



Kartläggning av serverhallar

Med viktberäkningar och geografisk sökning

David Pettersson, Jesper Hofling, Kristofer Larsson
Algoritmer och datastrukturer för geografisk informationsteknik,
DVG307, Högskolan i Gävle

davpet002@gmail.com

kristofer.larsson@live.se

hoflingjesper@gmail.com

2025-05-27

| | | |
|-----|--|----|
| 1 | Introduktion | 3 |
| 2 | Datastrukturer | 4 |
| 2.1 | Min-heap | 4 |
| 2.2 | Prioritetskö | 4 |
| 2.3 | Graf (med vertiser och kanter) | 4 |
| 2.4 | Quadtree | 5 |
| 2.5 | Delaunay | 5 |
| 3 | Algoritmer | 6 |
| 3.1 | Delaunay-triangulering | 6 |
| 3.2 | Dijkstra | 6 |
| 3.3 | Minimum Spanning Tree (MST) | 6 |
| 4 | Tester | 7 |
| 5 | Resultat | 10 |
| 5.1 | Kortaste vägen med hänsyn till vikterna (Dijkstra) | 10 |
| 5.2 | För en position inom ett område, ge information om närmaste serverhallar (Quadtree) | 13 |
| 5.3 | Visualisering av ett Minimum Spanning Tree (MST) | 15 |
| 6 | Utvärdering | 16 |
| 6.1 | Vidareutveckling | 16 |
| 7 | Referenser | 17 |
| | Bilaga A | 1 |
| | Bilaga B | 2 |

1 Introduktion

I enlighet med det upphandlade projektet för att kartlägga svenska serverhallar och dess sammankopplade nätverk ska en prototyp utvecklas. Denna rapport redogör för designval och beslut som kommer att fattas under utvecklingsarbetet. Detta för att slutprodukten ska ha alla möjligheter till att uppfylla kraven på prestanda, användbarhet och användarvänlighet. Målet är att utveckla denna prototyp som med hjälp av flera olika datastrukturer och algoritmer ska kunna beräkna kortaste vägen med hänsyn till vikter mellan två serverhallar och ge information om närmaste serverhallar för en angiven position eller punkt i en grafstruktur. Resultatet ska visualiseras med ett användarvänligt GUI som innehåller en tilltalande graf.

2 Datastrukturer

2.1 Min-heap

Min-heap ingår som struktur i prioritetsskön (se Bilaga Figur A1). Som namnet antyder, att Min-heap står för minimum-heap, så är denna implementering en heap som strävar efter att hålla noden med lägst värde högst upp i roten. Med ursprungspunkt ur roten hamnar sedan allt högre värden nedåt i trädet. Detta beteende är grundläggande för att algoritmerna i programmet ska finna rätt väg, med lägst motstånd svarande mot högst bandbredd.

2.2 Prioritetsskö

Prioritetsskön är en specialiserad struktur för att bibehålla en strukturerad och sorterad kö. Den nyttjas inom applikationen av olika algoritmer som behöver dess funktion. Bandbredd är måttet av framkomlighet inom det sammankopplade nätverket och serverhallarna vilket gör att ett lågt motstånd är att föredra. Ett lågt motstånd är i lika proportion relaterat till en hög bandbredd, därför kommer prioritetsskön att hålla elementet med lägst motstånd först i kön. På så sätt kan olika algoritmer inom programmet få ett beslutsstöd till vilken väg som är optimal.

2.3 Graf (med vertiser och kanter)

Grafen är en struktur som består av två abstrakta representationer av verkliga ting. Den består dels av vertiser eller s.k. noder som representerar serverhallarna och dels av kanter som representerar nätverksanslutningen mellan serverhallarna. En serverhalls överföringskapacitet och hur bra dess vidare anslutningar inom nätverket är definierar dess bandbredd. Alltså, en serverhall med hög bandbredd och anslutningar med hög bandbredd ger tillsammans ett segment inom det större nätverket som är prestandamässigt önskvärt att förlägga nätverksanslutningar via. Algoritmerna, när de appliceras på det nationella nätverket eller mellan två specifika serverhallar baserar sina matematiska beräkningar på en funktion av bandbredden och avståndet. Funktionen tar hänsyn till serverhallens bandbredd, bandbredden hos anslutningarna, samt hur långt avståndet är till nästa serverhall. Resultatet av en algoritm läses in i ett grafobjekt och kan användas för visualisering och implementering av faktiska optimerade rutter.

2.4 Quadtree

Quadtree är strukturen som tillåter inläsning av och sökning efter stora mängder punktojekt på ett tvådimensionellt plan. Planet representeras av en karta över Sverige i projektionen SWEREF99 TM och punktojekten representeras av vertiser som är serverhallarnas abstrakta representation. Vid initieringen av applikationen skapas ett nytt Quadtree-objekt där i samtliga serverhallar läggs till i form av vertiser innehållande koordinater i referenssystemet SWEREF99, serverhallens namn och dess bandbredd. Den speciella strukturen som fås genom implementeringen av ett Quadtree gör sökningar effektiva mätt i både tid och prestanda genom att en algoritm inte behöver söka igenom hela mängden av vertiser. Quadtree-strukturen som skapas vid inläsningen bygger på att vertiser fördelas till begränsade delmängder inom olika storlekar av rektangulärt avgränsade geografiska områden. Geografiska områden med en större och tätare förekomst av vertiser blir succesivt uppdelad i fler och mindre rektangulära områden. När en delmängd överskrider bryts området upp i fyra mindre rektanglar om och om igen. Men, i områden med gles förekomst av vertiser sker en måttlig eller ingen uppdelning alls. En sökning för att returnera serverhallar inom ett visst geografiskt område applicerar en centrumpunkt och en sök-rektangel på Quadtree-strukturen. Sökningen begränsas automatiskt till att endast ske inom de delområden som rektangeln skär, vilket är syftet med strukturen och det som gör den effektiv.

