LUNDS UNIVERSITET
Lunds Tekniska Högskola

Inlämningsuppgift, EDAF30

Kortaste vägen i en graf: Dijkstras algoritm

Maximilian Gustavsson

Dennis Pålsson

Lunds Tekniska Högskola, Campus Helsingborg

2024-12-20

Design

Projektet syftar på att bygga en datastruktur som representerar en karta med städer, vägar och deras avstånd. För att generalisera kallas städer “noder” och vägar “bågar” i denna uppgift. Dijkstras algoritm används för att hitta och visualisera de kortaste vägarna mellan olika städer. För att lösa uppgiften skapades följande klasser:

- Graph: Representerar kartan.
- Node: Representerar städer.
- Edge: Representerar vägar mellan städer.
- NodeSet: Hanterar noder under algoritmens körning.

Klasser

Edge: Representerar en båge mellan två noder i grafen som har attributen:

- *destinationNode*: slutnoden dit bågen leder.
- *edgeLength*: längden på bågen.

Funktionerna *getDestination* och *getLength* används för att hämta respektive slutnod och båglängd.

Node: Representerar en nod i grafen med attributen:

- *nodeName*: nodens namn.
- *nodeValue*: värdet som används i Dijkstras algoritm för att representera det kortaste avståndet från startnoden. Vid initiering sätts detta till maxvärdet för int.
- *parent*: referens till föregående nod på den kortaste vägen.

För att lägga till nya bågar används *addEdge*, och hela listan kan hämtas med *getEdges*.

NodeSet: Hanterar en mängd noder som bearbetas i Dijkstras algoritm. Funktionerna inkluderar:

- *add*: lägger till en nod i mängden om den inte redan finns.
- *removeMin*: tar bort och returnerar noden med det lägsta värdet.
- *isEmpty*: kontrollerar om mängden är tom.

Graph: Representerar en hel graf bestående av noder och bågar. Kan byggas manuellt eller läsas in från en fil. Funktionerna inkluderar:

- *find*: hittar en specifik nod i grafen baserat på dess namn.
- *resetVals*: återställer alla nodvärden till deras ursprungliga tillstånd.

Dijkstras algoritm

Dijkstra-filen innehåller tre stycken funktioner:

- *dijkstra*: implementerar Dijkstras algoritm för att beräkna de kortaste vägarna från en startnod.
- *generalDijkstra*: anpassar den klassiska algoritmen för att hitta den kortaste vägen baserat på antal passerade noder.
- *printPath*: visualiserar den kortaste vägen från startnod till slutnod samt längden beroende på vilken version av Dijkstras algoritm man har valt.

Testfiler

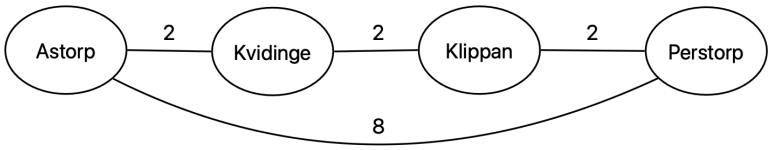
Ursprungligen fanns det fyra testfiler:

- *test_graph_small*: Testar klasserna *Node* och *Edge*.
- *test_nodeset*: Testar *NodeSet*-funktioner.
- *test_dijkstra*: Testar den klassiska algoritmen.
- *test_graph_nofile*: Testar *Graph* utan inläsning från fil.

Vi har lagt till ytterligare två, *test_graph_file* och *test_general_dijkstra* för att kunna testa funktionaliteten av hela programmet.

test_graph_file: Testar *Graph* med inläsning från textfilen *graf.txt*.

test_general_dijkstra: Testar funktionen *generalDijkstra* genom att se till att den väljer den direkta vägen från Åstorp till Perstorp med minst antal orter trots att det finns en kortare väg enligt vägavstånd genom Kvidinge och Klippan, se *figur 1*.



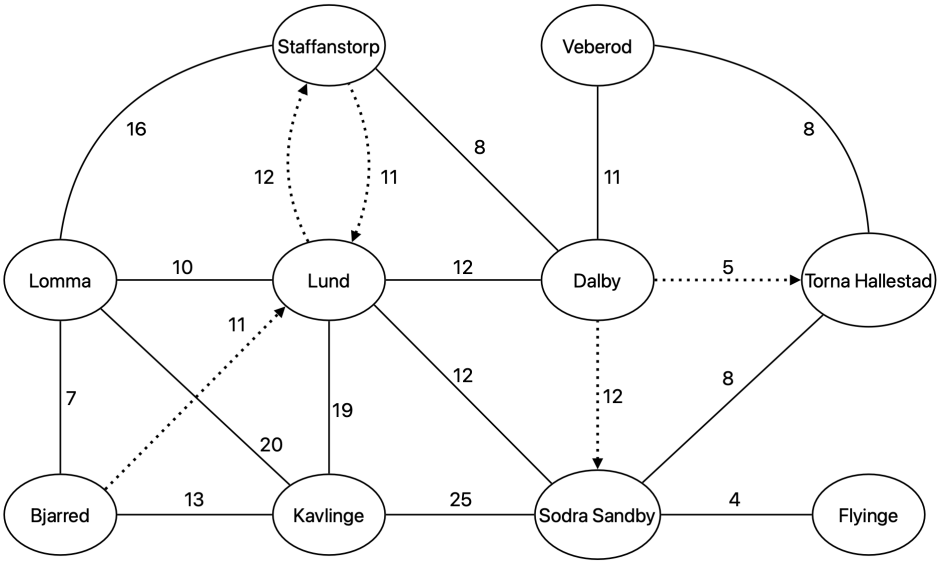
Figur 1. Graf med noderna och bågarna i test_general_dijkstra.

Användarinstruktion

All källkod finns i .zip-filen som packas upp i en vald mapp.
Programmet kompileras och körs genom *makefile* med följande kommandon:

- *make test_og:* Kör ursprungliga tester.
- *make test_all:* Kör alla tester.
- *make test_main:* Startar huvudprogrammet.

För huvudprogrammet behöver användaren välja två noder, en startnod och en slutnod, när man skriver in noderna är det viktigt att man följer de namn som anges i *figur 2* eftersom programmet är skiftlägeskänsligt. Det krävs också att användaren definierar den "kortaste vägen", detta gör man genom att följa det som skrivs ut av programmet och svara med antingen "A" för den kortaste vägen enligt vägavståndet eller "B" för den kortaste vägen enligt antal passerade noder.



Figur 2. Graf med noderna och bågarna i graf.txt. Streckade linjer med pil i ena änden betyder att vägen är enkelriktad.

Kommentarer

Vi ändrade i den ursprungliga textfilen *graf.txt* så att noden *Flyinge* har en båge tillbaka till *Sodra Sandby* för att få testprogrammet *test_graph_file.cc* att fungera.

Main-programmet fungerade utan problem för den ena medlemmen som har MacOS men för den andra som har Windows erhålls aldrig något resultat enligt *figur 3*.

```
./main
Grafen har skapats framgångsrikt från filen: graf.txt
Ange startnod: Lund
Ange slutnod: Lomma
Välj beräkningskriterium:
A) Kortaste väg enligt avstånd
B) Kortaste väg enligt antal steg
Val: B
Ingen väg hittades från Lund till Lomma.
```

Figur 3. Output från main efter att en startnod, slutnod och alternativ valts. Problemet kvarstår oavsett val av startnod, slutnod och alternativ och är fortfarande olöst.

Förutom att lösa ovanstående problem så kan man optimera programmet genom göra så det inte bryr sig om ifall man använder sig av stora eller små bokstäver när användaren skriver in noder.