
Forord

Denne rapport indeholder gruppe Js analyse og beskrivelse af den implementerede løsning af *BFST Project*. Implementationen og rapporten blev udarbejdet i perioden fra d. 24/2-2014 til d. 21/5-2014. Vi vil gerne takke Christian Nøhr Rasmussen, for at vi måtte benytte os af hans skalerede kystlinjedata, som vi anvender, når vi viser Krak-datasættet. Vi vil gerne takke Magnus Jacobsen og Sune Debel for god vejledning.

1.1 Omkring Bilag

I rapporten har vi vedlagt følgende bilag:

- **Kravsspecifikation:** Den af kursusansvarlige udleverede PDF.11.1
- **Krakdokumentation:** Den medfølgende dokumentation til vores datapakke.11.2
- **OSM-dokumentation:** Link til den officielle wiki for Open Street Map-formatet.11.3
- **Arbejdsfordeling:** Beskrivelse af gruppens arbejds- og ansvarsfordeling.11.4
- **logbog:** Vores log over den ugentlige proces.11.5
- **Arbejdsblade:** Vores log over det udførte arbejde.11.6
- **Samarbejdsaftale:** Vores aftale omkring hvordan vi ville håndtere vores arbejdsforløb.11.7
- **Forventningstabeller:** Oversigt over resultater fra programafprøvningen.11.9

Indholdsfortegnelse

1	Forord	1
1.1	Omkring Bilag	1
2	Introduktion til Problem	5
3	Analyse af design	6
3.1	Visualisering	6
Vejfarver og -tykkelser	6	
Panorering	6	
Zoom	7	
Visning af nærmeste vejnavn	7	
OpenStreetMap eller Krak	8	
3.2	Rutevejledning	8
Sidebar	8	
4	Problemanalyse	10
4.1	MVC	10
4.2	Filtrering og klargøring af data	10
Krak	10	
Open Street Maps	11	
4.3	Grupper	12
4.4	Datastruktur	13
Linjeplacering	14	
Nærmeste vej	15	
4.5	Navigation	15
Dijkstra	16	
A*	17	
Ruteomkostninger	18	
4.6	Adressegenkendelse	18
4.7	Tekstuel repræsentation af ruter	19
Måleenheder	19	
Retning af vejskifte	19	
4.8	Visning af kort	20
Cache og forskydning	21	
Anden løsning	22	
Tre planer	22	

INDHOLDSFORTEGNELSE	3
Udsnit	23
Centraliseret lagring af tiles	23
Gruppering	24
Tegning af tiles	27
Tråde	28
Object pools	28
Fake zoom	29
5 Implementationsbeskrivelse	30
5.1 Model	30
5.2 View	30
5.3 Controller	30
5.4 Tests	31
5.5 Utility	31
5.6 Data	31
6 Afprøvning	32
6.1 Afprøvning	32
Metode	32
Ækvivalensklasser	32
Dokumentation	32
7 Brugervejledning	33
Kortmanipulation	33
Rutevejledning	34
8 Evaluering af programmet	35
8.1 Hastighed	35
8.2 Fejl og mangler	35
8.3 Brugervenlighed	36
9 Evaluering og refleksion over proces	37
9.1 Evaluering og reflektion over proces	37
Projekt del I	37
Projekt del II	38
Opsummering	39
10 Konklusion	40
11 Bilag	41
11.1 Kravsspecifikation	41
11.2 Krakdokumentation	48
11.3 OSMdokumentation	62
11.4 Arbejdsfordeling	62

Implementering	62
Rapport	62
11.5 Logbog	62
11.6 Arbejdsblade	72
11.7 Samarbejdsaftale	93
11.8 Grene i Prioritetskø	95
11.9 Forventningstabeller	98
PriorityQueue	98
Vector	102
Box	106

Introduktion til Problem

Vores opgave går ud på at lave et interaktivt danmarkskort med indbyggede zoom og panorering-funktioner. Kortet skal have alle veje og stier i det udleverede Krak-datasæt markeret og være i stand til at vise navnet på den vej, der er nærmest på musemarkøren. Programmet skal være i stand til at vise den hurtigste rute fra to udvalgte punkter på kortet, og skrive en rutevejledning til dette. Programmet skal være intuitivt og være responsivt nok til ikke at skabe gene. Ud over Krak-datasættet skal programmet også være kompatibel med Open Street Map (OSM) datasæt. For den præcise kravsspecifikation se 11.1.

Ud over de obligatoriske krav, har vi implementeret følgende features:

- **Udregn hurtigst rute til alle punkter fra givent startpunkt:** Brugeren kan finde den hurtigste vej, ved at højreklikke på sit startpunkt, og herefter vælge slutdestination ved at venstreklirkke der. Ruten findes med det samme uden forsinkelser.
- **Skjulbar sidebar:** Vi har lavet en sidebar, som kan skjules, så brugeren kan se mest muligt af kortet, når dette er ønskværdigt.
- **Dynamisk vejbredde:** Vejene angives med en bredde, der ændres alt efter zoom-niveau, så brugeren tydeligt kan se hovedveje på kortet.
- **Løbende Navneapproksimation:** Når brugeren prøver at finde en rute, ved at indtaste en adresse, vil programmet løbende foreslå adresser, der minder om det skrevne.
- **Kystlinje:** Vi har tegnet Danmarks kystlinje.

Analyse af design

I dette afsnit fordybes der i de designvalg der er blevet foretaget til brugergrænsefladen, samt overvejelser der lå bag de valg.

3.1 Visualisering

Vejfarver og -tykkelser

Når en bruger benytter sig af kortet, er det en forudsætning, at brugeren skal kunne skelne mellem forskellige vejtyper. Særligt relevant bliver dette, når ruteplanlægningsfunktionen benyttes, da brugeren skal kunne skelne mellem motorveje og hovedveje. Den oplagte løsning er at adskille vejtyperne fra hinanden ved at tildele vejene forskellige farver, hvilket også er et af kravene stillet i kravsspecifikationen.

Et kort, hvor vejene differentieres efter farver, siger ikke nok om deres vejtyper. Det vil kræve at brugeren på forhånd kender til hvilke farver, der er knyttet til vejtyperne.

En større adskillelse af vejtyperne kan opnås ved at tegne dem med forskellige vejtykkelser. Med dette design skabes et udtryk der reflekterer virkeligheden, hvilket gør det lettere for brugeren at skelne mellem vejene.

Panorering

Panorering er en central funktion i et kortprogram. Vi overvejede 2 forskellige måder at implementere panorering på. Dette kunne enten være at trække i kortet eller trykke på visuelle piletaster. En beslutning, om at der ikke skal være et visuel interaktionssystem, i form af taster og zoom knapper, blev truffet tidligt i processen. Et visuelt interaktionssystem kunne blænde brugeren for at benytte andre måder at interagere med programmet på. Derudover benytter en bruger sig typisk af en mus. Det kunne fremstå som værende forstyrrende, hvis der er piletaster på skærmen.

På baggrund af ovenstående overvejelser valgte vi, at panorering skal foregå ved at trække i kortet. Tastaturets piletaster kan også benyttes, der udgør et alternativ, som er en naturlig måde at navigere i kortet, som ikke er forstyrrende.

Zoom

En af de helt store funktioner er zoom-funktionen. Der skulle her tages højde for, hvordan denne skulle komme til udtryk hos brugeren samtidig med at gøre det bekvemt at benytte.

Der kan zoomes på følgende måder:

- Zoom med taster.
- Zoom med mus
- Rektangulær zoom.

Når der zoomes med taster eller mus, så sker det relativt til centrum af det aktuelle udsnit. Den anden mulig løsning havde været, hvor der zoomes relativt til musemarkør. Da vi ikke kunne implementere denne løsning. Von Malthe synes også det er en irriterende zoomfunktion, så det er ok.

Hvis brugeren hurtigt vil zoome ind på et specifikt udsnit af kortet, så er de to første zoom metoder ikke tilstrækkelige. Med rektangulær zoom kommer man hurtigere til det ønskede udsnit. Det er en særdeles gunstig løsning til hurtig og målrettet zoom.

Synlige vejtyper

Ved et zoomniveau, hvor eksempelvis hele Danmark er synligt, er det kun relevant at få fremvist motorveje og hovedveje. Man kan med fordel undlade at synliggøre mindre og ubetydelige veje i sådan et zoomniveau. Brugeren er ikke interesseret i at se de mindre veje, og det skaber et mere overskueligt billede at undlade dem.

Ved større zoomniveau synliggøres de mindre veje, da man må antage, at det er hvad brugeren ønsker at få fremvist.

Dynamisk vejtykkelse ved zoom

For yderligere at give programmet et overskueligt udtryk, har vi tilføjet en mindre visuel funktion. Ved zoom vokser vejtykkelsen dynamisk. Det giver zoomfunktionen en lækker detalje og programmet et pænt look.

Visning af nærmeste vejnavn

Musemarkøren vil altid vise det nærmeste vejnavn.

Vi havde to løsningsforslag til at vise nærmeste vejnavn.

1. Vise vejnavn ved musemarkøren.
2. Vise vejnavn i en statusbar.

Den første løsning kan forstyrre brugeren, da vejnavnene konstant vil ligge over kortet, nær musemarkøren, og derved være i vejen. I statusbaren er det ikke lige så iøjenvældende.

OpenStreetMap eller Krak

Et af de obligatoriske krav til programmet er, at brugeren kan benytte sig af enten Krak- eller OpenStreetMap-datasættet. Det oplagte er, at brugeren får denne valgmulighed ved opstart af programmet. I tilfælde af at brugeren ønsker at skifte datasæt i et kørende program, kunne man give brugeren for dette. Vi valgte dog at nedprioritere denne mulighed. For at skifte mellem datasæt må brugeren lukke programmet ned, genåbne det og derfra vælge det datasæt, brugeren ønsker at benytte.

3.2 Rutevejledning

Rute planlægning

Brugeren kan få planlagt en rute på følgende to måder:

- Klikke to steder på kortet.
- Indtastning af udgangspunkt og destination i sidebaren.

At klikke to steder på kortet er en indlysende måde at få lavet en rutevejledning. Præcisionen af denne form for søgning kan variere, alt afhængig hvilket zoomniveau brugeren befinner sig på. Hvis der er zoomet betydeligt ud, falder præcisionen markant. Når der er zoomet tilpas ind, er dette en udemærket måde at planlægge sin rute på. En udvidelse til denne metode, ville være at markere klikkene med start og slut punkter. For en mere nøjagtig ruteplanlægning kan man benytte sidebaren.

Uanset hvilken måde der benyttes, kunne implementering af husnumre øge præcisionen til ruteplanlægning.

Sidebar

I sidebaren indtastes udgangspunkt og destination, ligeledes udskrives rutevejledningen i denne.

Sidebaren fylder meget, og hvis den er fremvist konstant, kan det ødelægge oplevelsen af programmet. Det er ikke bekvemt at have sidebaren fremme, når man navigere i kortet.

Det skal derfor være muligt at skjule sidebaren, når den ikke bruges, og få den fremvist, når den skal benyttes. Til dette er der en diskret knap nede i venstre hjørne.

Adressegenkendelse

Programmet vil ikke kunne beregne en rute, hvis der er blevet indtastet et vejnavn, der ikke er defineret i datasættet. Derfor har vi implementeret, at når der indtastes data i tekstfeltet, vil der automatisk komme en rullemenu til syne. Menuen indeholder mulige adresser, som minder om brugerens indtastning. Der vil i menuen altid være fem muligheder præsenteret

i prioriteret rækkefølge, da det vil virke ubekvemt for brugeren at få præsenteret for mange muligheder.

Dette er med til at give brugeren hurtigere søgning og samtidig fjerne brugerens bekymringer om fejlindtastninger.

Problemanalyse

4.1 MVC

Vi har valgt at designe vores program ud fra tanken omkring Model-view-controller-designmønstret. Dette sikrer en stærk adskillelse mellem funktionalitet, input og repræsentation. Dette hjælper også med et mere modulært miljø, der giver mere afprøvbar kode, forståelse for en klassens opgaver såvel som brugbarhed i senere projekter.

4.2 Filtrering og klargøring af data

Vi udviklede to moduler til at filtrere og klargøre data — et for hvert datasæt. Fælles funktionalitet for de to moduler er at:

- Frasortere unødvendige data.
- Ensrette formatet som data gemmes i.
- Konvertere koordinater således at de ligger inden for et 1000 gange 1000 koordinatsystem med udgangspunkt i øverste venstre hjørne.
- Maks x og y værdier gemmes til brug ved indlæsning.
- Rette op på evt. fejl og mangler i datasæt.

Krak

For nogle af vejstykkerne i datasættet var hastigheden 0 km/t, hvilket vi anser som en fejl, i de data vi har modtaget. Dette viste sig at være problematisk, eftersom vores rutealgoritme bruger vejstykernes hastigheder til at beregne den hurtigste vej.

For at afhjælpe dette problem valgte vi at beregne hastigheden ud fra vejstykets længde og estimerede køretid med følgende formel:

$$s = \frac{\frac{l}{1000}}{\frac{t}{60} \cdot \frac{1}{1.15}}$$

Hvor s er hastigheden i km/t, l er længden i meter og t er den estimerede køretid i minutter. Formlen tager højde for at Krak har lagt 15% til deres estimerede køretider.

Denne løsning er ikke perfekt, da det havde været optimalt at aflæse alle hastigheder frem for at udlede dem, men var et nødvendigt onde for at opnå den ønskede funktionalitet.

De kystlinjedata vi fik fra en anden gruppe var angivet som en serie af forbindelser mellem koordinater: $(x_1, y_1), (x_2, y_2)$. Disse data bliver i modulet splittet ud i en række punkter og vejstykke.

Open Street Maps

Da vi havde designet vores første del af programmet ud fra Krak-datasættet, valgte vi at benytte denne datastruktur som udgangspunktet for vores program. Derfor havde vi et behov for at omforme OSM-datasættet til dette format. Der var dog flere forhindringer ved dette. For det første var mængden af data i OSM-datasættet enormt. Der var så meget data, at ingen af vores mappedatamater var i stand til at kunne læse hele datasættet på én gang. Derudover stemte datasættet ikke fuldkommen overens med retningslinjerne i dokumentationen, f.eks rækkefølgen af forskellige værdier i en node. Til sidst var datastrukturen anderledes bygget op end Krak. Alle disse hindringer skabte en følelse af, at man ikke kunne se skoven for bar' (quad)træer, at information om dataet var utilgængelig. Dataet kunne først ses, efter at man havde filtreret i den, og det var svært at vide, hvad man skulle filtrere efter. I starten forsøgte vi filtrere dataet hurtigst muligt, ved at gemme den ønskede data i hukommelsen, men hukommelsesforbruget blev hurtigt for stort. Herefter skiftede til en langsommere løsningsmodel, hvor vi benyttede swapfiler på fastpladelagereret. Generelt lykkedes det os at få store dele af den ønskede data med over, men der var dog nogle ting, vi ikke kunne få til at virke i fyldestgørende grad:

- **Vejhastigheder:** Det mest sigende mærkat(tag) vi kunne finde, var knyttet til vej-objekter(way) i OSM. Desværre var det kun 1/3 af vejene der havde dette mærkat tilknyttet. Resten havde ingen hastighedsmarkering. Vi fandt i dokumentationen et tegn på, at man kunne gå ud fra, at der var visse hastigheder i byzoner og lignende. Desværre kunne vi ikke finde belæg for, at sådanne zoner var tydeligt markeret i datasættet. En potentiel løsning på dette kunne være at benytte de punkter, der markerer byer, og ud fra dem lave en antagelse om, at alle vejstykker(edges), der er inden for x afstand fra dette punkt, er inde for bygrænsen. Men denne antagelse ville være baseret på løse gæt, og derved heller ikke give et sandfærdigt indtryk. Istedet har vi givet dataet uden hastighedsmærkat standardværdier efter vejtype.
- **postnumre knyttet til veje:** Vi skulle benytte denne information i tilfælde at flere veje havde samme navn. Desværre understøttede OSM ikke dette, da OSM havde knyttet deres postadresser til adressepunkter, der ikke var forbundet med veje. Igen ledte vi efter en potentiel zonemarkering, men kunne ikke finde belæg for, at sådanne zoner var tydeligt markeret i datasættet. En potentiel løsning ville være at finde nærmeste adressepunkt på en vejdel, og så benytte det postnummer, som der er knyttet til dette punkt. Denne løsning har vi ikke implementeret.