2.5 Delaunay

Delaunay är den grundläggande datastrukturen som de flesta av de andra strukturerna i prototypen bygger på. Den består av en geometrisk struktur där samtliga vertiser har kopplats samman med kanter för att forma ett nätverk. Det är detta anslutna geometriska nätverk som algoritmerna kan appliceras på. Denna tillgängliggörs inom prototypen genom att dess struktur lagras i ett grafobjekt.

3 Algoritmer

3.1 Delaunay-triangulering

Delaunay-trianguleringen är den algoritm inom prototypen som skapar förbindelsen mellan alla enskilda vertiser. Detta genom att utifrån vertisernas positioner skapa en geometrisk struktur av intet överlappande trianglar. Resultatet blir ett anslutet nätverk som kan användas för vidare beräkningar.

3.2 Dijkstra

Dijkstras algoritm, som beskrivs av Weiss, M.A. (2010), används inom prototypen för att finna den mest lämpade anslutningen mellan två serverhallar. Där den mest lämpade anslutningen, eller rutten, är den med högst bandbredd. På så sätt kan den både detektera lämpliga rutter för nya anslutningar eller alternativa rutter för omdirigering av trafik under tider för hög nätverksbelastning.

3.3 Minimum Spanning Tree (MST)

Ett Minimum Spanning Tree (MST) är en grafstruktur som är resultatet av en algoritm. Prototypens MST svarar mot det mest effektiva sättet att ansluta alla svenska serverhallar till ett sammankopplat nätverk samtigt som den högsta möjliga bandbredden uppnås. Algoritmen som applicerades är Prim's-algoritm och den arbetade på delaunay-trianguleringens struktur.

4 Tester

Givet de stora mängder data som den slutliga applikationen ska hantera har det asymptotiska beteendet hos algoritmerna och funktionerna studerats.

Tidstesterna är utformade för att på olika storlekar av datamängder innehållande vertiser utföra upprepade tester, vilket medför exaktare medelmätvärden.

Delmängdernas storlekar var: $n = 50 * 2^k$, $k = 0, 1, 2, \dots, 7$ och vardera test upprepades 10 000 gånger.

Resultaten redogörs för och tydliggörs genom ordonotation, kvoter och diagram. För en komplett redovisning av samtliga kvoter, se Bilaga Figur B1.

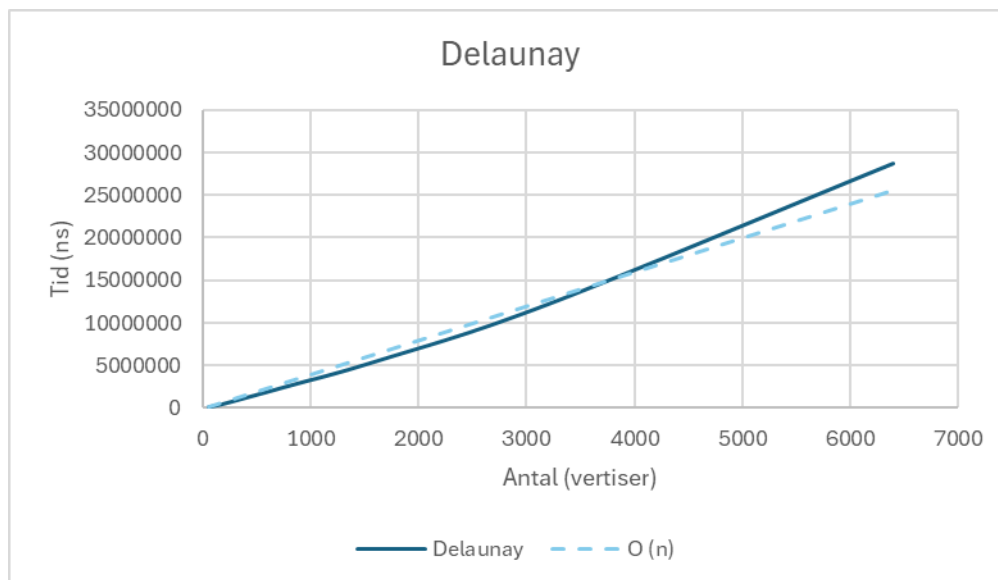
Delaunay

(Java Topology Suite, v1.19.0, Bowyer-Watson)

Förväntad Ordo: $O(n \log n)$ till $O(n^2)$.

Utfall: Medelkvoten, $T_{medel} = \left(\frac{1}{6} \sum_{i=1}^6 T_i\right) \approx 2,07$, utefter bilaga A.

Redogörelse: Resultatet är i enlighet med förväntningarna. Ett T medel på 2,07 svarar mot en tidskomplexitet på $O(n)$, se Figur 1.



