Aleksander Wilberg

IS-211 Algoritmer & datastrukturer | Universitetet I Agder, Kristiansand, 2022

Mandatory Assignment 3b

Github repository:

<https://github.com/Wildenberg/Assignment3>

TSP.git

# Trestruktur

Diagram

Description automatically generatedFor å finne den billigste ruten fra Timisoara til Bucharest, så var det var til god hjelp å sette opp en trestruktur med alle relevante reiseruter, som TSP kan ta fra start til mål. Denne trestrukturen har utelukket alle byer som er irrelevante, som ikke er et bindeledd mellom Timisoara og Bucharest. Vi kan da se at brukeren har 6 forskjellige veier som han kan ta, og besøke totalt 13 forskjellige byer. Når vi ser på minst antall byer som han må besøke, så er det 5, mens det høyeste antall byer som kan besøkes på en tur er 8. Prisen derimot på den dyreste reisen er ikke basert på antall byer man besøker, men heller lengden på reisen man tar.

# Dijkstra’s algoritme

For å starte implementasjonen av Djikstra’s algoritme, så lagde jeg først en «starting vortex n» tabell for å finne verdiene fra start til hver av destinasjonene mellom Timisoara og Bucharest. Dette resulterte i en tabell vist under:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| selected vortex | Arad | Zerind | Oradea | Sibiu | Fagaras | Rimnicu Vilcea | Lugoj | Mehadia | Drobeta | Craiova | Pitesti | Bucharest |
| Lugoj | 118 | ∞ | ∞ | ∞ | ∞ | ∞ | 111 | ∞ | ∞ | ∞ | ∞ | ∞ |
| Arad | 118 | ∞ | ∞ | ∞ | ∞ | ∞ | 111 | 181 | ∞ | ∞ | ∞ | ∞ |
| Zerind | 118 | 193 | ∞ | 258 | ∞ | ∞ | 111 | 181 | ∞ | ∞ | ∞ | ∞ |
| Mehadia | 118 | 193 | 264 | 258 | ∞ | ∞ | 111 | 181 | ∞ | ∞ | ∞ | ∞ |
| Zerind | 118 | 193 | 264 | 258 | ∞ | ∞ | 111 | 181 | 256 | ∞ | ∞ | ∞ |
| Drobeta | 118 | 193 | 264 | 258 | ∞ | ∞ | 111 | 181 | 256 | ∞ | ∞ | ∞ |
| Sibiu | 118 | 193 | 264 | 258 | ∞ | ∞ | 111 | 181 | 256 | 376 | ∞ | ∞ |
| Oradea | 118 | 193 | 264 | 258 | 357 | 338 | 111 | 181 | 256 | 376 | ∞ | ∞ |
| Rimnicu Vilcea | 118 | 193 | 264 | 259 | 357 | 338 | 111 | 181 | 256 | 376 | ∞ | ∞ |
| Fagaras | 118 | 193 | 264 | 260 | 357 | 338 | 111 | 181 | 256 | 376 | ∞ | ∞ |
| Craiova | 118 | 193 | 264 | 261 | 357 | 338 | 111 | 181 | 256 | 376 | 435 | 568 |
| Pitesti | 118 | 193 | 264 | 262 | 357 | 338 | 111 | 181 | 256 | 376 | 435 | 568 |
| Bucharest | 118 | 193 | 264 | 263 | 357 | 338 | 111 | 181 | 256 | 376 | 435 | 536 |

En ulempe, som ikke er relevant med tanke på denne oppgaven men som er greit å ha med, er problemet som oppstår når man bruker negative verdier mellom noder. Dette er ikke mulig når det er snakk om avstander, men om det er snakk om verdier innen finans eller forretning så er det et relevant problem som man må ta hensyn. Grunnen til dette er det at om man har negative verdier, så kan dette gi et forvridd svar og syn på hvilken kombinasjon av noder som er «billigst» (Bari, 2018).

|  |  |  |  |
| --- | --- | --- | --- |
| n | Node | Total Cost From Start | Previous node |
| 0 | Timisoara | 0 | Timisoara |
| 1 | Lugoj | 111 | Timisoara |
| 2 | Arad | 118 | Timisoara |
| 3 | Zerind | 193 | Arad |
| 4 | Oradea | 264 | Zerind |
| 5 | Sibiu | 258 | Arad |
| 6 | Fagaras | 357 | Sibiu |
| 7 | R.V. | 338 | Sibiu |
| 8 | Mehadia | 181 | Lugoj |
| 9 | Drobeta | 256 | Mehadia |
| 10 | Craiova | 376 | Drobeta |
| 11 | Pitesti | 435 | R.V. |
| 12 | Bucharest | 536 | Pitesti |

Etter å ha fått en oversikt via tabell over, satt jeg også opp tabellen til høyre som viser igjen billigste vei fra Timisoara til Bucharest, men med en bedre oversikt over faktiske kostnader fra Timisoara til n i tillegg til tidligere node. Jeg har dedikert ett tall til hver av nodene, for enklere oversikt.

# TSP i Pseudo kode

Array cities[]

# References

Bari, A. (2018, 09. February). *3.6 Dijkstra Algorithm - Single Source Shortest Path - Greedy Method.* Retrieved from Youtube: https://www.youtube.com/watch?v=XB4MIexjvY0&t=20s

Kopec, D. (2021, 17 Juli). *How to solve the classic Traveling Salesman Problem in Java.* Retrieved from oracle: https://blogs.oracle.com/javamagazine/post/how-to-solve-the-classic-traveling-salesman-problem-in-java