Vi har valgt begrænse os til de punkter, der er en del af en vej, og ud af de veje beholder vi kun dem, der indeholder et vejtypemærkat. Det er nødvendigt at lave et tilvalgssortering frem for en frasortering, da der er så store datamængder. Metadata i OSM-datasættet var generelt gemt i en række mærkater, der består af en nøgle og en værdi. Antallet af mærkatnøgler er dog utrolig stort, og der er ikke nogle form for praktisk begrænsning af brugen af nøglerne. Vi har efter længere research valgt at gøre brug af følgende nøgler:

- **name:** Dette mærkat indeholder vejnavnet
- **highway:** Dette mærkat indeholder vejtypen(motorvej, landevej, gågade etc.)
- **maxspeed:** Dette mærkat indeholder den højeste tilladte transporthastighed
- **natural=coast:** Dette mærkat indeholder kystlinjen

For at få et scope 0->1000 på vores xy-koordinatsæt, måtte vi benytte følgende formel på vores UTM-koordinater $newCoordinate = (UTMCoord - minUTMCoord) * (1000 / (maxUTMCoord - minUTMCoord))$

Derudover spejler vi længdebredderne, så verdenshjørnerne passer til den almindelige kortstandard.

4.3 Grupper

Vi valgte at sortere vores data ud i større grupper. Dette skaber mulighed for tilpasning og større optimering. De valgte grupper er:

0. Motorveje
1. Hovedveje, heriblandt motortrafikveje
2. Stier og markveje
3. Gågader
4. Søfartsveje
5. Kystlinje
6. Andet

Det er ikke betimeligt at få fremvist alle veje uafladeligt når der er zoomet langt ud. Gågader, stier og markveje er først relevante at få fremvist, når der er zoomet tilstrækkeligt ind. På baggrund af dette, er det relevant at opstille en løsning, der adskiller detaljegraden på kortet i forhold til zoomgraden.

Ud fra den viden vi har, om hvilke grupper vi har at gøre med, skal der tilknyttes farvedata til grupperne. Vejtyperne skal også differentieres af deres vejtykkelse, da veje

ligeledes har forskellige tykkelser i virkeligheden. Da det ikke er hensigtsmæssigt at definere alle veje til at have ens vejtykkelser, skal der tilknyttes noget ekstra data om vejtykkelsen af de forskellige grupper.

Ydermere valgte vi at udvide funktionaliteten ved at implementere "dynamiske vejtykkelser ved zoom". Dvs. at vejtykkelserne vokser og skrumper dynamisk alt afhængig af om der zoomes ind eller ud.

Da vejtykkelsen skal tilpasse sig relativt til zoomgraden, kan man blive fristet til at benytte sig af formlen:

$$\text{dynamiskvejtykkelse} = \frac{\text{vejtykkelse}}{\text{zoom}} \quad (4.1)$$

Formlen gav nogle voldsomme resultater i form af enorme vejtykkelser ved tilstrækkelig højt zoomniveau. Et alternativ var at man for hver af de respektive grupper definerede vejtykkelserne ved forskellige zoomniveauer. Det ville give et mindre dynamisk look, men tilstrækkeligt opfylde funktionaliteten.

Det viste sig, at en passende løsning var, at lade vejtykkelserne vokse med 5% af hvad formlen 4.1 gav. Dette kommer til udtryk ved følgende formel:

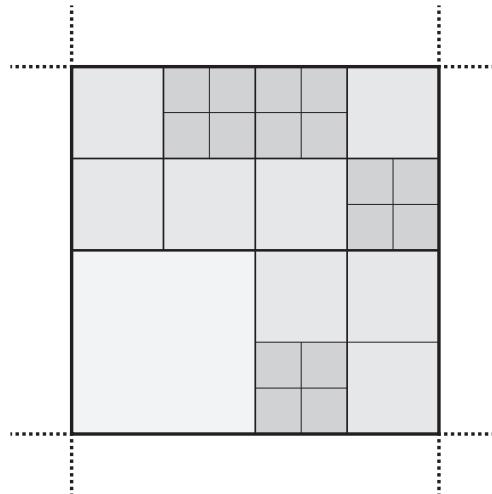
$$\text{dynamiskvejtykkelse} = \text{vejtykkelse} * (1 + (5\% * \frac{\text{vejtykkelse}}{\text{zoom}})) \quad (4.2)$$

4.4 Datastruktur

I implementeringen af ethvert kort må den bagvedliggende datastruktur nøje overvejes, da denne i høj grad afgør applikationens ressourcekrav. Overvejelserne omkring datastrukturen involverede blandt andet muligheden for kun at indlæse specifikke dele af dataet samt søgeegenskaber. Det givne datasæt består af knudepunkter og tilhørende vejsegmente, som forbinder to af disse knudepunkter. Knudepunkterne indeholder information om deres individuelle placering, mens vejsegmente er forbundet med en række forskellige oplysninger såsom vejnavn og -type. Da kortdataet består af vejsegmente, er det også af betydning for valget af datastruktur. Værd at overveje var også datastørrelsen — datakilden fra Krak indeholder over en million knudepunkter, og disse skal på ethvert tidspunkt under programkørslen kunne tilgås, hurtigt.

Flere forskellige datastrukturer besidder de nævnte egenskaber: heriblandt en sorteret tabel, et k-d-træ og en quadtree-struktur. Den sorterede tabel brillerer med en ukompliceret implementering. Quadtree og k-d-træ er komplicerede at implementere, men udmærker sig ved deres simple datasegmentering samt hastige indlæsning af rektangulære områder.

K-d-træet splittes ved medianelementet og opdeler træet i to subproblemer. Der splittes igen omkring median elementet i hvert subproblem, hvortil der opstår to subproblemer per split. I en quadtree-struktur, derimod, indeholder hver *quad* fire andre quads — dens børn. Individuelle quads kan acceptere et begrænset antal elementer, hvorefter ethvert fremtidigt forsøg på indsættelse vil blive videregivet til dennes børn rekursivt. Når et rektangulært område efterspørges, foregår dette ligeledes rekursivt. En quad, som



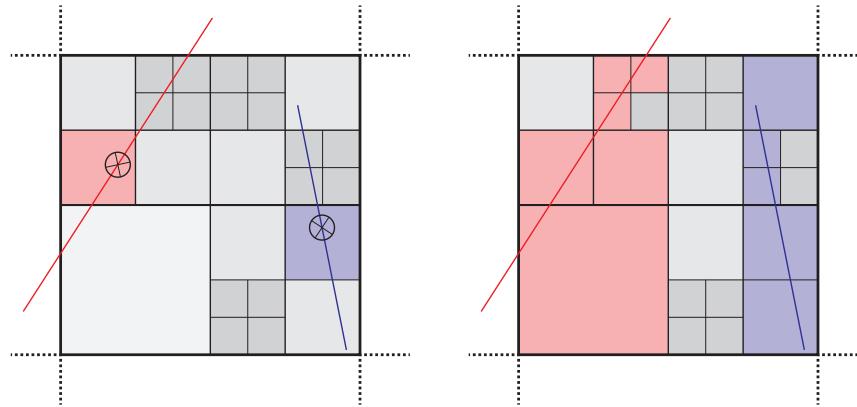
Figur 4.1 – Eksempel på udsnit af en quadtree-struktur hvori dybden af de individuelle quads fremhæves af disses farve, fra lys til mørk.

modtager en forespørgsel, vil videregive forespørgelsen til sine børn, og derefter returnere det tilbagemeldte resultat sammenlagt med sine egne elementer.

Det vurderes at en sorteret liste ville være for langsom til vores behov. Det var svært at sætte fingeren på en konkret forskel i ydeevnen mellem k-d-træet og quadtree-strukturen. Grundet et langt større kendskab til quadtree-strukturens virkemåde, sås denne således som den mest passende løsning. Vi fandt det vigtigt at vælge og udarbejde datastrukturen hurtigt, da udviklingen af resten af programmet afhæng deraf.

Linjeplacering

Da en quadtree-struktur arbejder med et koordinatrum ved indsættelse og forespørgsel af data, er det nødvendigt at skabe en sammenhæng mellem et vejsegment og koordinatsystemet. Indledningsvist løste vi dette problem med blot at styre indsættelsen af en linje efter dens centrum. (4.2 tv.). Dette forårsagede dog et større problem, hvor relativt lange llinger ikke blev tegnet, da disses centrum kunne være uden for den besøgte quad. Vores umiddelbare løsning til dette var, at udvide den forespurgte rektangel den halve længde af den længste linje i den adspurgte quad. Således var det garanteret, at samtlige linjer, som skar det efterspurgt rektangel, faktisk blev fundet. Denne implementering medførte, at andre dele af programmet, som gjorde brug af quadtree-strukturen, måtte regne med at modtage vejstykker uden for det forespurgte rektangel. Det ville senere vise sig at udgøre et reelt problem for programmets hastighed under tegneprocessen. Derfor ændrede vi måden hvorpå linjer indsættes i træet til en mere ideel tilgang, hvor enhver quad som berøres af en linje, indeholder denne. (4.2 th.). Indhentning af et dataområde er således garanteret at returnere de linjer, som enten ligger indenfor eller skærer det eftersøgte område — og intet andet.



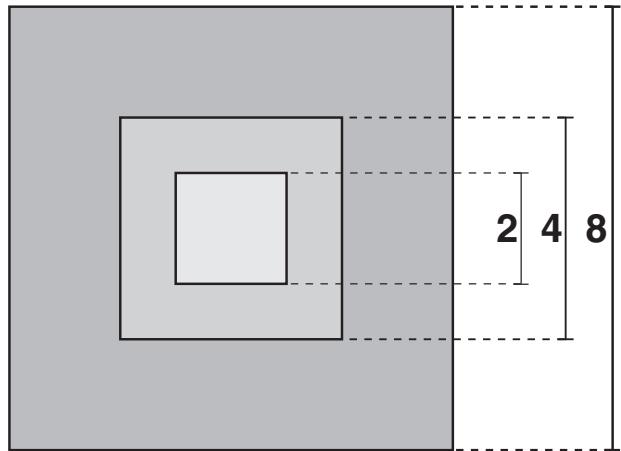
Figur 4.2 – Venstre: Linjens centrum afgør hvilken quad den placeres i. Højre: Linjen placeres i enhver quad, som denne skærer.

Nærmeste vej

For at opfylde det formelle krav om at synliggøre navnet på vejen nærmest markøren, samt at muliggøre angivelse af udgangspunktet og destinationen for navigation med markøren, måtte en algoritme udarbejdes. Udfordringen bestod i at gennemsøge quadtree-strukturen i et begrænset område omkring markøren. Hertil opstod et løsningsforslag, i at finde den dybeste quad, som markøren holdes over, og iterere over dennes indhold. Det medførte dog et problem, når markøren befandt sig tæt på en quads horisont, fordi det da ikke kunne garanteres, at den nærmeste vej befandt sig i den fundne quad. I stedet for at løse dette, ved at implementere en måde hvorpå nabo-quads kunne findes, valgte vi at udnytte allerede eksisterende funktionalitet til at løse det oprindelige problem. Ved at forespørge den øverste quad om vejstykker i en begrænset rektangel uden om markøren, og derefter gennemløbe resultatet, kunne det nærmeste vejstykke hurtigt findes. Fandt en forespørgsel ikke nogen vejstykker fordobledes det forespurgte rektangel, og der forsøgtes igen. Ved at fordoble søgearealet sænkes antallet af nødvendige iterationer, uanset om markøren er meget langt væk fra fastlandet eller midt i hovedstaden.

4.5 Navigation

I implementeringen af rutennavigation måtte forskellige kortest-vej algoritmer overvejes. En passende algoritme til formålet udnytter en graf over kortdataet, som bliver genereret under opstart. Derudover må algoritmen også inddrage faktorer som længden og hastighedsgrænsen af et vejstykke i dens vurdering af den optimale rute. Da vi ønskede at opfylde det udvidede krav, om at destinationen skulle kunne ændres kontinuerligt og flydende, til rutennavigation, var det oplagt at anvende en algoritme, som genererer et kortest-vej-træ. Med et sådant træ er det nemlig muligt at udlede den korteste vej fra træets startknude og til enhver anden knude i grafen i konstant tid.

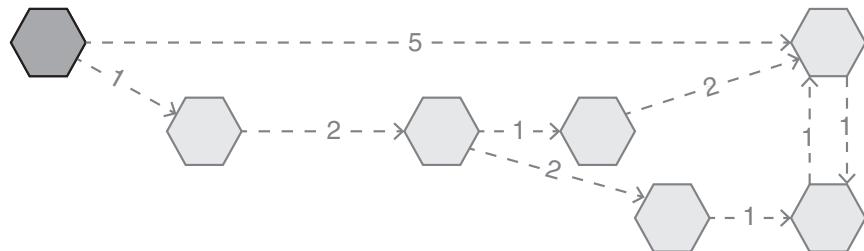


Figur 4.3 – Søgerektanglet fordobles for hver iteration, indtil mindst én linje er fundet.

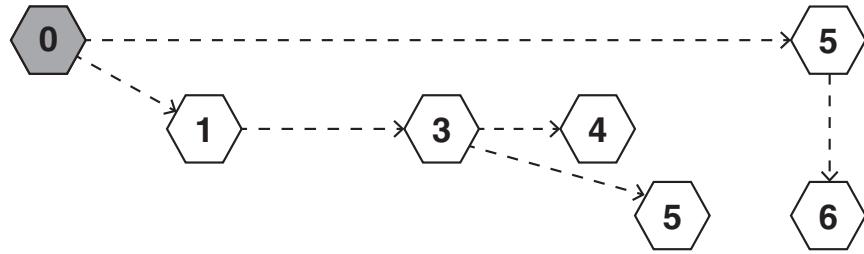
Dijkstra

Sådanne kortest-vej-træsalgoritmer trumfes af E. W. Dijkstras grafsøgningsalgoritme. Dijkstras algoritme tager udgangspunkt i en startknude, hvorfra det resterende træ genereres. Hver af dennes tilstødende knuder besøges. Et besøg består i at sammenholde den kendte afstand, mellem startknuden og den besøgte, med den nye afstand. Den nye afstand ses som den kendte afstand mellem startknuden og den besøgende knude, sammenlagt med afstanden mellem den besøgende og den besøgte knude. Har en knude aldrig været besøgt, er afstanden til denne blot angivet som $+\infty$, hvorfor ethvert nyt forslag er en forbedring. Besøgsrækkefølgen varetages af en prioritetskø, som prioriterer de tilstødende knuder med lavest afstand til startknuden højst. Betragtes 4.4 og 4.5 ses en graf og det resulterende træ, hvorfra den korteste vej mellem startknuden og enhver anden knude er tydelig.

Indledningsvist implementerede vi Dijkstras ud fra en antagelse om, at det netop ville være hurtigst at generere et kortest-vej-træ, når slutknuden måtte kunne flyttes frit. Dijkstras var som forventet resursekrævende under genereringen af træet, hvilket med en graf i Open Street Maps størrelsesorden viste sig som et uacceptabelt tidsforbrug.



Figur 4.4 – Eksempel på en vægtet orienteret graf.

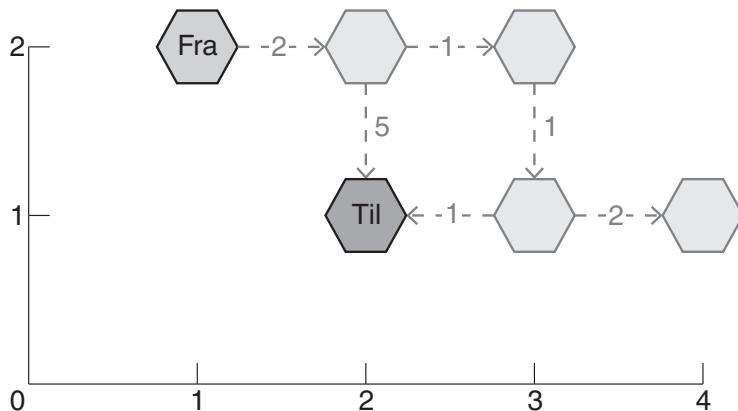


Figur 4.5 – Det af Dijkstras resulterende kortest-vej-træ med udgangspunkt i den fremhævede startknude. Distancen mellem de individuelle knuder og startknuden ses angivet.