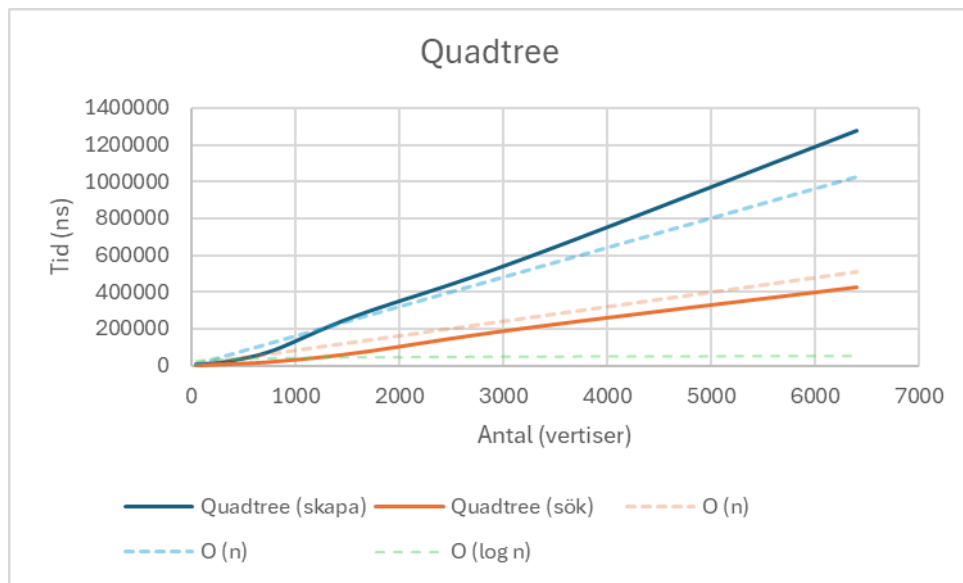
Figur 1. Ett diagram över Delaunay-algorithmens tidskomplexitet samt referens.

Quadtree

Förväntad Ordo: Skapa och söka $O(\log n)$.

Utfall: Medelkvoten, $T_{medel} = \left(\frac{1}{6} \sum_{i=1}^6 T_i\right) \approx 2,38$ (skapa), $\approx 2,20$ (sök).

Redogörelse: Resultatet är inte enligt förväntningarna men orsaken kan ligga i strukturen av testdata. Båda T medel på 2,38 och 2,20 svarar mot en tidskomplexitet på över $O(n)$, omkring $O(n \log n)$. Båda är mycket högre än förväntat. En möjlig orsak går att finna om man ser till de första testerna på storlekarna $n \leq 800$, vilket är tydligt i Figur 2. T medel är här 1,9 för skapa och 1,8 för sök.



Figur 2. Ett diagram över Quadtree-algoritmernas tidskomplexitet samt referens.

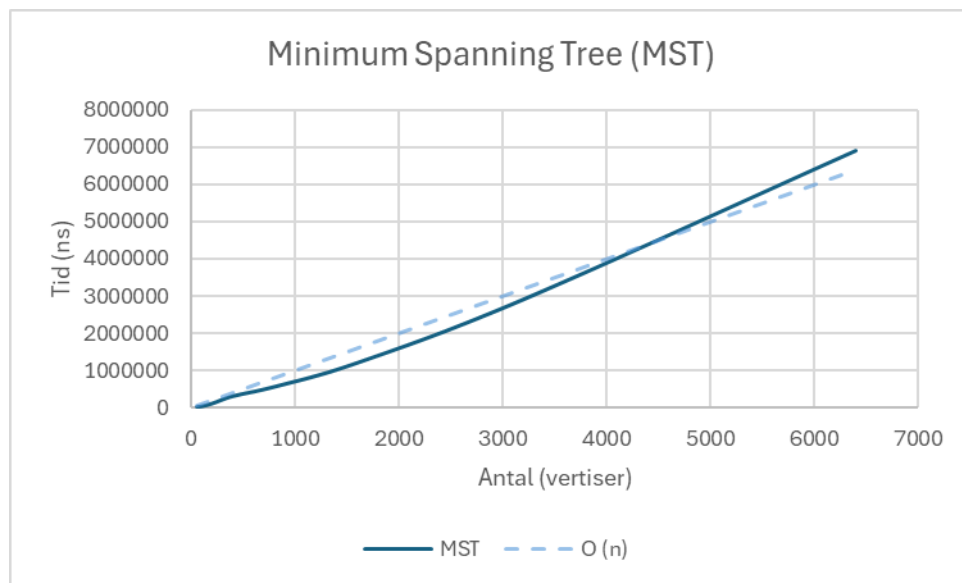
Minimum Spanning Tree (MST)

(Med Prims algoritmen)

Förväntad Ordo: $O((n + k)\log n)$, $k = \text{kanter}$. $\approx O(n \log n)$, för glesa grafer.

Utfall: Medelkvoten, $T_{medel} = \left(\frac{1}{6} \sum_{i=1}^6 T_i\right) \approx 2,20$.

Redogörelse: Tidskomplexiteten för ett MST står i direkt proportion till antalet vertiser och kanter i grafen som algoritmen appliceras på. Tidstesternas största datamängd var 6400 vertiser och därav går det genom skattning att placera prototypens graf inom gruppen glesa grafer. Utfallet som visualiserats i Figur 3 svarar mot förväntningarna.



Figur 3. Ett diagram över MST-algoritmens tidskomplexitet samt referens.

