beo: 100km, Zg-Sko: 180km
          Sar-Beo: 60km, Sar-Sko: 100km, Beo-Sko: 50km
n = 5

1. mesto = "Ljubljana"
   mesto.sosedi = ["Zagreb", "Sarajevo", "Beograd", "Skopje"]

   1.1. razdalja("Ljubljana", "Zagreb") < min_razdalja
            - min_razdalja = razdalja("Ljubljana", "Zagreb")
            - najblizje_mesto = "Zagreb"
            - trenutno_mesto = "Zagreb"
        razdalja("Ljubljana", "Sarajevo) > min_razdalja
        razdalja("Ljubljana", "Beograd") > min_razdalja
        razdalja("Ljubljana", "Skopje") > min_razdalja
   1.2. razdalja("Zagreb", "Ljubljana") < min_razdalja && "Ljubljana" in obiskana_mesta
        razdalja("Zagreb", "Sarajevo") < min_razdalja
            - min_razdalja = razdalja("Zagreb", "Sarajevo")
            - najblizje_mesto = "Sarajevo"
            - trenutno_mesto = "Sarajevo"
        razdalja("Zagreb", "Beograd") > min_razdalja
        razdalja("Zagreb", "Skopje") > min_razdalja
   ...
   1.5. vrne { ["Ljubljana", "Zagreb", "Sarajevo", "Beograd", "Skopje"] : 320 }
```

```
2. mesto = "Zagreb"
   mesto.sosedi = ["Ljubljana", "Sarajevo", "Beograd", "Skopje"]

   2.1. razdalja("Zagreb", "Ljubljana") < min_razdalja && "Ljubljana" in obiskana_mesta
        razdalja("Zagreb", "Sarajevo") < min_razdalja
            - min_razdalja = razdalja("Zagreb", "Sarajevo")
            - najblizje_mesto = "Sarajevo"
            - trenutno_mesto = "Sarajevo"
        razdalja("Zagreb", "Beograd") > min_razdalja
        razdalja("Zagreb", "Skopje") > min_razdalja
   ...

5. mesto = "Skopje"
   len(obiskana_mesta) == n:

   "Najkrajša pot je sledeča: ["Ljubljana", "Zagreb", "Sarajevo", "Beograd", "Skopje"], skupna razdalja pa je 320km."

"""
```