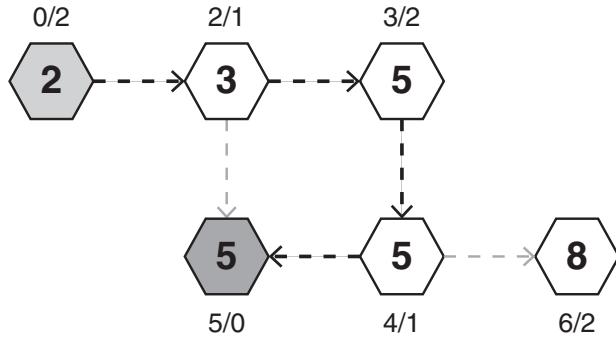
A*

En anden algoritme, som derfor blev inddraget i overvejelserne, var A*-algoritmen. Denne genererer ikke, ligesom Dijkstras, et kortest-vej-træ, men finder blot den korteste vej mellem en given start- og slutknude. Da A* udvider funktionaliteten af Dijkstras algoritme, er deres adfærd meget tilsvarende. Hvor Dijkstras blot igagtager den traverserede afstand mellem den besøgte knude og startknuden, vurderer A* også en heuristik $h(x)$ for afstanden til *slutknuden*. Besøgsrækkefølgen varetages stadigvæk af en prioritetskø, men nu prioriteres der efter funktionen $f(x) = g(x) + h(x)$, hvor $g(x)$ er afstanden mellem den besøgte knude og startknuden, som set i Dijkstras.

I et forsøg på at forøge hastigheden af navigationsudregningen, implementerede vi således A* og sammenholdte dennes tidsforbrug med Dijkstras. A* implementeringen viste sig i praksis at være langt det hurtigste valg, selvom ændring af destinationsknuden var mere krævende. I en kortapplikation, hvor både udgangspunktet og destinationen må forventes at blive ændret ved hver kørsel af algoritmen, er A* det suveræne valg.



Figur 4.6 – Eksempel på en orienteret vægtet graf opstillet i et koordinatsystem som angiver distanceheuristikken. Heuristikfunktionen $h(x)$ angives som Manhattan-afstanden ved $|x_2 - x_1| + |y_2 - y_1|$.



Figur 4.7 – Den af A* resulterende korteste vej mellem start- og slutknuden. Værdierne i knuderne angiver $f(x)$, mens venstre- og højresiden af skråstregen angiver henholdsvis $g(x)$ og $h(x)$. (Det er bestemt ikke en selvfølge, at samtlige knuder besøges, som er tilfældet i dette eksempel.)

Ruteomkostninger

Kvaliteten af både Dijkstra og A* algoritmernes resulterende korteste vej afhænger, af hvilke informationer et linjestykkes vægt defineres ved. Et intuitivt svar er vejens egentlige længde, hvorfor dette også blev anvendt i de indledende implementeringer. Det viste sig dog hurtigt at være en uhensigtsmæssig tilgang, da den korteste vej ikke nødvendigvis er den hurtigste. Derudover forårsagede det også store mængder vejskift, hvilket er uønsket for både længden af rutebeskrivelsen i programmet, samt for lethedten af den egentlige kørsel. For at løse dette udregnedes¹ i stedet tidsforbruget for et linjesegment.² Krak arbejder med en trafiktillægstid på 15% i deres datasæt, hvilket vi har ladet vores implementering inspirere af.

4.6 Adresseegenkendelse

For ikke blot at kunne definere udgangspunktet og destinationen for rutenavigation med markøren, var det nødvendigt at implementere manuel adresseindtastning. Implementeringen byggede indledningsvist på opretholdelsen af et symboltabel med adresser som nøgler og vejstykker som værdier. Således kunne en efterspurgt adresse hurtigt forbindes til en række vejstykker. Præcisionen af brugerens indtastning var altaførende for løsningens effektivitet, da det indtastede skulle stemme fuldstændig overens med adressen i datasættet. Derfor fandt vi det nyttigt at implementere adresseegenkendelse, så korrektheden af den indtastede adresse ikke begrænsede brugeren. Til formålet implementerede vi en algoritme for *Levenshtein-distance*, som sammenligner to ord bogstav for bogstav og returnerer distancen mellem disse. Distancen mellem to ord forøges med én, hver gang et bogstav slettes eller indsættes, og med to hver gang et bogstav substitueres. Udover denne normale

¹Kraks datasæt indeholder køretider på veje, men vi valgte at omgå disse for også at imødekomme andre datasæt. (herunder OSM).

² $Tid = \text{laengde}/\text{hastighed} * 1.15$

distancebetragtning, ønskede vi at muliggøre omkostningsfri autofuldførelse af adresser. Derfor forøger indsættelse ikke distancen, såfremt de foregående bogstaver var ens mellem ordene — altså når distancen i forvejen er nul. Resultatet af dette er, at distancen mellem ”rued lang” og ”rued langgaards vej” er nul.

Da antallet af unikke adresser er ganske højt, var det køretidsmæssigt tungt at sammenligne en indtastning med samtlige kendte adresser. Dette var ganske mærkbart under indtastning, da det forsagede en kort forsinkelse efter hver karakterindtastning. Vi løste indledningsvist dette ved først at påbegynde adressegenkendelsen, når brugeren ikke havde foretaget en indtastning i et kort tidsrum. Senere optimerede vi denne løsning ved simpelthen at placere kørslen af adressegenkendelse på en separat tråd, som ikke fastlåste brugergrænsefladen under kørsel. Herved kunne en ny adressegenkendelse påbegyndes (og erstatte den gamle), hver gang brugeren foretog en ændring i indtastningen. Dette førte til en mere flydende indtastning samt minimal ventetid på adresseforslag.

4.7 Tekstuel repræsentation af ruter

Måleenheder

For at vise længden af et vejstykke på en komprimeret og brugervenlig måde, valgte vi at angive længden i forskellige enheder afhængigt af vejstykets længde. Udgangspunktet var at vise længden i meter, uanset længde, men det viste sig ikke at være optimalt, da det både fyldte meget og var unødig præcist. Derfor vedtog vi følgende:

- Længde vises i meter, hvis længden er mindre end 1000 meter. $L < 1000$.
- Længde vises i kilometer med to decimaler, hvis længden er mere end 1000 meter. $L \geq 1000$.

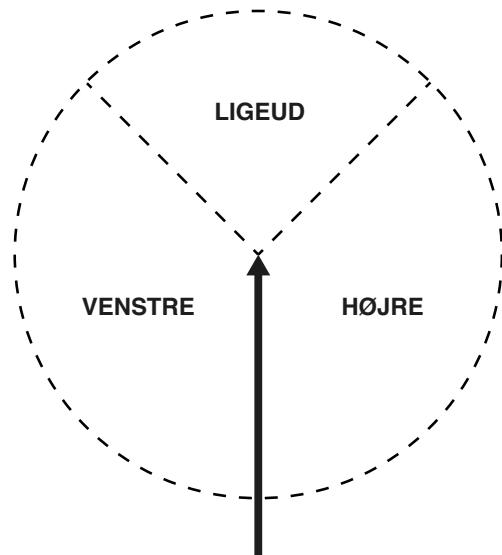
Retning af vejskifte

Vores første idé var at beregne vinklen mellem de to veje, der udgør det enkelte vejskifte, og på den måde afgøre om der er tale om et sving, og i så fald hvilken retning der drejes i. Vi var ikke særlig glade for denne løsning til at starte med, da vi fandt det problematisk, at vi selv skulle definere de forskellige typer sving frem for at aflæse det fra vores data. Denne skepsis forsvandt, efter at vi blev enige om, at vi ikke kunne finde noget data om typen af vejskifte. Valget faldt altså på denne metode, eftersom vi ikke havde noget værdigt alternativ.

Vi besluttede os for følgende definitioner for de forskellige typer vejskifte:

- Vinklen mellem to vejstykker bliver beregnet således, at rotation med uret angives med positivt fortegn, og rotation mod uret angives med negativt fortegn.
- Hvis vinklen er mindre end 45° stor, uanset fortegn, er der ikke tale om et sving. $|A| < PI/4$.

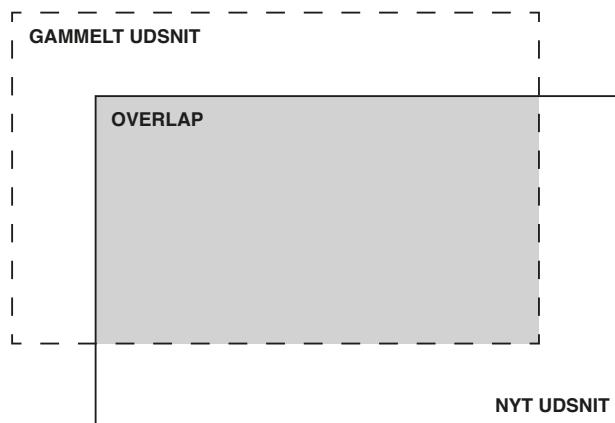
- Hvis vinklen er større end 45° og er positiv er der tale om et højre sving. $A \geq PI/4$.
- Hvis vinklen er mindre end 45° og er negativ er der tale om et venstre sving. $A \leq -PI/4$.



Figur 4.8 – Definition af typer af vejskifte.

4.8 Visning af kort

Når brugeren navigerer kortet og bliver på samme zoomniveau, vil der være dele af kortet, som allerede er tegnet, der skal tegnes igen. Der er et overlap mellem det gamle og det nye udsnit af kortet (figur 4.9).



Figur 4.9 – Overlap mellem gammelt og nyt udsnit.

Til at starte med tegnede vi hele udsnittet, inklusiv overlap, men det viste sig ikke at være hurtigt nok, og responstiden blev problematisk. Det blev derfor besluttet, at vi havde brug for en mere effektiv løsning, der kun tegner den del af et nyt udsnit, der ikke overlapper med det gamle udsnit.

Dette afsnit er skrevet på et tidspunkt, hvor vores implementering af quadtræer ikke var optimal. En af ulemperne ved denne implementation var, at den returnerede en margin af vejstykker, uden om den rektangel der var efterspurgt. Denne margin udgjorde i mange tilfælde helt op til 70-80% af de returnerede vejstykker. Efter optimeringen er denne ulempe ikke eksisterende.

Analyserne i dette afsnit tager udgangspunkt i implementeringen før optimeringen, og der er derfor lagt stor vægt på, at ovennævnte ulempe eksisterede. Analyserne er dog fortsat relevante, men gevinsten ved at bruge de nævnte løsninger er langt mindre, end den var før optimeringen.

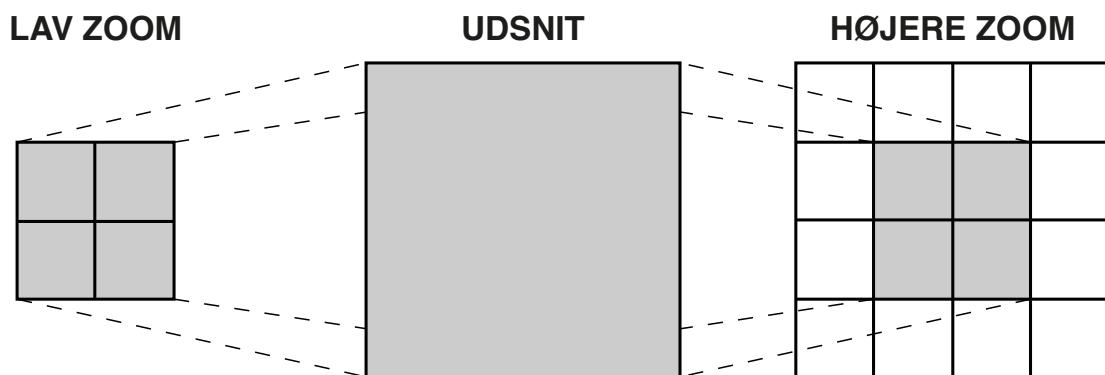
Cache og forskydning

Den første løsning, der blev overvejet, var at gemme et billede af det gamle udsnit til en cache og forskyde det, således at det lå rigtigt i forhold til det nye udsnit. Efterfølgende tegnes den del af udsnittet, som ikke er inkluderet i overlappet. Denne løsning var meget simpel, og vi formoder, at den havde været nem at implementere. Den havde dog en række klare ulemper, hvilket førte til, at vi ikke implementerede den:

- Hvis brugeren navigerer tilbage til et udsnit, de allerede har set, skal dette udsnit tegnes igen.
- Der skal gemmes billeddata til en cache, hver gang udsnittet ændrer sig, således at denne data kan gentegnes som en del af det næste udsnit.
- De udsnit, der skal tegnes, er tynde, hvilket skaber stor ineffektivitet som følge af den måde, som vejstykker hentes fra quadtrees.

Anden løsning

Den anden løsning vi overvejede, og den vi endte med at implementere, var inspireret af online kort-programmer som for eksempel Google Maps og OSM. Det centrale koncept i denne løsning er at dele kortet op i en masse små rektangler bestående af bitmap data (tiles). Når der er zoomet langt ud, udgøres kortet af få tiles, og når der er zoomet langt ind, udgøres kortet af mange tiles (figur 4.10). Når der skal tegnes et udsnit, tegnes de tiles, der er indenfor udsnittet.



Figur 4.10 – Antallet af tiles, som kortet udgøres af, er afhængigt af zoom.

Denne løsning løser helt eller delvist problemerne ved den ovenstående løsning:

- De tegnede data er strukturerede, og det er dermed muligt at gemme dem, og vise dem igen, når det er relevant.
- Der skal kun gemmes billeddata til et lager af tiles, når der bliver tegnet nye tiles.
- De udsnit, der skal tegnes, er firkantede og har en vis størrelse, hvilket gør problemet med overflødig data mindre.

Tre planer

Når løsningen med tiles anvendes, er der tre rektangulære planer, der skal tages højde for:

- **Model:** 1000 enheder langt på den længste led. Ændrer aldrig størrelse.
- **Tiles:** Dimensioneret efter et bitmap der ville kunne indeholde tiles for hele det aktuelle zoomoniveau. I praksis bliver sådant et bitmap aldrig oprettet, men blot tiles som repræsenterer fragmenter af dette bitmap. Ændrer størrelse afhængigt af zoom-niveau.
- **Skærm:** Samme dimensioner som det vindue som kortet vises i på skærmen. Ændrer størrelse når brugeren ændrer størrelsen på vinduet.

Udsnit

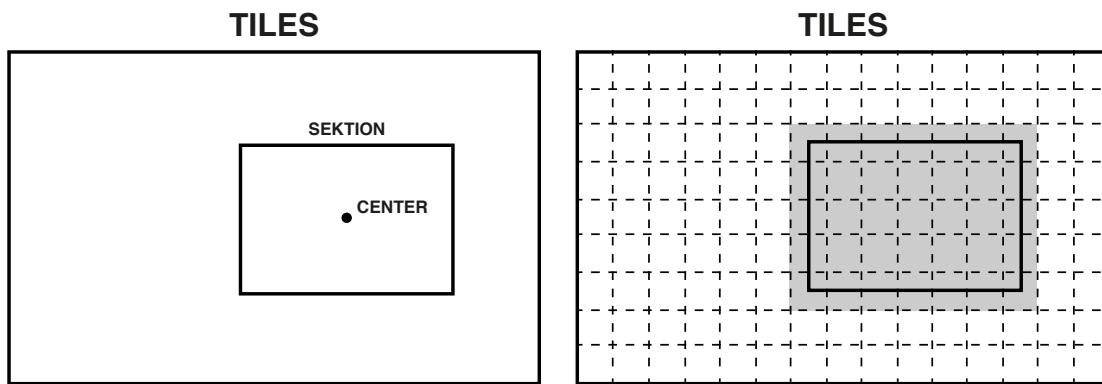
Programmet opdaterer løbende to variable — center og zoom — for at holde styr på hvilket udsnit som brugeren kigger på. Center definerer hvilket punkt, udsnittet er centreret omkring, og zoom definerer hvor stor andel af kortet, der er synligt i udsnittet. Vi valgte denne løsning som et alternativ til at definere udsnittet som en boks med absolute koordinater, for at opnå øget fleksibilitet.

Hvilke tile skal vises

Når kortet skal tegnes, skal det afgøres hvilke tiles, der skal vises på skærmen.

Først beregnes det, hvor udsnittet befinner sig på tiles. Dette gøres ved at tage en boks, der har samme dimensioner som skærm, og forskyde den i forhold til tiles, således at den nu er centeret omkring center. Denne boks kalder vi for *sektion* (figur 4.11, venstre).