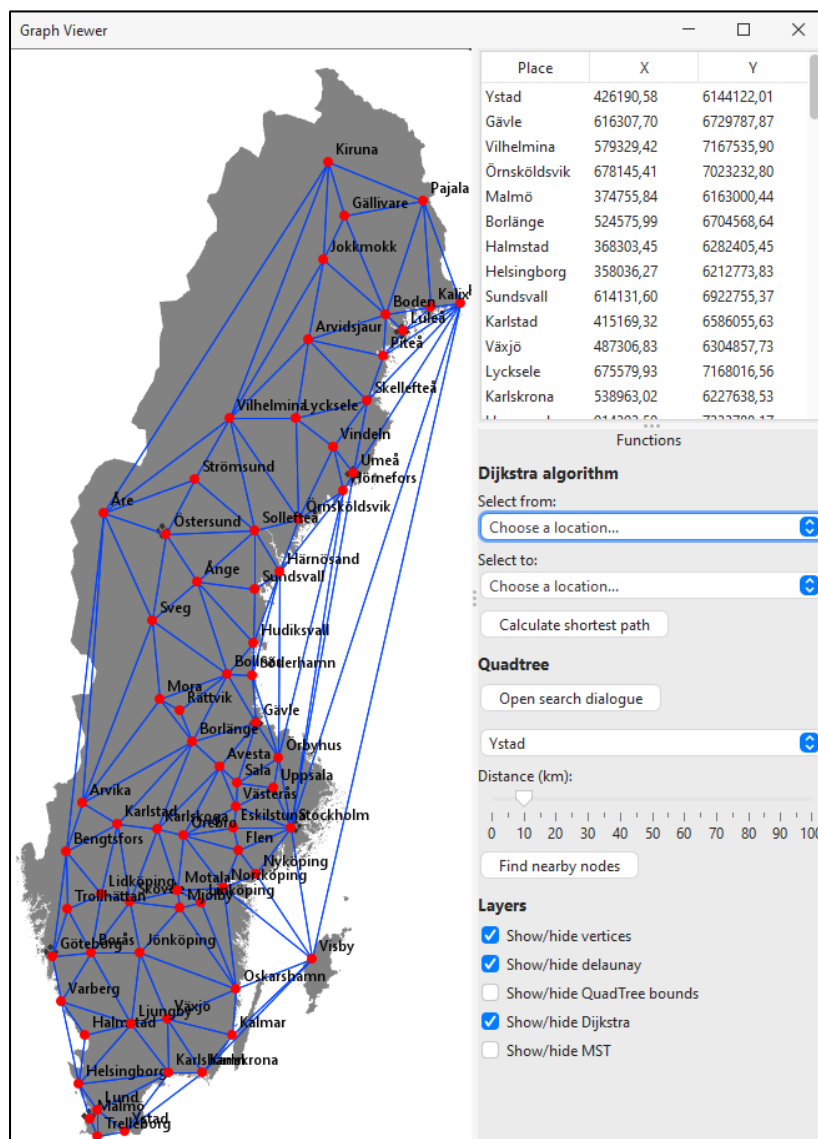
Utöver testerna som ingår i analysen av tidskomplexiteten, har det även genomförts ett antal JUnit-tester som kontrollerar att datastrukturerna och algoritmerna fungerar som de bör och är robusta i avseendet tillförlitlighet.

5 Resultat

I det här avsnittet kommer resultatet av applikationens huvudfunktioner att visualiseras.

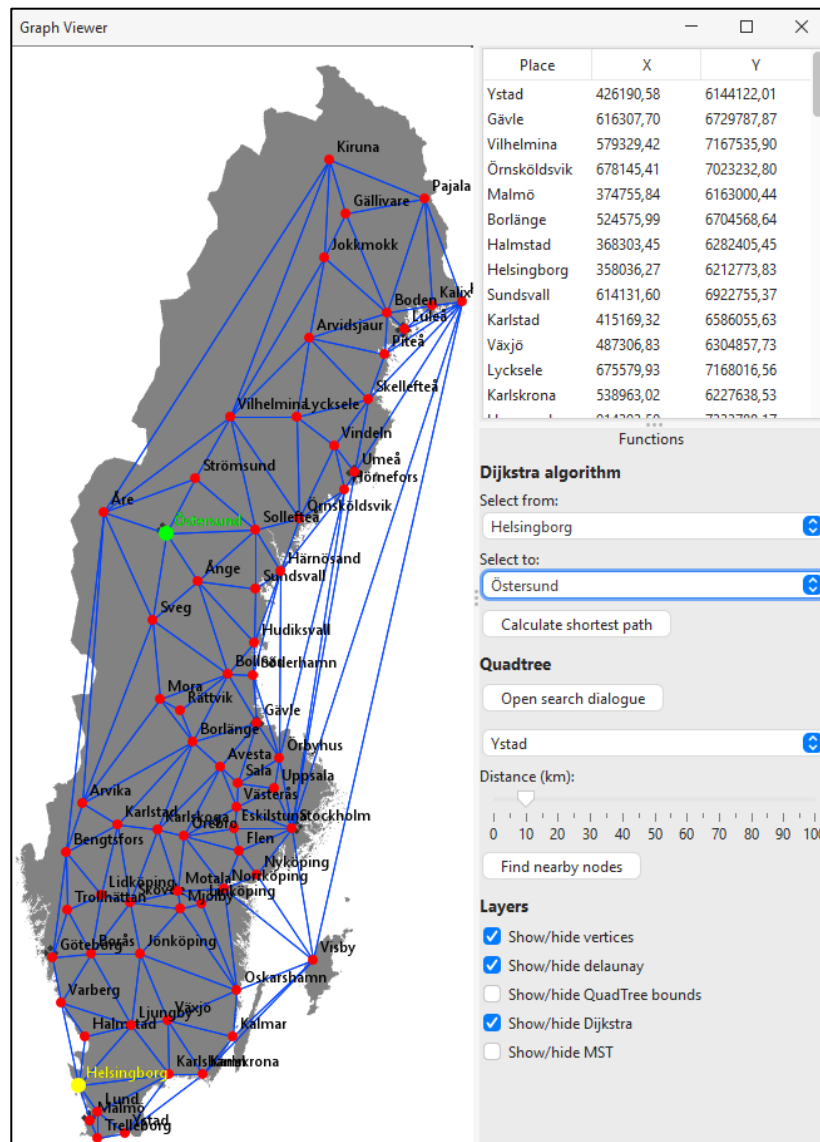
5.1 Kortaste vägen med hänsyn till vikterna (Dijkstra)

Prototypen innehåller till vänster en Sverige-karta med vertiser som representerar serverhallarna och i höger tabell visas koordinaterna för dem (se Figur 4). Under rubriken "Dijkstra algorithm" finns det en rullgardinsmeny där det går att välja från-punkten som ska användas i algoritmen och en andra rullgardinsmeny för till-punkten.