Dernæst beregnes hvilke tiles, som sektion overlapper (figur 4.11, højre). Det er disse tiles, som skal vises på skærmen.



Figur 4.11 – De tiles, der overlapper med sektion, skal vises.

Centraliseret lagring af tiles

De løsninger, som vi er inspireret af, gør to ting markant anderledes end vi gør:

- De begrænser antallet af zoom niveauer. Hvis man ikke gør dét, er der uendeligt mange zoom-niveauer, og dermed uendeligt mange tiles der potentielt skal lagres.
- De har en vedligeholdt database over tiles, hvor tiles løbende hentes fra. Ved at gøre dette undgår de at tegne nye tiles, når et udsnit efterspørges — I stedet hentes tiles fra den centrale database.

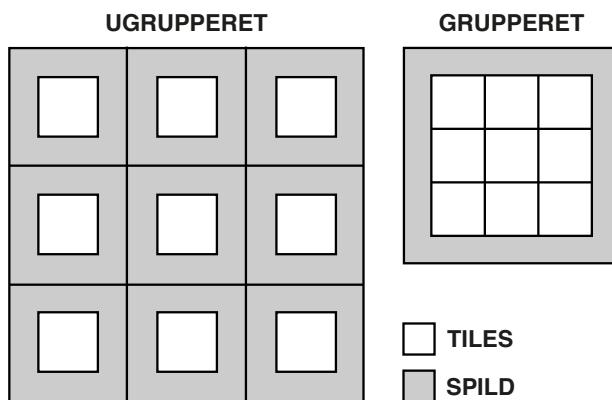
Vi overvejede en kort overgang at implementere en database løsning, men besluttede os for, at det ville være for stor en opgave. Dette ville være en oplagt mulighed for

senere optimering af programmet, og kunne både implementeres med en lokal eller ekstern database.

Eftersom vi ikke anvender en databaseløsning, giver det ikke mening at reducere antallet af zoomniveauer. Da tiles alligevel skal tegnes løbende, ville dette kun være en fordel, hvis brugeren vendte tilbage til et udsnit på et zoomniveau, som vedkommende allerede havde set. Da vi ikke tænker at dette scenarie opstår særlig tit, fravælge vi at begrænse antallet af zoomniveauer, da det i vores øjne ville forringe programmets funktionalitet unødvendigt.

Gruppering

Som nævnt tidligere skal tiles tegnes løbende som følge af at vi ikke har en database, hvor vi kan hente dem fra. Til at starte med tegnede vi tiles enkeltvis, hvilket viste sig at være meget ineffektivt. En stor andel af de vejstykker der hentes fra modellen, og efterfølgende skaleres og tegnes, er slet ikke indenfor tilen. Dette problem kan reduceres hvis tiles så vidt muligt grupperes og tegnes i rektangulære blokke. Hermed bliver mængden af spild minimeret (figur 4.12).



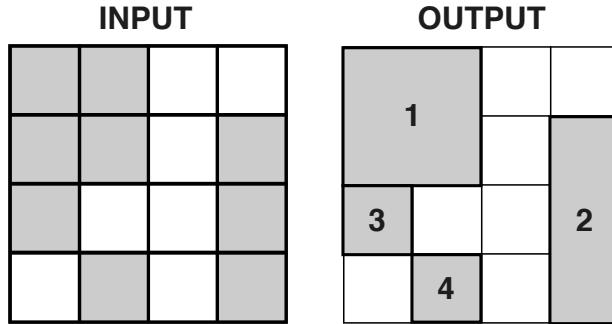
Figur 4.12 – Spild reduceres hvis tiles grupperes i rektangulære blokke.

Derudover er det væsentligt hurtigere at hente vejstykker fra quadtræerne, hvis tiles grupperes i store rektangler, frem for at der hentes vejstykker for tiles enkeltvis. Grunden til dette er, at vores quadtræer er optimeret således at alle vejstykker, der er i en quad og dens børn, returneres, hvis den pågældende quad er fuldstændigt indenfor den boks, der efterspørges. Det grupperede rektangel bryder med et minimalt antal quads, og er derfor langt mere effektivt i mange tilfælde. Den største besparelse sker når der skiftes zoomniveau, og det nye udsnit indeholder et stort antal vejstykker, der er distribueret over mange quads.

Definition af algoritme

Vi havde brug for en algoritme, der kunne lave rektangulære grupperinger ud af de tiles, der endnu ikke er tegnet. Algoritmen tager en liste af tiles, der er sorteret rækkevis og

efterfølgende kolonnevis, som input. Algoritmen finder det størst mulige rektangel og tilføjer det til en liste over rektangler. Algoritmen gentager denne procedure, indtil alle tiles er grupperet, og returnerer en liste af rektangler (figur 4.13).



Figur 4.13 – Numrene angiver i hvilken rækkefølge rektanglerne optræder i listen af rektangler, som algoritmen returnerer.

Brute force

Vores første løsningsforslag var en simpel brute force algoritme. Indledningsvist opbygger algoritmen en tabel, hvor det markeres hvilke tiles, der mangler at blive tegnet. Den naive løsning til dette problem er en brute force algoritme, der undersøger alle mulige rektangler i denne tabel. Hvis rektanglet er det største indtil videre, og kun består af tiles der mangler at blive tegnet, gemmes rektanglets position. Denne procedure gentages, indtil alle mulige rektangler er undersøgt.

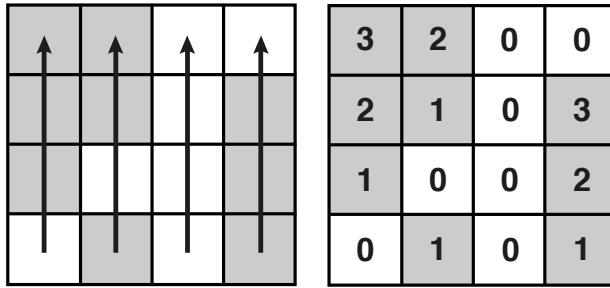
Det tager $\sim \frac{N^2 * M^2}{4}$ eller $O(N^2 * M^2)$ tid at gennemgå alle rektanglerne i et rektangel af $N * M$ størrelse. Da hvert af cellerne i rektanglerne skal undersøges, for at finde ud af om det enkelte rektangel udelukkende består af utedannede tiles, er den estimerede køretid for brute force algoritmen $O(N^3 * M^3)$.

Dynamisk programmering

Efter at have overvejet brute force algoritmen, fandt vi frem til en algoritme, der løser problemet mere effektivt ved hjælp af dynamisk programmering.

Ligesom brute force algoritmen opbygger den en tabel, hvor det markeres hvilke tiles, der mangler at blive tegnet. Herefter opbygges endnu en tabel over sammenhængende søjler, ved at traversere kolonnerne i tabellen nedefra og op og beregne højden af den enkelte søjle dynamisk (figur 4.14). Højden af en søjle er højden af den søjle, der er nedenfor plus én.

Herefter skannes rækkerne i tabellen fra venstre mod højre, og arealet af det størst mulige rektangel, der indeholder hver enkelt celle af tabellen, beregnes dynamisk ved hjælp af data fra søjletabellen. Undervejs i skanningen gemmes positionen af det største rektangel. Hermed en gennemgang af de første celler skridt for skridt (w er rektanglets højde, h er rektanglets bredde, og A er rektanglets areal):

**Figur 4.14** – Tabel af sammenhængende søger opbygges.

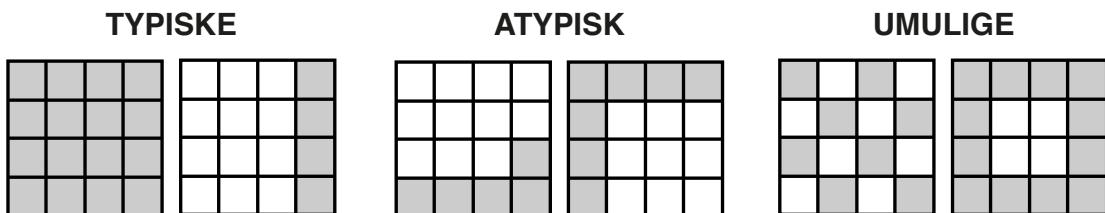
- **Celle (1,1)**: $w = 1$, eftersom vi har undersøgt én celle. $h = 3$ (aflæst i søger-tabellen). $A = 1 * 3 = 3$, og rektanglet gemmes som det største indtil videre.
- **Celle (2,1)**: $w = 2$, eftersom vi har undersøgt to celler. $h = 2$ ($2 < 3$). $A = 2 * 2 = 4$, og rektanglet gemmes som det største indtil videre ($4 > 3$).
- **Celle (3,1)**: $w = 0, h = 0, A = 0$ eftersom denne celle ikke er en del af en søger.

Afsluttende tilføjes det største rektangel til listen af rektangler, og de tiles der udgør rektanglet, markeres som tegnede.

Denne algoritme kører i $O(N * M)$ for et rektangel, der er $N * M$ stort, og blev derfor valgt til fordel for brute force algoritmen, der kører i $O(N^3 * M^3)$ tid.

Case analyse

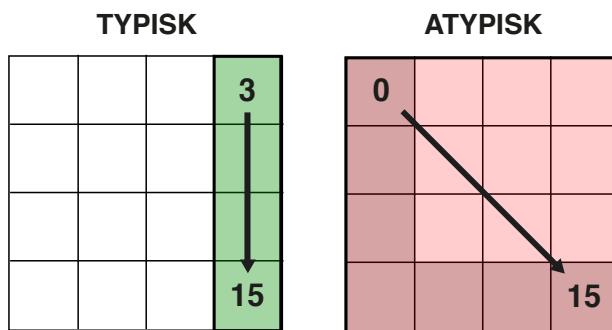
Efter at have implementeret ovennævnte løsning gik vi i gang med at analysere hvilke cases, der opstår ofte i praksis og hvilke, der sjældent eller aldrig opstår (figur 4.15).

**Figur 4.15** – Udvalg af typiske, atypiske og umulige cases.

Som det kan ses på figuren, har vi inddelt cases i dem, der er typiske og ofte opstår, dem der atypiske og sjældent opstår, samt dem der aldrig opstår, fordi de er umulige. De typiske cases opstår, når brugeren skifter zoomniveau eller bevæger udsnittet horisontalt eller vertikalt. De atypiske cases opstår, når brugeren bevæger udsnittet diagonalt. De umulige cases opstår aldrig, fordi der er huller i dem, hvilket ikke kan lade sig gøre, hvis brugeren bevæger udsnittet i en kontinuerlig bane.

Følgende algoritme løser problemet med gruppering af tiles i konstant tid i typiske cases, og falder tilbage på algoritmen fra afsnit 4.8 i tilfælde af at der er tale om en atypisk case (figur 4.16):

- Listen af tiles filtreres, og der oprettes en ny liste af tiles, som endnu ikke er blevet tegnet.
- Fordi tiles er sorteret, på den måde de er i den oprindelige liste, kan man lave et rektangel, der går fra den første til den sidste tile fra den nye liste.
- Hvis antallet af tiles, som rektanglet dækker over, er det samme som antallet af utedgennede tiles, er problemet løst i konstant tid (figur 4.16 til venstre). Hvis dette ikke er tilfældet, er den pågældende case atypisk (figur 4.16 til højre), og der faldes tilbage til den mere tidskrævende algoritme.



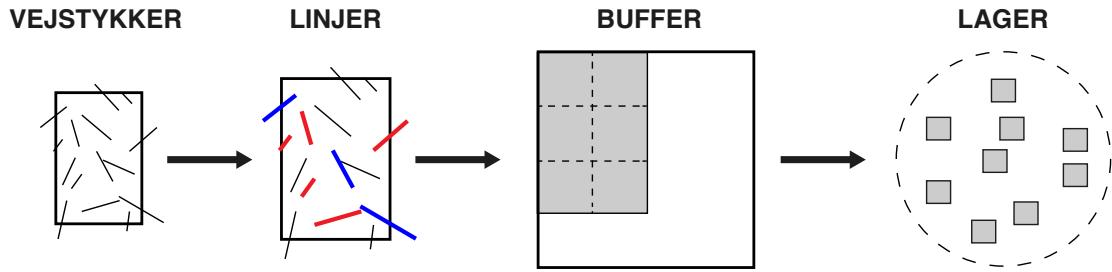
Figur 4.16 – Kørsel af algoritme på typisk og atypisk case.

Det er værd at nævne, at algoritmen potentielt giver et forkert svar, hvis en af de “umulige” cases opstår. Vi har ikke observeret nogen umulige cases, og vi formoder derfor, at de ikke opstår i praksis.

Altså endte vi med en algoritme, som er optimeret til de specifikke forhold der gælder for vores program, og som i en meget stor andel af kørslerne tager konstant tid. Det er ikke muligt at beregne en gennemsnitskøretid for algoritmen, eftersom vi ikke kan forudse, hvordan brugeren interagerer med kortet (hvilket afgør hvor mange atypiske cases, der opstår). Det eneste der kan garanteres er derfor, at køretiden i værste fald er den samme som algoritmen fra afsnit 4.8.

Tegning af tiles

Når listen af rektangler er fundet, gentages følgende procedure for hver rektangel (figur 4.17): Først beregnes en boks, som repræsenterer rektanglet på tiles-planet, og dernæst skaleres denne boks, således at den repræsenterer det samme rektangel på model-planet. Modellen forespørges efter vejstykkerne, der ligger inden for rektanglet, og koordinaterne for vejstykkerne skaleres til tiles-planet og gemmes som linjer. Linjerne tegnes til en buffer, og fragmenter af bufferen gemmes som tiles i et lager.



Figur 4.17 – Proces fra vejstykker til lagring af tegnede tiles.

Tråde

I forbindelse med tegning af tiles fandt vi det relevante at implementere trådning for at opnå følgende fordele:

- Brugergrænsefladen forbliver responsiv, imens nye dele af kortet tegnes på en tråd i baggrunden. Alternativet er, at opgaven kører på den samme tråd som brugergrænsefladen, hvilket resulterer i, at brugergrænsefladen er uresponsiv, indtil tråden er færdig med at tegne.
- Tråde giver mulighed for at tegne flere nye dele af kortet på samme tid ved at uddeletere opgaver til forskellige tråde, som køres på forskellige kerner i processoren.

Vi har valgt at anvende én tråd per kerne i forbindelse med tegning af tiles, hvilket har givet gode resultater på de maskiner, vi har testet programmet på. Hver gang der skal tegnes et rektangel, bliver der oprettet en ny opgave, som er klar til at blive eksekveret på en tråd. Hvis der er flere opgaver, end der er tråde, står opgaverne i kø, indtil der bliver en tråd ledig — ellers eksekveres de øjeblikkeligt.

Object pools

Object pools er et designparadigme, der anvendes, når man ønsker at genbruge objekter. Ved at bibringe referencer til objekterne sikres det, at de ikke bliver smidt ud af computerens hukommelse af *garbage collectoren*. Vi har valgt at genbruge instanser af følgende klasser i forbindelse med tegning af kortet:

- **Tråde**: Tråde er dyre at instantiere, og det giver derfor mening at genbruge dem så vidt muligt. Derudover er der allerede en implementering af object pooling af tråde i java, hvilket gjorde det nemt at poole tråde i forbindelse med vores program.
- **Buffers**: Bitmap buffers og de grafik-objekter, der skal tegne på dem, er dyre at instantiere, og det er derfor en fordel at genbruge dem.
- **Linjer**: Programmet har ofte brug for at have adgang til tusindvis af linjer, når der skal tegnes linjer til en buffer. På trods af at linjeobjekter er meget hurtige

at instantiere, viste det sig i praksis at være en stor optimering at genbruge dem. Dette formoder vi skyldes, at programmet har brug for store mængder af denne type objekter og genbruger dem mange gange.

Fake zoom

Når brugeren zoomer udsnittet ind og ud, smides alle de tiles, der tilhører det tidligere zoom-niveau, væk. Det er hermed nødvendigt at tegne et fuldt skærmbillede af nye tiles for det nye zoom-niveau. Dette kan tage relativt lang tid, især hvis oplosningen af brugerens vindue er stor, eller der er store mængder data indenfor udsnittet. Derfor er det i denne sammenhæng ikke praktisk at vente med at opdatere brugergrænsefladen, til at de nye tiles er tegnet. I stedet vises et skaleret øjebliksbillede af et tidligere zoom-niveau, imens tiles for det nye zoom-niveau tegnes i baggrunden. Ofte zoomes der mange gange på kort tid, og denne løsning gør det muligt hele tiden at give brugeren feedback på sit input.

Implementationsbeskrivelse

5.1 Model

Loader er ansvarlig for at indlæse data for det valgte datasæt og oprette instanser af Node og Edge objekter, der anvendes som datastrukturer for henholdsvis punkter og vejstykke. Ligeledes er loader ansvarlig for at indsætte Edges i QuadTrees, som gør det hurtigt at finde de Nodes og Edges der ledes efter.

Loader opretter flere forskellige QuadTrees der indeholder grupperinger af Edges. Groups er ansvarlige for at holde styr på hvilke typer af Edges, der hører til hvilken gruppe, og hvilke egenskaber hver enkelt gruppe har. Derudover anvendes Node og Edge objekterne også til generering af både en symboltabel over adresser og en graf til navigation. Grafen anvendes senere af AStar-klassen, som finder den hurtigste rute mellem to Nodes, mens symboltabellen benyttes til tekstuel angivelse af vejnavne i rutennavigationen.

5.2 View

FirstWindow, som er et vindue der tilbyder brugeren valget mellem at bruge Krak eller OSM datasættet, er ansvarlig for at resten af programmet startes op med det valgte datasæt.

Window er vinduet som indeholder programmet efter at datasættet er indlæst. Til dennes sidebar er det eksterne *MigLayout* bibliotek anvendt.¹ Canvas er det primære komponent i Window, hvori kortet bliver tegnet.

Painter er ansvarlig for at tegne linjer og bokse og bliver anvendt af Tiler (se 5.3) og Canvas.

DropTextField tillader brugeren at vælge mellem en række forskellige adresseforslag gennem dennes rullemenu.

5.3 Controller

AddressButtonListener og AddressFieldListener tilsammen ansvarlige for at reagere på brugerinput i sidebaren i forbindelse med navigation. Når AddressFieldListener

¹<http://www.miglayout.com/>

modtager information om at brugeren interagerer med de to tekst-input-felter, kalder den AddressFinder for at hente forslag til adresser.

Når der er fundet en rute, anvendes klassen Path som datastruktur samt til at generere en tekstuelt repræsentation af rutevejledningen.

KeyboardHandler og MouseHandler er ansvarlige for at holde øje med brugerinput relateret til selve kortet. Disse to klasser står blandt andet for at håndtere når udsnittet skal zoomes, panoerereres, eller nulstilles.

Tiler er ansvarlig for at tegne kortet og holde styr på hvilket udsnit af kortet der vises.

5.4 Tests

Klasserne i denne pakke er tilsammen ansvarlige for at teste udvalgte klasser med JUnit tests.

5.5 Utility

Vector er en repræsentation af en vektor og bruges til vektorberegninger samt angivelser af punkter. Box er en repræsentation af et rektangel defineret af to Vectors. Line definerer et linjestrykke, som er klar til at blive tegnet.

5.6 Data

Ydermere findes to moduler til filtrering og klargøring af kortdata, som alle bevirket til en fælles formatering af dataet. KrakPurger muliggører formatering af Kraks kortdata, mens saxEventHandler, saxFilter og GeoConvert² understøtter saxPurger³, som formaterer OSM-datasættet.

²<https://github.com/Jotschi/geoconvert/blob/master/src/main/java/de/jotschi/geoconvert/GeoConvert.java>

³Længde- og breddegradsmetoder fra <http://www.geodatasource.com/developers/java>

Afprøvning

6.1 Afprøvning

I bestræbelse efter sikring af applikationens kvalitet og pålidelighed er en række afprøvninger af dennes bestanddele udført løbende med udviklingen. Afprøvningen er ikke foretaget med en målsætning om at udtømme samtlige muligheder for systemfejl, men derimod blot for at forhindre defekter, i de afprøvede klasser. Derudover fandt vi det ofte meget nyttigt at teste vores kode løbende under udviklingsprocessen, da dette ofte gjorde udviklingen af nye funktionaliteter mere effektiv.

Metode

Da afprøvningsmetoden, som anvendes under udarbejdelsen af tests, er afgørende for afprøvningens dækningsgrad, fastsattes den indledningsvist som *black-box testing* med supplerende *white-box testing* af udvalgte dele af kodebasen, hvor vi fandt det gavnligt.

Eftersom udviklingen er foretaget i Java, er afprøvning udført i prøvemiljøet JUnit — et populært Java værktøj til netop afprøvning. JUnit er således anvendt under kørsel af prøverne, hvorfor disse også er skrevet op imod JUnits API.

Som nævnt er formålet med afprøvningen ikke at gennemteste samtlige koden fra fragmenter i applikationen, hvorfor kun en række konkrete klasser og deres metoder er afprøvet. *Vector* og *Box* klasserne er brugt på tværs af applikationen, og netop derfor er disse udvalgt.

Ækvivalensklasser

Vector og *Box* klasserne er dataklasser, som indeholder værktøjsmetoder til hyppige operationer. Derfor håndterer samtlige metoder i bund og grund kommatal. Ækvivalensklasserne for afprøvning af klassernes metoder er således negative input, positive input samt nul-input.

Dokumentation

Dokumentation for de udførte prøver findes i bilag 11.9. Her ses de forventede og faktiske resultater af afprøvningsmetoderne opstillet i forventningstabeller.

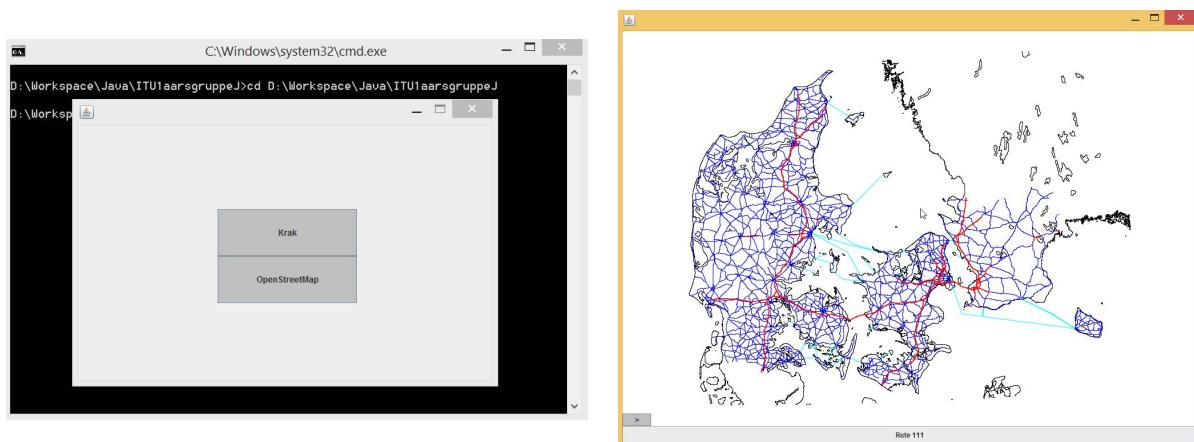
Brugervejledning

Programmet kan køres via den udleverede .jar-fil med følgende kommando “java -jar -Xmx2G map.jar”. Når programmet er startet op, skal man vælge hvilket data-sæt man vil benytte.

Kortmanipulation

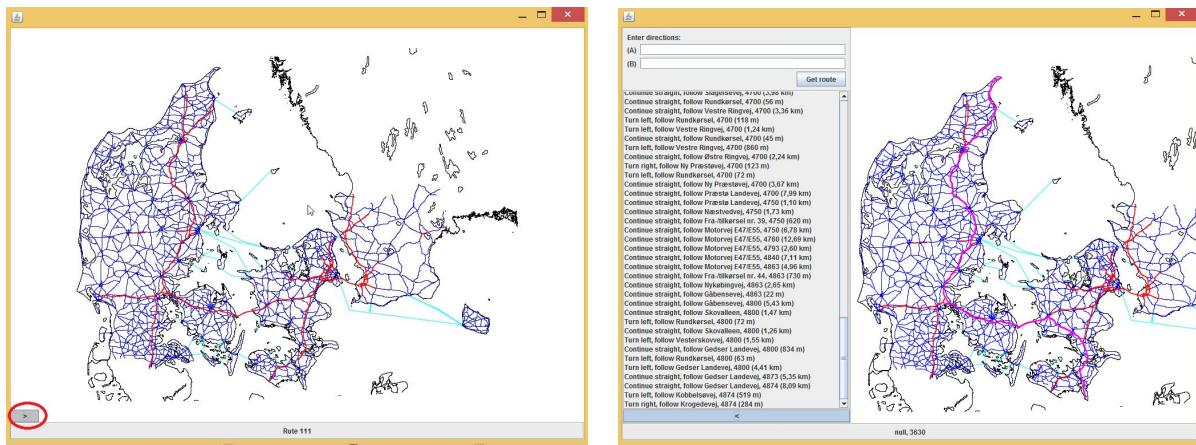
Tabel 7.1 – Tastaturgenveje

Kommando	Tast	Tast	Tast
Zoom kortet ind	i	+	num+
Zoom kortet ud	o	-	num-
Returner til startzoomgrad og center kortet	r		backspace
Bevæg kortet		piletasterne	
Vis/skjul rutevejledningsmenu		space	



Tabel 7.2 – Mussemanipulation

Kommando	bruger
Zoom kortet ind	Rul musehjulet væk fra bruger
Zoom kortet ud	Rul musehjulet hen mod bruger
Zoom kortet ind i markeret område	Højreklik, træk og slip så du har det ønskede udsnit
Returner kortet til startzoomgrad og centrer kortet	Tryk på den midterste museknap
Bevæg kortet	Venstreklik, træk og slip så du har det ønskede udsnit
åbn/luk rutevejledningsmenu	benyt knappen nede i venstre hjørne



Rutevejledning

Man kan enten benytte musen på kortet for at få en rutevejledning, eller man kan benytte sidebaren. Man benytter musen ved at højreklikke på sin start position, og så klikke med venstre musetast på sin slutdestination. I tilfælde af at man benytter sidebaren, skriver man udgangspunktets destination i felt (A) og destinationens adresse i felt (B), hvorefter man trykker på knappen: "Get route". Der vil nu vises en rutevejledning i sidebaren, såvel som at ruten vil være tegnet med en magenta farve på kortet.

Evaluering af programmet

Programmet understøtter alle de påkrævede funktionaliteter såvel som flere af de valgfrie. Vi er derfor grundlæggende tilfredse med programmets egenskaber. Vi kunne godt have tænkt os at have været mere kreative og fundet på nogle funktionaliteter selv, men vi havde ikke overskud til det.

8.1 Hastighed

Programmet er ~10 sekunder om at starte op med indlæsning af Krak data, og ~40 sekunder om at starte op med indlæsning af OSM data. Gruppen finder begge load tider acceptable, og vi har derfor ikke brugt særligt mange ressourcer på at forbedre dem.

Der er blevet arbejdet meget med hastighed i forbindelse med navigation af kortet (zoom og panorering), og vi mener grundlæggende, at programmets responsstider er rigtig gode på dette punkt. Det samme gælder udregning og visning af rutevejledninger.

Programmet kører generelt lidt langsommere, når OSM-datasættet er indlæst. Dette skyldes, at mængden af data er meget større, og at beregningerne på disse data som konsekvens tager længere tid.

8.2 Fejl og mangler

Vores filtrerede OSM datasæt indeholder ingen information, om hvilken by et vejstykke er associeret med. Det er hermed umuligt at skelne mellem veje, der har det samme navn i forskellige byer. Kun ~30% af vejene i vores OSM-datasæt indeholder information om hastighedsbegrensinger, fordi vi ikke har formået at udnytte datakilden optimalt. Som en konsekvens af dette er vores rute-beregning upræcis, da det for lang hovedparten af vejstykkerne ikke er de faktiske hastighedsbegrensninger der anvendes, men derimod en statisk standardværdi efter vejtype. Hvis det er motorvej er den 130 km/t, hovedvej 80 km/t og alle andre tilfælde 50 km/t.

Vi anser det som en mangel, at programmet ikke tager højde for forbudte højre og venstre sving, samt ensrettede og lukkede veje, selvom det ikke er et decideret krav. Uden denne funktionalitet er der ingen garanti for at de ruter som programmet beregner, er mulige og lovlige at følge i praksis.

8.3 Brugervenlighed

Vores program giver ikke brugeren nogen informationer om hvad mulighederne for interaktion er, og det er derfor op til brugeren selv at udforske programmets funktionaliteter. Dette kunne gå galt, hvis brugeren ikke er bekendt med digitale kort generelt, og derfor ikke har en forhåndsviden om typiske muligheder for interaktion i denne type program.

Vi havde ambitioner om at lave brugertests af vores program og forbedre brugervenligheden af vores program baseret på resultaterne af disse. Desværre måtte vi indse, at vi ikke havde overblik og tid til at få planlagt og foretaget dem.

Evaluering og refleksion over proces

9.1 Evaluering og reflektion over proces

I dette afsnit reflekteres og evalueres der over hele processen fra problemidentifikation til det endelige produkt. I bilag 11.4 ses en udførlig angivelse af hvilke dele af arbejdet, som de individuelle gruppemedlemmer har haft ansvaret for.

Projekt del I

I den indledende fase udarbejdede gruppen i fællesskab en samarbejdsaftale, hvori der blev sat prioriteringer, ambitionsniveau, præferencer m.m. Gruppen besluttede, at der efter hvert møde skulle føres log og udfyldes arbejdsblade, samt planlægge arbejdsopgaverne for næste gang. Samarbejdsaftalen var med til at afklare hvilke forventninger, som vi hver især havde til samarbejdet, og var med til at sætte fælles mål for projektet.

Der kunne nu tages hul på den næste fase, hvor problemet identificeres og defineres. Her foretog gruppen en grundig gennemgang af de obligatoriske krav samt hvilke valgfrie krav, der kunne tilføjes. Dernæst afgrænsede gruppen sig i første omgang til at tilføje en zoom-ud funktion udover de obligatoriske krav. Det var på daværende tidspunkt svært at tage stilling til, hvilke valgfrie funktioner vi ville tilføje. Det vigtigste for os var at kunne få lavet et første udkast af visualisering af kortet. På baggrund af dette var gruppen afklaret med at benytte Krak's data som datakilde, da den var nemmere at benytte og håndtere.

Det næste trin i forløbet var at opstille en løsningsmodel, hvor gruppen tog udgangspunkt i at strukturere koden i et Model-view-controller designmønster. Vi foretog en tavlediskussionen, der omhandlede hvilke klasser der skulle være i hvert modul. Dette var med til at give gruppen et godt overblik over, hvad programmet skulle indeholde, hvilke arbejdsopgaver der skulle udføres, og hvilke centrale klasser der skulle laves.

Efter at have opstillet en løsningsmodel, kunne vi arbejde hen imod at få lavet et første udkast af programmet. Det var ikke alle der fik skrevet kode i første omgang. Det var primært dem, som havde styr på at kode, der tog styringen. Dette kunne ikke undgås, da der i starten var få og store arbejdsopgaver, og da det var vigtigt for gruppen at opnå resultater på kortest tid. Grundet de få arbejdsopgaver opstod der tilfælde, hvor kun én person skrev kode, mens de andre fulgte med og kom med input. Dette medførte, at produktiviteten var relativ lav. Det var også denne del, der var tungest og tog længst tid.

Produktiviteten begyndte først at blive relativt høj efter, at vi fik tegnet kortet for første gang. Der blev arbejdet på forskellige moduler på én gang, og Model-view-controller designmønstret gav stor glæde og viste sig her at være særlig nyttigt. Der blev i gruppen arbejdet mere effektivt, og der blev tilføjet flere valgfrie funktioner. Vi fik i gruppen mere overskud og større overblik og kunne lave væsentlige ændringer med henblik på at nå til et egentligt produkt og færdiggøre første del af projektet.

Projekt del II

Kort efter aflevering af første del af projektet opstod der en mindre krise i gruppen. Et gruppemedlem havde haft nogle personlige problemer, der var opstået under projektforløbet. De andre gruppemedlemmer kendte ikke til gruppemedlemmets situation. De kunne fornemme at ambitionsniveauet hos ham ikke var i overensstemmelse med det ambitionsniveau, som blev sat under samarbejdsaftalen. De andre gruppemedlemmer konfronterede gruppemedlemmet. Gruppen fik renset luften og talt ud, og gruppemedlemmet forklarede kort, hvad han havde gået igennem, og samtidig gjorde han det klart for gruppen, at problemerne var blevet løst. Gruppen viste stor forståelse for gruppemedlemmets situation. Gruppen var igen på lige fod og var bedre klædt på til anden del af projektet.