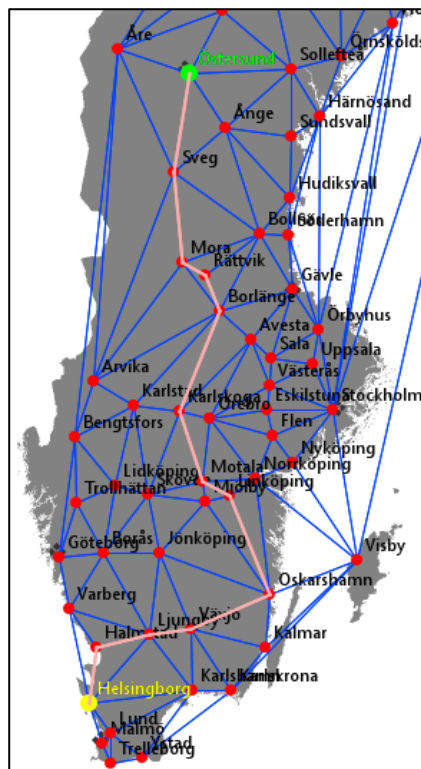
Figur 4. Hur applikationen ser ut vid igångsättningen.

Vill användaren beräkna algoritmen från exempelvis serverhallen i Helsingborg till Östersund, kommer det tydligt framgå i kartan att de vertex-punkterna har valts (se Figur 5).



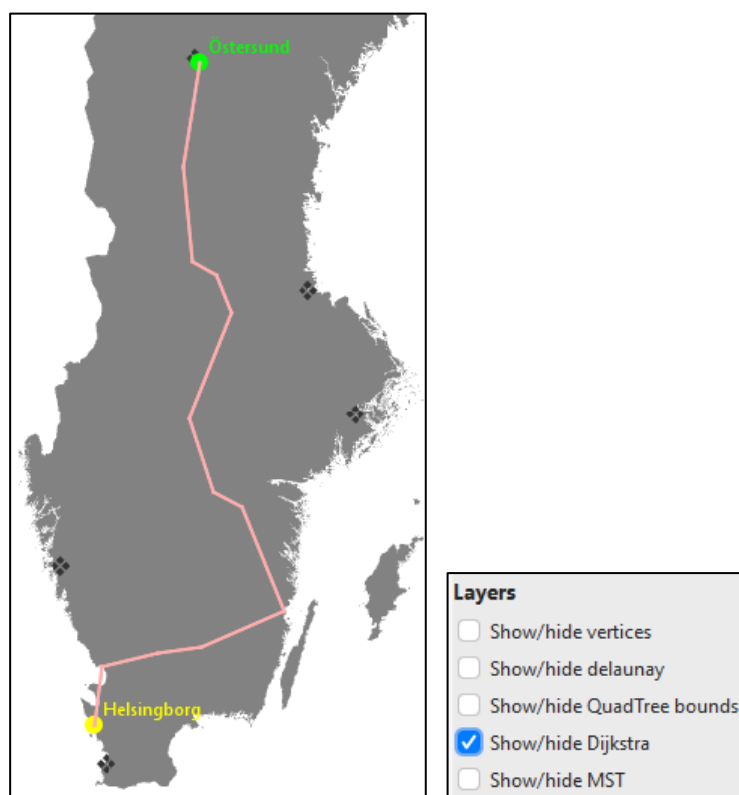
Figur 5. Applikationen när punkterna för Helsingborg och Östersund har valts för algoritmen.

Vill användaren sedan beräkna algoritmen klickar den på knappen “Calculate shortest path” och då ritas en linje ut mellan punkterna som visar kortaste vägen med hänsyn till vikten (se Figur 6).



Figur 6. Visar resultatet från algoritmen.

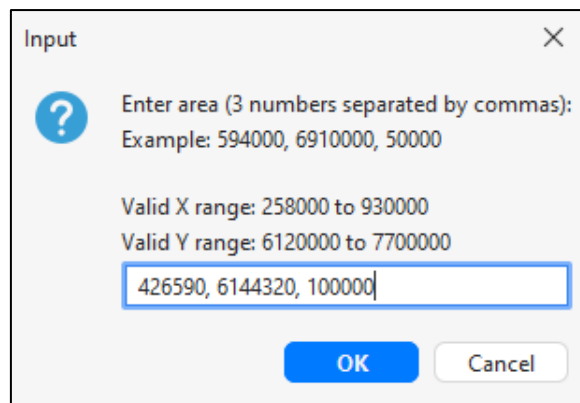
Användaren kan också välja att tända och släcka olika lager (nere till höger) för en tydligare visualisering (se Figur 7).



Figur 7. Visualisering när endast resultatet för Dijkstras algoritm är tätt.