Vi tog hul på anden del af projektet ved at gentage problemidentifikationsfasen. Der skulle tages højde for de nye obligatoriske krav og muligheder for udvidelser, som vores produkt skulle opfylde. I første omgang var målet blot at opfylde alle obligatoriske krav samt implementere én udvidelse. Det var endnu uvist, hvilke udvidelser vi ville implementere, men vi fik dog udelukket nogle udvidelser, som vi ikke fandt interessante.

Der blev i gruppen udarbejdet en arbejdsliste over de diverse arbejdsopgaver som skulle udføres. På daværende tidspunkt virkede det som en stor mundfuld med alt den nye funktionalitet som skulle tilføjes til programmet. Størstedelen af alle arbejdsopgaver kunne løses uafhængigt af hinanden, dette gjorde det muligt at alle i gruppen kunne sidde med arbejdsopgaver, hvilket var med til at øge produktiviteten og gøre arbejdslisten mere overskuelig.

Arbejdsopgaverne blev jævnt fordelt, det kunne ikke undgås, at nogen fik lidt større og mere omfattende arbejdsopgaver end andre. Der var dog stor valgfrihed under fordeling af arbejdsopgaver. Efter fordelingen kunne vi påbegynde implementeringen. Den overordnede arbejdsproces bestod nu af flere arbejdsforløb, som hver gruppemedlem var ansvarlig for. Der blev sat en endelig deadline, for hvornår alle arbejdsopgaver skulle være færdige. Der var nu op til den enkelte gruppemedlem at overholde sin deadline.

Denne arbejdsmåde var særdeles produktiv, og havde sine fordele, men det havde også sine ulemper. Den store fordel var, at når en gruppemedlem var færdig med sine arbejdsopgaver, kunne han hjælpe til med andre arbejdsopgaver eller fixe bugs og løse nyopståede problematikker. En af de store ulemper var, at når man stødte på problemer, var det nogle gange svært for de andre gruppemedlemmer at hjælpe, da de ikke var lige så godt sat ind i arbejdsopgaven som den ansvarlige. Dette medførte at der var perioder, hvor de enkelte arbejdsforløb var gået lidt istå, hvilket resulterede i at arbejdet blev forsinkel, og vi fik svært ved at nå vores deadline.

Da vi nåede den fastsatte deadline, havde vi færdiggjort stort set alle arbejdsopgave, men måtte dog erkende at vi ikke helt var der hvor vi gerne ville have været. Dette resulterede i, at vi måtte bruge mere tid på at få færdiggjort de sidste arbejdsopgaver, og påbegyndelsen af rapportskrivning blev udskudt.

Opsummering

Efter den første del af projektet talte vi om at lægge en bedre strategi for at undgå den lave produktivitet. Derudover talte vi om, at vi i gruppen ikke haft den store erfaring med at lave større programmer, hvilket er en af grundene til, vi stødte ind i denne problematik.

I anden del af projektet har vi formået at lægge en bedre strategi. Strategien har bygget på bred fordeling af arbejdsopgaver, der var med til at øge produktiviteten. Vi har denne gang haft større overblik og har kunne planlægge projektet bedre. Vi har i gruppen været gode til at forstå, hvor hinandens styrker og svagheder ligger, og vi har suppleret hinanden godt.

Konklusion

Vi kan konstatere, at vi har opfyldt samtlige obligatoriske krav stillet i kravspecifikationen. Derudover har vi opfyldt 5 ud af de 8 valgfrie opgaver. Vi finder vores implementation af opgaven stabil, og er generelt tilfredse med dens funktionalitet.

Der er dog nogle fejl og mangler, som vi gerne ville have rettet i implementationen, hvis vi havde haft mere tid. f.eks forbudte sving, såvel som OSM-datasættets rutevejlednings-begrænsninger. Derudover ville vi gerne have implementeret et mere brugervenligt design på baggrund af usability tests.

I tilfælde af at vi skulle arbejde videre med projektet havde vi rettet disse fejl, såvel som udvidet og forbedret funktionaliteten. Eksempler på dette kunne f.eks. være en databaseserver til tiles, lukkede polygoner og landskabsfarvning og print af rutevejledning.

11

KAPITEL

Bilag

11.1 Kravsspecifikation

BFST F2014 Examination Project Description

BSWU, IT University of Copenhagen
Søren Debois & Troels Bjerre

April 3, 2014

1 Introduction

This is the text of the ITU BFST-F2014 course's examination project description. Based on this description, you must produce (a) source code and (b) a project report and submit both electronically via learnit no later than 2pm, May 21, 2014. Both source code and project report must be produced by groups as indicated on the course homepage. It is not possible to submit individually.

In this project, you must design and implement a system for visualising and working with map data.

2 Requirements for the final product

The final system should fulfill the following requirements, which you presumably already designed and implemented.

- Draw all roads in the map dataset;
- use different colors to indicate types of road;
- adjust the drawing when the size of the window changes;
- focus on a specific area on the map, either by drawing a rectangle and zooming in it, or by zooming in the center of the map and pan the map afterwards
- unobtrusively show either continuously or on hover the name of the road closest to the mouse pointer, e.g., in a status bar.

Moreover, your product must also:

1. be able to compute a shortest path on the map between two points somehow specified by the user (e.g. by typing addresses or clicking on the map).
2. feature a coherent user interface for your map and accompanying functionality. Specifically, you must decide how the mouse and keyboard is used to interact in a user-friendly and consistent manner with the user interface;

3. be fast enough that it is convenient to use, even when working with a complete data set for all of Denmark. The start-up time can be hard to improve as long as the data is stored in text files, but after the start-up the user interface should react reasonably quickly.
4. allow the user to choose between using either the KRAK dataset and Open Street Map's map of Denmark as found on, e.g., <http://download.geofabrik.de/europe/denmark.html>. The above requirement that the program is fast enough applies also to OpenStreetMap data.
5. contain at least one extension feature, either from the list in Section 3 below, or one you invent yourselves. In the latter case, contact a lecturer to check that your feature is substantial enough.
6. for groups of 4, your solution must also be able to output a textual description of the route found in Item 1 giving appropriate turn instructions. E.g., "Follow Rued Langgaardsvej for 50m, then turn right onto Røde Mellemvej."

Some of what you have programmed previously can be reused without change in the final product, but there are probably also parts that require rethinking and reworking. Likewise, a lot of what you have written in your work sheets and hand-ins can probably be used in the final report, but in a modified and rewritten form.

3 Extensions

As mentioned, you must implement at least one of these. Extra credit for doing more than one.

- (a) Make it possible to compute a shortest path tree from a given starting point p , and highlight the path in the tree from p to a point on the map defined by *double-clicking* on it. (This should be essentially instantaneous once the tree is computed.) (b) You must also continuously update the displayed shortest path to wherever you drag the mouse.
- Improve the looks of the outermost zoom levels. For example, compute an approximation of the coastline so you can shade water and land in different colours. Coastline data can be obtained from: http://www.ngdc.noaa.gov/mgg_coastline/ or e.g. from: <http://download.geofabrik.de/europe/denmark.html>. Observe that both datasets are in Lat/Lon format.
- Use the Twitter API (<https://dev.twitter.com/docs/api/1/get/search>) to retrieve and display geotagged tweets on the map.
- Implement approximate matching for address searches, so "Ruud Langrds vej" is matched to "Rued Langgaards Vej". While reasonably easy to do for small data sets, it is an open research problem how to do this efficiently in general.
- Implement a "search-as-you-type" feature when searching for a road-name. This would provide the user with a drop-down list of road names that match the characters of the road name he has typed so far.
- Improve the user interface so that the user can choose between routes for biking/walking and for cars. Car routes should take into account speed limits/expected average speeds. A route for biking/walking should be the shortest and cannot use highways.

- Make your route planner obey restrictions on turns (e.g. illegal left turn); this requires understanding (and parsing) of the file `turn.txt`, which is included in the Krak data set. (Hint: Construct a new input graph where an intersection with k adjacent road segments is represented by k nodes.)
- Explore methods in the literature to make the route planning more efficient. (Such methods are used by Google Maps, and similar services, to quickly serve requests.) You can find relevant information in:
http://link.springer.com/content/pdf/10.1007%2F978-3-642-02094-0_7

You are encouraged to find additional or alternative extensions you find (more) interesting. Clear your custom extension with a lecturer, though.

As mentioned in Section 1, an important part of the final part of the project work consists of the group discussing which functionality you will implement and in which order and that the report describes your analysis (including arguments for your choices) and the extent to which you have succeeded in implementing the choices you have made. In other words, you should make it clear that you are working towards some goal (even if you don't make it) instead of randomly working on various possibilities.

4 Rules

- You are free to use third party libraries (apart from map APIs) as long as you cite your sources and reflect on your choice. Ask a lecturer if in doubt.
- Extensions to the basic system may be programmed by individual group members alone, or in subgroups. Together with the oral exam, this can give a basis for individual grading. However, it is the responsibility of the whole group that the final product is a well-integrated whole, and that the software for the basic functionality is of good quality.
- The final submission must be in the form of a zip-file containing a PDF project report, source code, a README file indicating how to build and run your program. Contact a lecturer before submission if your application is not written in Java.
- The project report should describe both process and product (including parts that you previously handed in). It must be no longer than 40 pages, excluding appendices. The report may contain parts describing an individual's contribution, but these parts should not exceed 4 pages per group member.

4.1 Evaluation of project work

An important part of the evaluation will concern the report's **description of product and process**, cf. the course's intended learning outcomes. Thus it is important that you document your work in diary and work sheets, that you work in a structured manner, analyze the problems and make decisions, make plans and try to follow them, and document experiments.

4.2 Form of the report

The project report should contain at least the following.

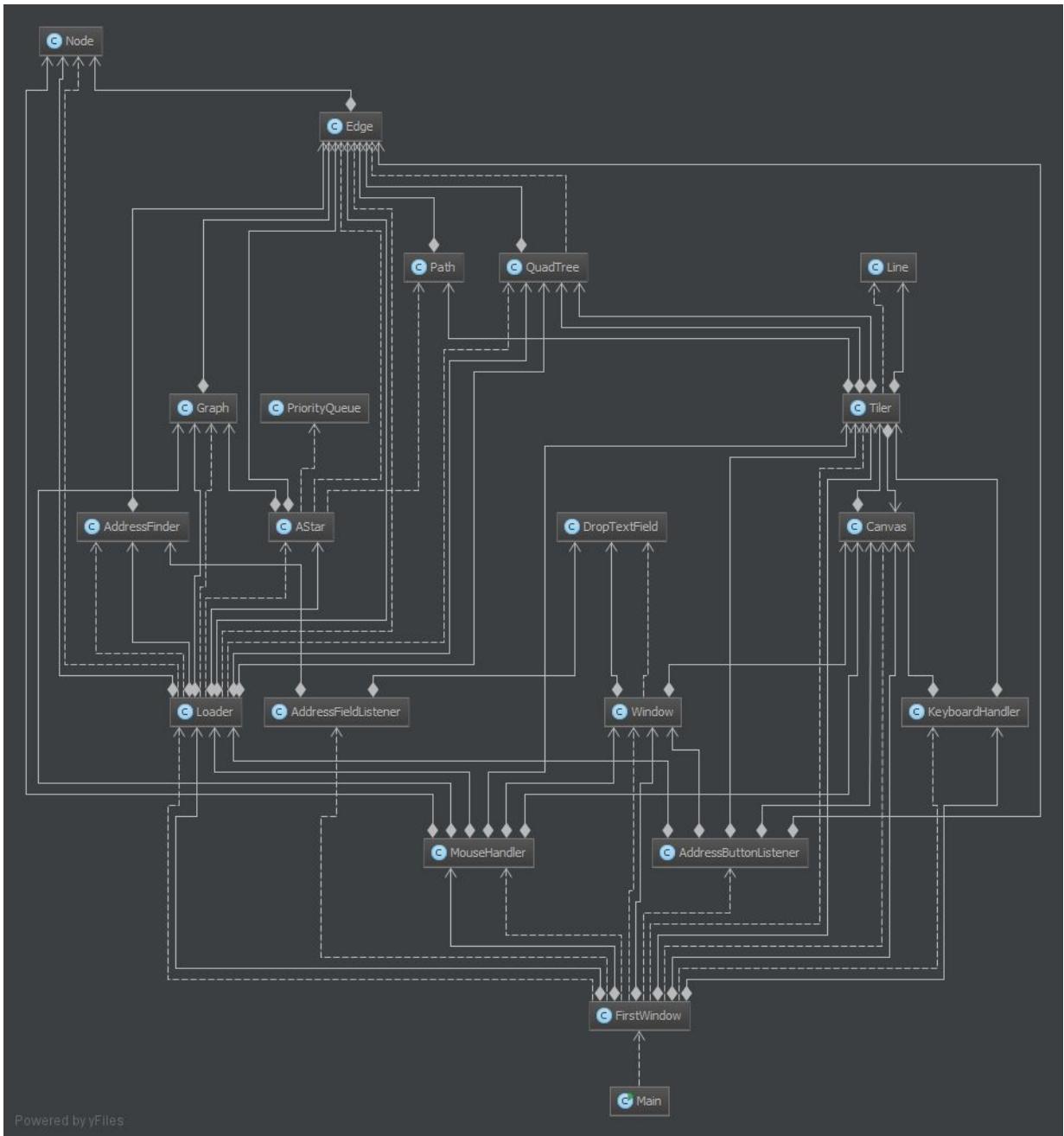
- Front page with project title, group number, names and emails of group members, hand-in date, name of course (“First-Year Project, Bachelor in Software Development, IT Univ. of Copenhagen”).
- Preface (where, when, why).
- Background and problem area, data set, your requirements for the product.
- Analysis, arguments, and decisions regarding the design of the user interface.
- Analysis, arguments, and decisions on how to implement the user interface, including choices of algorithms and data structures. Also include descriptions of any experiments you have made to decide on choices on algorithms and data structures.
- Short technical description of the structure of the program using simple UML diagrams.
- Systematic test of the resulting system. The test can be concerned with unit tests of classes that you have implemented. Moreover, you should also include a white-box test of a part of your program that has non-trivial control flow (e.g., a method with nested `if` statements and `while`/`for` loops).
- User manual for the system (brief, e.g., using screen shots complemented by a description of functionality).
- Product conclusion: the extent to which the product fulfills your requirements, including a list of what is missing, and a brief description of which new ideas to functionality, design, and implementation that you have but have not explored.
- Process description with reflection on the process (including a description of how you could have improved the process). This must include an overview of who in the group (if not all members) were main contributors of different parts of the product.

The appendices should contain:

- Group norms (constitution).
- Diary / log book.
- Work sheets for parts of your project.
- Optionally, selected source code for central parts that you refer to in the technical description.

5 Advice

- Remember that you have already worked for quite a long time on this project; hence it is easy to forget how little outsiders (such as the external examiner) understand about the data set, the graph representation, the problems regarding visualization, etc. Therefore, you should remember to explain these items in the report instead of just talking about “UTM32”, “KrakNode”, etc. Your explanation should be brief and to the point; try to use drawings and diagrams where appropriate.
- It can be a challenge to fit the report within the maximum number of pages; you should not do so by changing margins, using small fonts, etc., but rather by repeatedly cutting away the least relevant parts. If in doubt, ask a lecturer.
- Remember to structure the text using sections, subsections, paragraphs, tables, figures, list of references, etc.
- When you discuss a problem then remember to include pros and cons for different possible solutions, and an argument for your choice.
- Make sure that you have implemented the core requirements before implementing extensions!
- Use a step-wise refinement procedure for your product. First analyze, design and implement a simple solution, then move to a more complicated solution.
- Remember to use what you have learned in BADS when arguing for choice of algorithms and data structures! Also remember that empirical observations about the data set can be used to argue for or against different possible solutions. Likewise, timing experiments can also be used to argue for one solution possibility over another.
- Make sure that everyone in the group has ownership of the program, i.e., understands what is going on and is confident to make changes. One way to ensure this is to explain parts of the program to each other (this is also a good way to test if the code is understandable).
- Have fun!



11.2 Krakdokumentation

Introduction to the KRAK roadmap data and associated code

Introduction to the KRAK roadmap data and associated code

Access to the Krak data

Please note that we have been given access to the Krak roadmap data strictly for teaching and research. The data must not be redistributed to anyone not participating in this course, and should be deleted after the course ends.