5.2 För en position inom ett område, ge information om närmaste serverhallar (Quadtree)

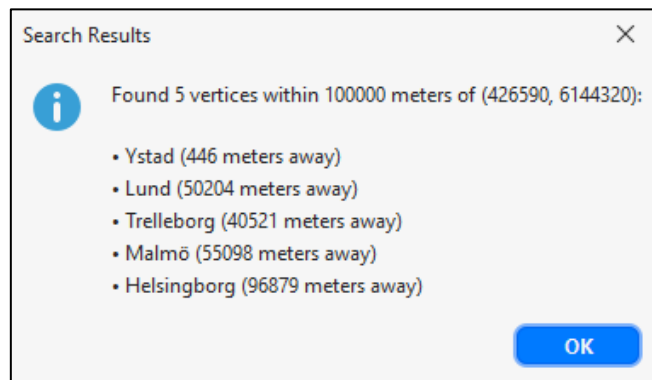
Quadtree funktionen i applikationen kan användas på flera sätt, dels med en manuell sökning och även en enklare sökning genom att välja en serverhall att utgå ifrån. Den manuella sökningen är skapad och tänkt för lite mer specifik och avancerad användning. För att använda den kan användaren klicka på knappen “Open search dialogue” och då kommer det upp en ruta med lite information om sökområdets räckvidd och vad användaren kan mata in för värden (se Figur 8). Tabellen med serverhallarna i det övre högra hörnet kan också användas som referens för koordinaterna vid sökning. Eventuella felinmatningar hanteras med ett popup-fönster som visar ett felmeddelande.



The dialog box is titled "Input" and contains a question mark icon. It prompts the user to "Enter area (3 numbers separated by commas):" with an example: "594000, 6910000, 50000". It also specifies the "Valid X range: 258000 to 930000" and "Valid Y range: 6120000 to 7700000". A text input field contains the coordinates "426590, 6144320, 100000". At the bottom are "OK" and "Cancel" buttons.

Figur 8. Användaren kan mata in tillåtna värden för en centrumkoordinat och områdets räckvidd.

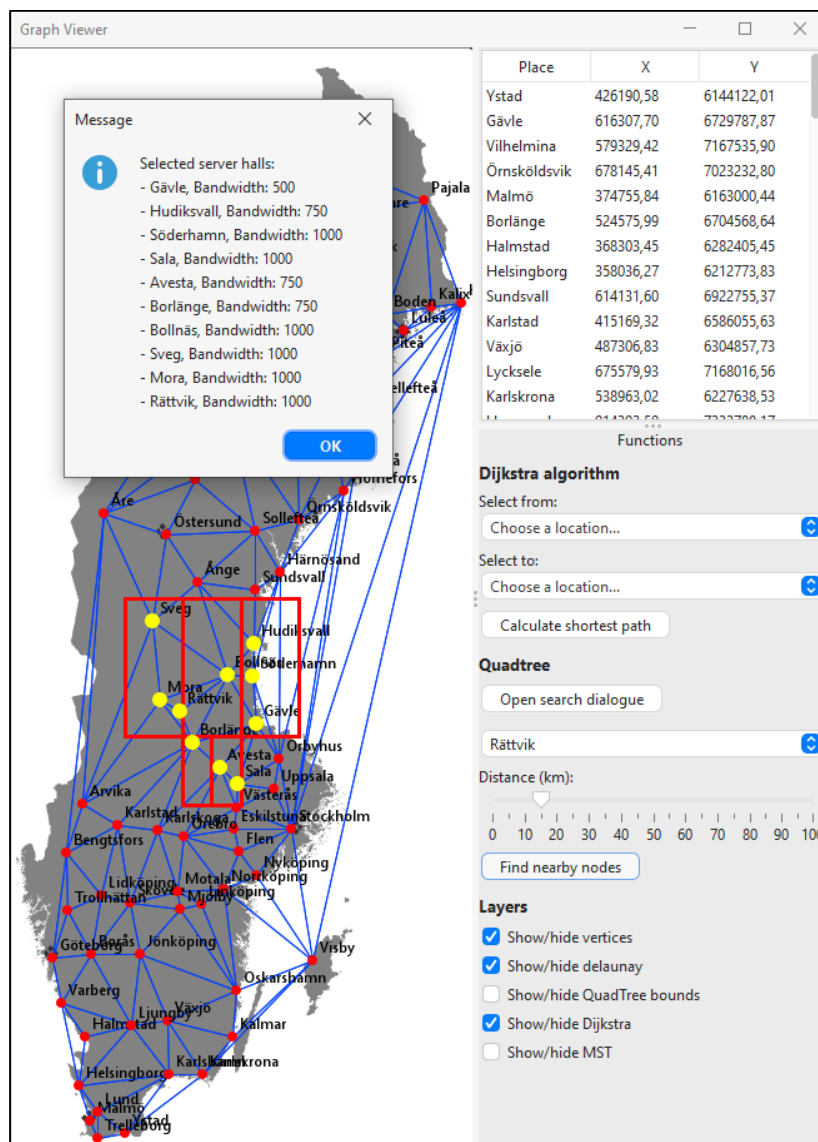
När en sökning för ett område har gjorts, kommer ett fönster att visas med information om de serverhallar som ligger inom sökområdet som användaren har matat in (se Figur 9).



The dialog box is titled "Search Results" and contains an information icon. It states "Found 5 vertices within 100000 meters of (426590, 6144320):". Below this is a list of server hall locations and their distances: Ystad (446 meters away), Lund (50204 meters away), Trelleborg (40521 meters away), Malmö (55098 meters away), and Helsingborg (96879 meters away). An "OK" button is at the bottom right.