Structure of the Krak roadmap data

The most interesting part of the roadmap data are found in the Txt subfolder in the files ‘kdv_node_unload.txt’(Node file from here on) and ‘kdv_unload.txt’(edge file from here on) which contains the roadmap data exported to a simple text format.

The node file contains a list of nodes representing intersections in the road network. Each node is associated the following data:

Its geographic position is listed as X-COORD and Y-COORD. It is also associated to identifiers KDV# and KDV-ID. We will use KDV-ID as a global identifier, that is it will remain the same if we use a subset of the nodes, while we will keep KDV# continuous from 1 to the number of nodes in the file. The entry ‘ARC#’ will not be used.

The Edge file contains a set of edges corresponding to road segments between pairs of intersections.

The FNODE# and TNODE# gives the KDV# of the endpoints of the segment. DAV_DK# is used as a continuous identifier similar to KDV# while DAV_DK-ID is used as a global edge identifier remaining the same even if we only use a subset of the edges. Another field that is very important is LENGTH which gives the length of the road segment.

The table below gives interpretations of the remaining fields.

Field name	Description
TYP	Road type indicator (details only available in Danish)
VEJNAVN	Road name
FROMLEFT/RIGHT	The beginning of the house numbers on the left/right
TOLEFT/RIGHT	The end of the house numbers on the left/right
FROMLEFT/RIGHT_BOGSTAV	Same for letters associated to houses
TOLEFT/RIGHT_BOGSTAV	Same for letters associated to numbers
V_SOGENR	Parish number on the left side of the road segment
H_SOGENR	Parish number on the right side of the road segment
V_POSTNR	Left zip code
H_POSTNR	Right zip code
KOMMUNENR	Not relevant

VEJKODE	Road ID specific for commune (Not relevant)
SUBNET	Not relevant for this project
RUTENR	Not relevant for this project
FRAKOERSEL	Highway turn-off
ZONE	Type of area: 10 – City 20 – Holiday housing 30 – Industrial
SPEED	‘Ideal speed’ (treat as speed limit)
DRIVETIME	Estimated driving time
ONE_WAY	tf = one way in the direction To-From ft = one way in the direction From-To n = No driving allowed <blank> = No restrictions
F_TURN	Related to turn restrictions, if you wish to use such information, talk to a supervisor or examine the Danish Krak documentation and /Txt/turn.txt
T_TURN	See above
VEJNR	KRAK internal road ID (Not relevant for this project)
AENDR_DATO	Change date (Not relevant for this project)
TJEK_ID	Road segment ID (Not relevant for this project)

The code framework

The code framework contains detailed comments, which you should read. The following is just a bird-eye view of the code explaining the overall layout.

Basic Graph Library

The basic graph library is contained in the Java package graphlib, and consists of Edge.java, Node.java and Graph.java. It is a simple implementation of an adjacency list representation of a graph.

KrakLoader

The KrakLoader can read and parse the text files supplied by krak and recreate the corresponding graph using the basic graph library.

InstanceCreator

The instance creator can be used to systematically create a small test road network. It loads the krak text files and allows custom filters to include or exclude nodes and edges. The result can be written to new text files. Please read the comments for the instance creator in detail to be able to trace road segments between the original and the new road network.



Kraks Danske Vejnet

Professionel - Rute

Dokumentation

Kraks Forlag AS
Kraks Kort & GIS
Virumgårdsvej 21
2830 Søborg

Telefon: 45 95 65 00
Fax: 45 95 65 65
E-post: geoinfo@krak.dk



Indholdsfortegnelse

1. Installationsvejledning	1
2. Adresse- og Vejnetsdatabase.....	2
3. Danmarkskort	7



1. Installationsvejledning

Nye brugere af Kraks Danske Vejnet skal kopiere data fra cd-rom til harddisken. Mappen **Krak** kopieres fra cd-rom til c:

Brugere som modtager opdatering, skal slette det gamle Kraks Danske Vejnet, hvorefter den nye kopieres over på harddisken. **HUSK** at flytte alle egne projekter og data, der ligger i mappen **krak**, før du sletter.

Når vigtige filer er blevet gemt andetsteds så kopieres mappen **krakkdvpro** fra cd-rom til **c:\krak** på din harddisk. (Man kan godt arbejde med dataene via cd-rom drevet, men computeren arbejder hurtigere, når de er gemt på harddisken).

Derefter kan projektet åbnes og Kraks Danske Vejnet er klar til brug.

Adresse-geokodning

Før adresse-geokodning kan foretages skal man:

1. Kopiere alle filer (undtagen readme filerne) fra mappen "dansk_adrstil" under mappen "kdv" til mappen "Esri/av_gis30/arcview/geocode".
2. Derefter oprettes et geokodning indeks ved at åbne ArcView, åbne et project og vælg kdv temaet. Med kdv temaet valgt, gå du i properties i menulinien Theme og vælger geokodning. Her vælges 'Danske veje med postnr' som Address Style. Opdateringen tager ca. 3-5 minutter og derefter er du klar til at geocode dine adresser. **Husk at gemme projektet og lukke ArcView, så dit nyoprettede geokodningsindeks gemmes.**

Bemærk: Man skal slette alle skivebeskyttelses formater på de filer man vil arbejde med. Dvs. at man skal fjerne skrivebeskyttelsen fra alle filer i vejdatabase ved at fremhæve dem (klikke på dem), vælg egenskaber efter højre museklik eller fra menuen filer, og så fjerne fluebenet i feltet skrivebeskyttet. Dette kan klares nemt fra stifinderen.



2. Adresse- og Vejnetsdatabase

Mappen **KDV** indeholder hele det danske vejnet inkl. adresser med husnumerintervaller på de enkelte vejstykke.

Ophavsrettigheder: Kraks Forlag AS har rettigheder til videreføring, distribution og salg af databasen. DAV (Dansk Adresse- og Vejdatabase) har ophavsret til den grundlæggende database.

Dækningsområde: Hele Danmark.

Nøjagtighed: Vejsegmenterne har en geometrisk nøjagtighed på ca. 5m udtrykt som punktmiddelfejl.

Definition: Mellem to vejkryds vil der være mindst et vejsegment. En ændring af en eller flere attributværdier (f.eks. postnummerændring mellem to kryds) vil bevirket, at der vil være mere end et vejsegment mellem de to kryds. Dvs. at vejen af tekniske grunde kan blive delt op i mindre segmenter uden at det er umiddelbart synligt på kortet. Vejsegmenterne udgør et samlet netværk, som gør det muligt at foretage afstandsberegning, mm.

Koordinatsystem: System 34 - Sjælland/Jylland, System 45 - Bornholm - eller aftalt utm-projektion

Kortenheder: Meter

Format: ArcView GIS Shapefil format

Bemærk: Som følge af en uens politik omkring husnummer tildeling i de enkelte kommuner, følger husnumrene ikke altid datamodellen. Husnumrene kan også være ulige på begge sider af vejen, stige på den ene side og falde på den anden side etc.

I mappen **KDV** findes det danske vejnet som følgende ArcView-shapefiler:

Kdv.shp – Vejnet for hele Danmark	
Tabelattributter:	Forklaring:
Shape	Line (linie)
Length	Længde på vejsegment
Vejnavn	Vejnavn og færgerutenavn
Typ - indeholder alle <i>type</i> definitioner der ligger i databasen. Typ identer med kursiv skrift er ikke taget i brug fra 1.1.02.	1 – Motorveje 2 – Motortrafikvej 3 – Primærrute > 6 meter 4 – Sekundærrute > 6 meter 5 – Vej 3 - 6 meter 6 – Anden vej 8 – sti 10 – markvej 11 – gågader 21 – proj. motorvej 22 – proj. motortrafikvej 23 – proj. primære vej 24 – proj. sekundære vej 25 – Proj. vej 3-6 m 26 – Proj. vej < 3 m 28 – Proj. sti 31 – Motorvejsafkørsel 32 – Motortrafikvejsafkørsel 33 – Primærevejsafkørsel 34 – Sekundærevejsafkørsel 35 – Anden vejafkørsel 41 – Motorvejstunnel 42 – Motortrafikvejstunnel <i>43 – Primærevejstunnel</i> <i>44 – Sekundærevejstunnel</i> <i>45 – Anden vejtunnel</i> <i>46 – Mindre vejtunnel</i> <i>48 – Stitunnel</i>

Kdv.shp – Vejnet for hele Danmark	
Tabelattributter:	Forklaring:
	80 – Færgeforbindelse
	99 – Stednavn eksakt beliggenhed ukendt (f.eks. et torv hvor der ingen <u>vejadresse</u> fandtes i det originale dataset.)
Fromleft	Husnummer start på segmentets venstre side
Toleft	Husnummer slut på segmentets. venstre side
Fromright	Husnummer start på segmentets højre side
Toright	Husnummer slut på segmentets højre side
Fromleft_b	Laveste husbogstav på segmentets venstre side
Toleft_bog	Højeste husbogstav på segmentets venstre side
Fromright_	Laveste husbogstav på segmentets højre side
Toright_bo	Højeste husbogstav på segmentets højre side
V_sognenr	Sognekode på segmentets venstre side
H_sognenr	Sognekode på segmentets højre side
V_Postnr	Postnummer på segmentets venstre side
H_Postnr	Postnummer på segmentets højre side
Kommunenr	Unikt kommunenummer
Vejkode	Unik vejkode indenfor hver kommune
Rutenr	Unikt rutenummer
Frakoersel	Motorvejsfrakørsel
Zone	10 - By 20 - Sommerhus 30 - Industri Værdierne er vejledende, idet der forestår en opdatering før de er synkroniseret med de respektive områder i "Danmarkskort"-mappen.
Speed	Idealhastighed på vejsegment
Drivetime	Kørselstid pr. segment ved idealhastighed + 15%
One_way	tf = ensrettet modsat digitaliseringsretning (To-From) ft = ensrettet i digitaliseringsretning (From-To) n = ingen kørsel tilladt (henholdsvis typ 8, 21-28) <blank> = ingen ensretning
Aendr_dato	Dato for eventuelle ændringer i grunddata.
Vejnr	Kraks interne vejnummber som kan bruges med andre Kraks produkter.
Tjek_id	Unikt identifikationsnummer for alle vejsegmenter

Kdv.shp – Vejnet for hele Danmark	
<i>Tabelattributter:</i>	<i>Forklaring:</i>

I mappen **KDV** forefindes foruden vejnettet også en fil indeholdende informationer om svingrestriktioner. Filen er en negativ liste indeholdende information om hvilke vejsegmenter der IKKE må køres fra/til (F_tjek_id/T_tjek_id). Desuden indeholder turn.shp information om retningen af F- og T-segmentets retning i forhold til den retning der køres mellem de to segmenter - denne information er nødvendig for at tage hensyn til beskrivelsen af u-sving.

Ved et forbud mod u-sving er F- og T-tjek_id ens - derfor anvendes F_edge_dir hhv. T_edge_dir til at beskrive i hvilken ende af segmentet forbudet gælder. Således vil F_edge_dir og T_edge_dir indeholde FT hhv. TF såfremt restriktionen gælder i T-enden af segmentet.

turn.shp – Svingrestriktioner	
<i>Tabelattributter:</i>	<i>Forklaring:</i>
Shape	Line (linie)
F_tjek_id	Unik ident (tjek_id) på fra vejsegmentet i KDV.shp
T_tjek_id	Unik ident (tjek_id) på til vejsegmentet i KDV.shp
F_edge_dir	tf/ft - retning af fra vejsegmentet (nødv. pga. u-sving)
T_edge_dir	tf/ft - retning af til vejsegmentet (nødv. pga. u-swing)

I mappen **Vejklasser** er vejene delt op i fire klasser som hver har sit eget tema. Derved kan man nemt lave kort med kun de større veje når små veje ikke er interessante.

Motorvej.shp – Motorveje	
<i>Tabelattributter:</i>	<i>Forklaring:</i>
Shape	Line (linie)
Typ	Udtræk af KDV hvor typ = 1, 31 og 41
Vejnavn	Tekstbeskrivelse
Tjek_id	Unik ident

Motortrf_primvej.shp – Primærveje	
<i>Tabelattributter:</i>	<i>Forklaring:</i>
Shape	Line (linie)
Typ	Udtræk af KDV typ = 2, 3, 32, 33, 42 og 43
Vejnavn	Tekstbeskrivelse
Tjek_id	Unik ident

Sekund_vej.shp – Sekundærveje

<i>Tabelattributter:</i>	<i>Forklaring:</i>
Shape	Line (linie)
Typ	Udtræk af KDV hvor typ = 4, 34 og 44
Vejnavn	Tekstbeskrivelse
Tjek_id	Unik ident

Andre_veje.shp – Andre veje

<i>Tabelattributter:</i>	<i>Forklaring:</i>
Shape	Line (linie)
Typ	Udtræk af KDV hvor typ = 5,6,11,10,13,35,45 og 46
Vejnavn	Tekstbeskrivelse
Tjek_id	Unik ident

Bemærk: Kraks Forlag garanterer ikke at *alle* attributfelter er udfyldt. Dette gælder f.eks. ensretninger og rutenumre.

3. Danmarkskort

Mappen **Danmarkskort** på cd'en i Kraks Geoinfo Pakke indeholder kortdata, som gør det muligt at fremstille kartografiske kort over hele Danmark.

Ophavsrettigheder: Kraks Forlag AS
Format: ArcView GIS Shape-filformat

I mappen findes følgende af Danmarkskortets elementer som ArcView-shapefiler:

Danmark.shp - Danmarkskort	
Tabelattributter:	Forklaring:
Shape	Polygon (flade)
Area	Områdeareal (m ²)
Perimeter	Områdeomkreds (m)
Type	6 – Land 7 – Vand 10 - Tyskland

By.shp – byer/bebyggede områder	
Tabelattributter:	Forklaring:
Shape	Polygon (flade)
Area	Områdeareal
Perimeter	Områdeomkreds
Navn	Områdets eventuelle navn

Industri.shp – industriområder	
Tabelattributter:	Forklaring:
Shape	Polygon (flade)
Area	Områdeareal
Perimeter	Områdeomkreds
Navn	Områdets eventuelle navn

Sommerhus.shp – sommerhusområder	
Tabelattributter:	Forklaring:
Shape	Polygon (flade)
Area	Områdeareal
Perimeter	Områdeomkreds
Navn	Områdets eventuelle navn

Soe.shp – natur: sører	
Tabelattributter:	<i>Forklaring:</i>
Shape	Polygon (flade)
Area	Områdeareal (m2)
Perimeter	Områdeomkreds (m)
Navn	Områdets eventuelle navn

Skov.shp – natur: skovområder	
Tabelattributter:	<i>Forklaring:</i>
Shape	Polygon (flade)
Area	Områdeareal (m2)
Perimeter	Områdeomkreds (m)
Navn	Områdets eventuelle navn

Lysning.shp – natur: lysninger	
Tabelattributter:	<i>Forklaring:</i>
Shape	Polygon (flade)
Area	Områdeareal (m2)
Perimeter	Områdeomkreds (m)
Navn	Områdets eventuelle navn

Jernbane.shp – jernbane	
Tabelattributter:	<i>Forklaring:</i>
Shape	Line (linie)
Type	Tomt felt, ingen definitioner

Vandloeb.shp – Vandløb	
Tabelattributter:	<i>Forklaring:</i>
Shape	Line (linie)
Typ	16 - Vandløb

Transport.shp – stationer, færgehavne og lufthavne	
Tabelattributter:	<i>Forklaring:</i>
Shape	Point (punkt)
Transpnavn	Tekstbeskrivelse
Type	3 – S-tog stationer 4 – Stationer, generelt 10 – Trinbræt 15 – Lufthavne 16 – Flyvepladser 17 – Færgehavne

Geested.shp – stednavne	
Tabelattributter:	<i>Forklaring:</i>
Shape	Point (punkt)
Stednavn	Stednavne
Type	2 – Stednavn (henholdsvis byer) 20 – Camping 21 – Havn 22 – Lystbådehavn 23 – Golfklub 24 – Sygehus 25 – Kirker og sogn 26 – Forlystelsespark 27 – Museer og seværdighed 28 – Seværdighed 29 – Haver og park 30 – Zoologiske have 31 – Vandrerhjem 32 – Hal

11.3 OSMdokumentation

http://wiki.openstreetmap.org/wiki/Main_Page

11.4 Arbejdsfordeling

Implementering

- **Ans:** Box, Vector, Window, FirstWindow, Groups, QuadTree.
- **Bjørn:** Box, Line, Vector, KeyboardHandler, MouseHandler, Path, Tiler, KrakPurger, Edge, Groups, Loader, Node, TestBox, TestVector, Canvas, Painter.
- **Malthe:** Box, Vector, AddressButtonListener, AddressFieldListener, AddressFinder, MouseHandler, AStar, Bag, Djikstra, Edge, Graph, Loader, Node, PriorityQueue, QuadTree, TestAStar, TestBox, TestPriorityQueue, TestVector, DropTextField.
- **Mark:** KeyboardHandler, GeoConvert, saxEventHandler, saxFilter, saxPurger.