Figur 9. En lista med serverhallarna som ligger inom sökområdet, som har specificerats med centrumkoordinat och storleken på området (10 000m).

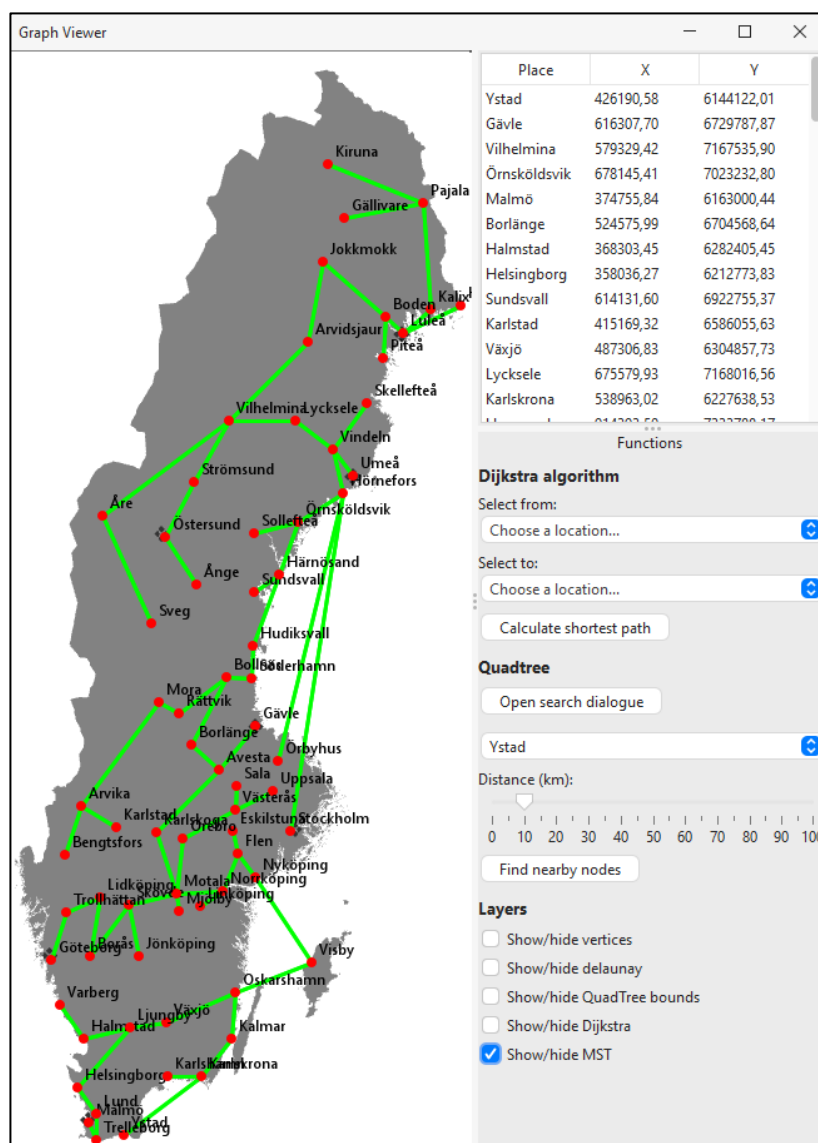
Vill användaren i stället göra en enklare sökning som visar i grafen vilka serverhallar som ligger inom sökområdet, kan den välja en serverhall, ställa in storleken på sökområdet med slidern och sedan klicka på knappen “Find nearby nodes”. Då kommer det upp en ruta med information om serverhallarna inom området och i kartan markeras de valda punkterna med gul färg och det ritas ett Quadtree i deras område (se Figur 10).



Figur 10. Resultatet av funktionen när serverhallen för Rättvik har valts, med ett sökområde på 15km.

5.3 Visualisering av ett Minimum Spanning Tree (MST)

En extra-funktion i applikationen är att det i grafen är möjligt att visa ett MST för serverhallarna på kartan (se Figur 11). För att se trädet kan användaren välja att endast kryssa i checkboxen för MST.



Figur 11. Visar ett MST för de inlagda serverhallarna i kartan.

6 Utvärdering

Det inledande utvecklingsarbetet drog inspiration från laborationernas uppgifter eftersom även prototypen skulle innehålla bland annat en grafstruktur och en prioritetskö med tillhörande heap. Under arbetets gång har datastrukturerna utvecklats i nya riktningar för att nå lösningar som uppfyller kraven för detta projekt. Datastrukturerna syftar på Quadtree, Min-heap, prioritetskön, grafen (Edge och Vertex). I kombination med research och föreläsningssanteckningar, har även AI också varit en del av arbetet för att bolla idéer. Algoritmerna dijkstra och MST har en stark koppling till prioritetskön, vilket i sin tur har en stark koppling till Min-heap. Därav har diskussion med AI varit givande och gett en större förståelse vilket har underlättat utvecklingsprocessen. Generellt sagt, har AI varit ett nyttigt verktyg som skapar en miljö där kopplingarna mellan datastrukturen och algoritmerna kan effektiviseras, se även bilaga A, figur A1.

Huvudfunktionerna som skulle implementeras i applikationen fungerar som förväntat. Med användningen av SWEREF-koordinater ger det en bra grund för användning av positions-data i Sverige.

När det kommer till tidstesterna så utmärker sig Quadtree. Täta kluster av vertiser kan leda till stora djup i strukturen, vilket tar mer tid att skapa och sedermera att söka inom. Fler tester bör utföras med kontrollerat data. Nuvarande prestanda är god för högst 800 serverhallar.

6.1 Vidareutveckling

Förslag på förbättringar i framtiden är först och främst bättre testdata, i synnerlighet för Quadtree. Ett tillägg för att visualisera sökområdet för Quadtree är också viktigt för användarvänligheten. Vidare kan sägas att visualiseringen av djupet inte är tillräcklig i prototypen. Här finns möjligheter för att färglägga varje nivå, och på detta sätt tydliggöra hur träd-strukturen är skapad, vilket underlättar för vidare testning. Att Quadtree skriver sitt eget objekt i grafen är nästan implementerat men behöver slutföras. När Quadtree genererar ett eget grafobjekt öppnas också möjligheten till att gömma eller visa det på kartbilden.

När det kommer till Dijkstra saknas välgrundade tidstester då algoritmen kräver en större slumpfaktor än vad som ges i den nuvarande prototypen. Alla rutter startar från samma nod, den första noden i listan av vertiser, vilket är det som omöjliggör empiriska mätningar.

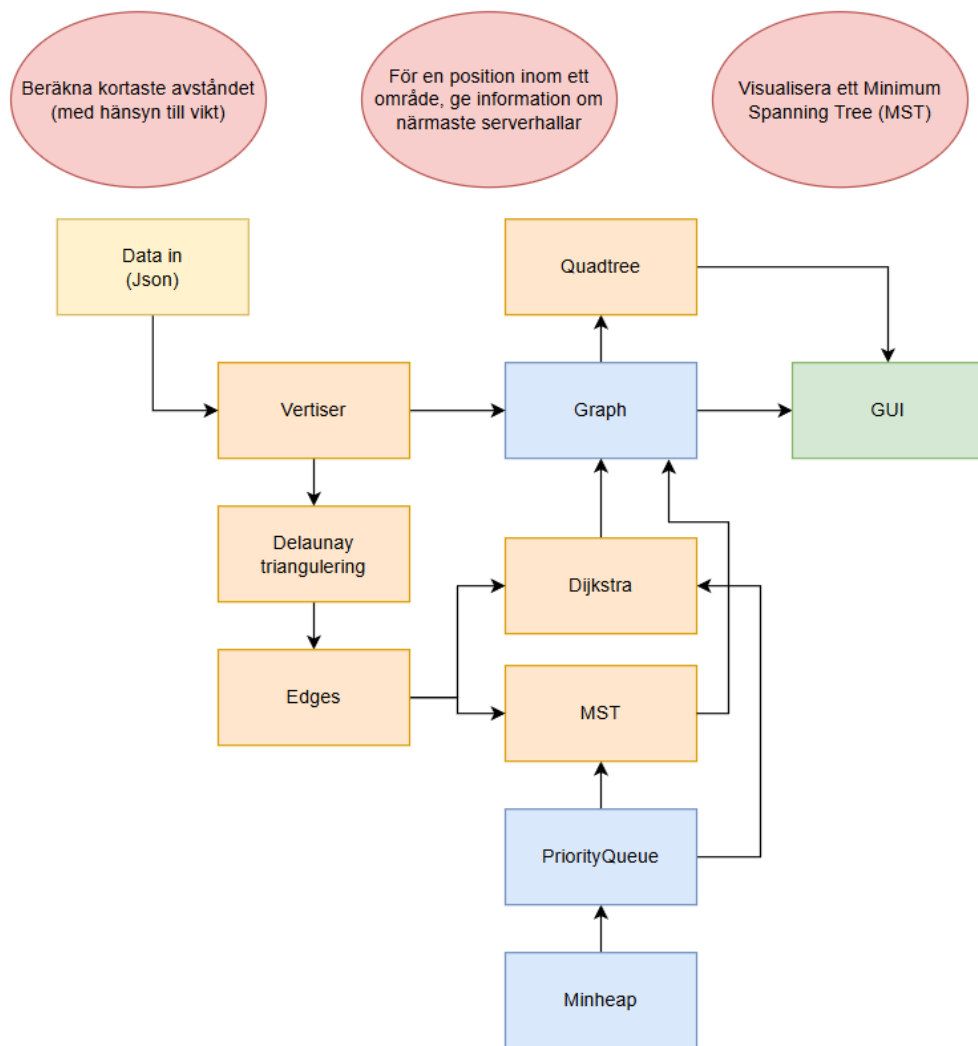
MST skulle kunna vidareutvecklas genom att det ska gå att välja var roten ska utgå ifrån. Det skulle kunna ge ett effektivare resultat beroende på vilket område som är intressant.

7 Referenser

Weiss, M.A. (2010). *Data Structures and Problem Solving Using Java (4th ed)*. Pearson education.

Graphs and Paths, 13, 501-544.

Bilaga A



Figur A1. Användningsfall och flödesdiagram.

Bilaga B

| Tabell för kvoter | | | | $kvoten = \frac{T(2n)}{T(n)}$ |
|-------------------|----------|------------------|----------------|-------------------------------|
| Storlek (n) | Delaunay | Quadtree (skapa) | Quadtree (sök) | MST |
| 100 → 200 | 1,58 | 1,37 | 1,85 | 1,59 |
| 200 → 400 | 2,03 | 1,62 | 1,73 | 2,61 |
| 400 → 800 | 2,17 | 2,74 | 1,84 | 2,61 |
| 800 → 1600 | 2,10 | 3,31 | 2,14 | 1,79 |
| 1600 → 3200 | 2,07 | 3,33 | 3,02 | 2,12 |
| 3200 → 6400 | 2,22 | 2,14 | 2,88 | 2,40 |

Figur B1. Resulterande kvoter efter tidstester.