Rapport

- **Alle:** Forord, Introduktion til Problem, Konklusion, Arbejdsfordeling, Logbog, Arbejdsblade, Samarbejdsafale.
- **Ans:** Analyse af design, Grupper, Evaluering og refleksion over proces.
- **Bjørn:** Filtrering og klargøring af data (introduktion og Krak), Tekstuel repræsentation af ruter, Visning af kort, Implementationsbeskrivelse, Evaluering af programmet.
- **Malthe:** Datastrukturer, Navigation, Adressegenkendelse, Implementationsbeskrivelse, Afprøvning, Forventningstabeller.
- **Mark:** Filtrering og klargøring af data (OSM), MVC, Brugervejledning.

11.5 Logbog

Logbog

17.05.2014 - 21.05.2014

- Rapportskrivning og sidste tweaks til kodebasen.

17.05.2014

- Ekstraordinært og 100% valgfrit lørdags møde. Mark var der ikke.
- Rigtig fint møde hvor vi fik drøftet en hel del ting relateret til rapporten.
- Vi besluttede os for at optimere vores QuadTræer på trods af at det var langt over tid og vi burde have skrevet rapport. Det er på nuværende tidspunkt uklart om det var investeringen værd.

14.05.2014

- Fremmøde: Ans havde ikke mulighed for at være der, alle andre var der.
- Kort dag. Stillede spørgsmål til instruktører, primært om rapportskrivning og A* algoritme, og gik hjem igen for at skrive.

12.05.2014

- Fremmøde: Mark var syg, alle andre var der
- Vi planlagde rapportskrivningen i rimelig detalje, og uddelegerede de fleste afsnit
- Evaluerede lidt på projekt-processen generelt
- Diskuterede diverse afsnit i rapporten
- Fik spurgt om vores problemer angående vores program, og fik svar på spørgsmålene. Kortet har nu rimelig performance på mac, og OSM data kan indlæses indenfor rimeligt tidsrum.

07.05.2014

- Fremmøde: Alle var der :)
- Vi havde code-freeze i dag, men ved dagens afslutning er vi kun næsten der hvor vi ville være. Centrale features i OSM data mangler, og vi har problemer med Mac. Udover det er vi sådanset færdige jævnfør listen over planlagte features.
- Evaluering af produktivitet:
 - En god produktiv dag. Vi synes vi kom langt.
 - Bjørn sad meget og arbejdede med en bug der ikke ville fanges og kun fandtes på hans computer
- Evaluering af samarbejde:
 - Fint samarbejde på tværs.

06.05.2014

- Fremmøde: Malthe og Bjørn var der. Ans var hjemme og passe sin syge lillebror, og Mark arbejdede fra DIKU.
- Evaluering af produktivitet

- Her på ITU gik det ganske godt fremad, og vi havde indtryk af at tingene også skred fremad for Mark.
- Evaluering af samarbejde
 - Godt samarbejde med Mark selvom han ikke var fysisk tilstede

05.05.2014

- Fremmøde: Alle var der, Mark en anelse forsinket.
- Evaluering af produktivitet
 - Fin produktivitet, gode fremskridt med programmet.
- Evaluering af samarbejde
 - Gode til at supplere hinanden.
 - Godt humør.

30.04.2014

- Fremmøde: Alle er der.
- Evaluering af produktivitet:
 - Malthe, Mark og Ans arbejdede hver især med adressegenkendelse, OSM parsing, og dynamiske vejtykkelser.
 - Bjørn hjalp til med de forskellige opgaver når der var problemer.
 - Ikke vanvittigt høj produktivitet da meget af dagen gik med bugfixing hvad angår adressegenkendelse og OSM parsing.
 - Ans fik lavet en implementation af dynamiske vejtykkelser som fungerede godt.
- Evaluering af samarbejde:
 - Generelt godt samarbejde om tingene.

28.04.2014

- Fremmøde: Bjørn var syg og havde meldt afbud.
- Været til evaluering af vores første del. De var umiddelbart glade.
- Evaluering af produktivitet:
 - Malthe: Vi har ikke fået så meget, har siddet fast i problemer.
 - Mark: Har også siddet fast i problemer, så er ikke fået meget længere, tager hjem og arbejder videre på det-
 - Ans: Dynamisk vejtykkelse ved zoom er fixet, så livet er fedt
- Evaluering af samarbejde:
 - Vi har arbejdet solo koncentreret. Vi har haft enkelte spørgsmål til hinanden men det var meget begrænset

23.04.2014

- Fremmøde: Alle var der
- Evaluering af produktivitet:
 - Malthe: Vi har ikke fået så meget, har siddet fast i problemer.
 - Bjørn: Gik rimelig godt i starten, men sad efterfølgende fast i problemer.
 - Mark: Slow but steady. En hel del småproblemer, men har løst det meste.

- Ans: Ok, fik lavet vindue. Stødte senere på problemer med vejtykkelser.
- Evaluering af samarbejde:
 - Vi har været gode til at hjælpe hinanden når det var nødvendigt, og ellers arbejdet selvstændigt.

21.04.2014

- Vi holdt påskeferie onsdag den 16. april.
- Fremmøde
 - Alle til tiden, flot på en helligdag
- Evaluering af produktivitet
 - En hel del tid brugt på bugfixing og generel diskussion om strategi og funktionalitet.
 - Vi er nået ret langt i løbet af påskeferien med pathfinding og OSM, thumbs up til Mark og Malthe.
- Evaluering af samarbejde
 - Godt samarbejde omkring bugfixing.

14.04.2014

- Fremmøde
 - Malthe, Ans og Bjørn nogenlunde til tiden
 - Mark lidt senere, måtte gå tidligt pga. træthed, gruppen stod bag denne beslutning
- Vi diskuterede den udvidede funktionalitet og arbejdede hver især videre med vores arbejdsopgaver.
- Evaluering af produktivitet
 - Malthe er ikke glad for sin produktivitet i dag. Han har ikke opnået noget i dag føler han. Bjørn ser mere optimistisk på dette og tænker at Malthe har gjort sig nogle erfaringer.
 - Mark havde, pga. træthed, heller ikke den store produktivitet.
 - Ans: Udemærket, fik arbejdet videre med løsning, men havde håbet på at blive færdig.
 - Bjørn synes han har haft en ganske okay produktivitet, i og med at han beskæftigede sig med en hel del bugs og tweaks der var hurtige at løse.
- Vi har aftalt at dem der har tid til det arbejder lidt videre i ferien.
- Evaluering af samarbejde
 - Gode diskussioner om de forskellige løsninger, og samarbejde om at finde fejl.

09.04.2014

- Fremmøde
 - Alle var der, yayyyy. :o)
- Vi lavede en arbejdsliste over de funktionaliteter som vi skal lave:
 - **Must have**
 - Shortest path algorithm (1)
 - Ny søge-struktur-thore-overwhelming-ting

- Klik-gui-punkt-til-punkt-shizzle
- Usability test (2)
 - Minimum 3 subjects
 - Teste brugervenlighed
 - Handle på resultater
- Performance (3)
 - 10 fps
 - Performance optimize
 - Shortest path skal køre på under 5 sekunder i DK
- OSM (4)
 - Parse OSM XML with SAX
 - UI til at vælge mellem KRAK og OSM ved programstart
- Valgfri (5)
 - Første punkt... Click and drag/move shortest route ui...
- Route descriptions (6)
 - Reducer antal punkter...
 - Swing ting eling (liste i sidebar/vindue/whatever)
 - Output (raw text)
- **Should have**
 - Purge data (OSM)
 - Performance fix for pan
- **Could have**
 - Adressegenkendelse og indtastning (forbundet til shortest path ting...)
 - Extension (2), map shading
 - Vejtykkelser dynamisk ved zoom (lækkert)
 - Relativ zoom til mus ved markør jaaah
 - FPS counter i øverse venstre hjørne
- Vi delte nogle af opgaverne ud, og aftalte hvad der skulle laves til næste gang (til efter påskeferien i Marks tilfælde).
- Codefreeze 07.05.2014 (efter denne dag kodes der ikke mere, og rapport afsnit fordeles)
- Evaluering af produktivitet:
 - Fik rimelig effektivt uddelegeret opgaver og planlagt fremad.
- Evaluering af samarbejde:
 - Gik fint, godt samarbejde om at danne overblik

31.03.2014

- Fremmøde
 - Mark, Malthe og Bjørn mødte op. Ans var et par timer forsinket, og måtte gå igen få timer efter.
- Vi fik skrevet det sidste af rapporten færdig.
- Dette var den sidste dag vi arbejdede på første fase af projektet. Vi var der i alt 10 timer hvilket viste sig at være lidt for mange, men der var på dette tidspunkt ikke rigtig nogen alternativer.

- Evaluering af produktivitet:
 - Høj produktivitet det meste af dagen, og mange gode diskussioner omkring rapportens indhold, såvel som grammatik og det danske sprog generelt.
- Evaluering af samarbejde:
 - God arbejdsfordeling og teamwork. God brug af individuelle styrker og ressourcer.

26.03.2014

- Fremmøde
 - Alle var der - jubiii.
- Gruppen glemte at lave logbog og arbejdsblade mandag d. 24. I korte træk formåede vi at:
 - Gøre programmet færdigt og fixe bugs.
 - Gruppen dannede sit et overblik over rapportskrivningen og skrev en outline.
 - Vi brugte læringsmålene som udgangspunkt (godt tip fra Sune).
- Vi fortsatte arbejdet på rapporten
 - Læste hinandens afsnit igennem og gav nogenlunde konstruktiv feedback.
 - Vi diskuterede en hel del omkring hvad der skulle stå i hvilke afsnit.
 - Blev lidt mere fortrolige med LaTeX.
- Evaluering af produktivitet:
 - Vi var ikke særlig produktive, men fik diskuteret en masse ting.
- Evaluering af samarbejde:
 - På trods af at der et par gange i forbindelse med diskussioner blev lidt heftig stemning som følge af at gruppens deltagere var meget passionerede, gik samarbejdet fint.

19.03.2014

- Fremmøde
 - Lidt forsinket, men alle var der
- Vi arbejdede på fire forskellige ting i dag
 - Malthe arbejdede på panning. Det viste sig at være svært, og den nuværende implementation er ikke optimal.
 - Ans arbejde på at få opdelt vinduet i to paneler, således at der nu er en label i bunden af vinduet der på sigt skal vise det nærmeste vejnavn.
 - Mark implementerede zoom vha. keyboard. Implementationen fungerer som planlagt.
 - Bjørn implementerer selection zoom. Det var svært, men implementationen fungerer som planlagt.
- Vi havde problemer med internettet. Det blev løst for Mark, Malthe og Bjørns vedkommende med ethernet kabler. Ans brugte det trådløse uden alt for store problemer.
- Vi havde igen i dag vanskeligheder med Git. Det hjalp lidt at vi endelig er begyndt at forstå det mærkelige konflikt hieroglyffer. Brugte stadig en hel del tid.
- Følgende opgaver tilbage

- Mouse panning (delvist implementeret)
 - Show closest road (label er implementeret)
 - Zoom details
 - Keyboard zoom (er implementeret, men skal reagere på de rigtige taster)
 - Mouse roll zoom (kan hurtigt udledes fra keyboard zoom)
- Vi har lavet en halv aftale om at mødes i morgen ~14.30 da vi ikke helt er der vi gerne ville være planmæssigt med vores implementation.
- Evaluering af produktivitet
 - Ret høj produktivitet, eftersom vi havde gang i fire forskellige ting, og lykkedes med at kombinere dem til sidst.
 - Som nævnt ovenfor sørkede forsinkelse, internet, og Git arbejdet lidt.
- Evaluering af samarbejde
 - Første dag vi allesammen skrev kode. Kæmpe fremskridt.
 - Mark og Ans undervurderer deres evne til at skride god og korrekt kode (siger Bjørn).

17.03.2014

- Fremmøde
 - Mark lidt for sent, meldte forsinkelse
 - Ans et par timer for sent, meldte forsinkelse
 - Stadig godt at der bliver meldt forsinkelser. Malther og Bjørn startede arbejdet som planlagt
- Vi har fået refaktoreret væsentlige dele af vores kode
 - Translator er skrevet om og der er blevet tilføjet hjælpefunktioner
 - Event listeners er blevet rykket ud i controller fra view
 - Fået skrevet klasse der fjerner unødig data fra datafiler
- Vi har problemer med at kortet er strukket en smule i højden lige nu
 - Vi har ikke kigget efter problemet endnu, løses næste gang
- Følgende opgaver er tilbage
 - Mouse panning
 - Mouse select zoom
 - Mouse roll zoom
 - Keyboard zoom
 - Zoom details
 - Show closest road
- Vi havde igen store vanskeligheder med Git der kostede os en time. Det er fortsat vanskeligt for os at merge branches og bare generelt få det til at gå op i en højere enhed.
 - Ville være en fordel hvis andre end Mark havde en fornemmelse af hvad Git er, sådan at vi kunne samarbejde om at løse problemerne
- Evaluering af produktivitet
 - Vi har været ret koncentrede og der har været god produktivitet
 - Vi har taget hånd om nogle svære områder i vores kode, og refaktoreret ret meget, hvilket betyder at vi ikke har fået lavet særlig meget ny funktionalitet

- Uddelegering af opgaver er gået fint, og vi har næste hele tiden haft gang i mere end én ting ad gangen
- Evaluering af samarbejde
 - Vi havde stadigvæk lidt svært ved at forstå hvad hinanden talte om i forbindelse med Translator klassen, men det løste sig.

12.03.2014

- Ans meldte afbud eftersom han havde været involveret i et mindre trafikuheld.
- Vi startede med at skrive de ting ned vi manglede i forhold til kravspecifikationen.
- Derefter forsøgte vi at arbejde på flere ting ad gangen ud fra de nedskrevne punkter, hvilket gik ganske godt.
 - Malthe blev god til at flette konfliktende commits.
 - Vi andre blev gode til rollbacks.
- Evaluering af produktivitet:
 - Produktiviteten var meget højere i dag end ved de foregående møder fordi vi lykkedes med at uddelegerede opgaver og arbejde på flere forskellige moduler på én gang. Her vil vi gerne sige tak til MVC framework.
 - Listen vi lavede i starten gav et godt overblik.
 - Vi var meget opslugte af arbejdet, da det var spændende, og var på den måde meget effektive da vi som konsekvens holdt minimale pauser.
- Evaluering af samarbejde:
 - Det gættes godt i dag. Ikke så meget at tilføje, når der ikke rigtig er problemer.

10.03.2014

- Mark udeblev, og vi havde ikke held med at få fat i ham.
- Ans, Malthe og Bjørn satte sig og kodede sammen, og fik påbegyndt controller modulet. Efter dagens arbejde kunne vi tegne kortet for første gang.
- Evaluering af produktivitet
 - Produktiviteten er stadig forholdsvis lav, hovedsageligt fordi vi har problemer med at uddelegerere arbejdet, og derfor kun arbejder på én ting ad gangen.
 - Vi havde nogle problemer med kompilering relateret til pakker som tog en hel del tid at løse.
- Evaluering af samarbejde
 - Skidt at vi ikke ved hvorfor et af gruppemedlemmerne ikke er her, og ikke har fået etableret kontakt.
 - Fint humør og progression.
- Vi skal i dag til status-samtale med Søren.

05.03.2014

- Mark er syg og har meldt afbud.
- Vi startede dagen ud med at tale forbi hinanden, i en diskussion om nye klasser og strategier. Der gik der noget tid på, som skabte dårlig stemning.

- Efterfølgende valgte vi en ny strategi. Vi satte os ned og kodet sammen, det hjalp, da vi havde et fælles referance punkt, og der var ikke tvivl om hvad det var vi talte om. Det resulterede i at vi fik lavet et godt stykke arbejde og vi fik talt tingene igennem og fik afklaret nogle punkter.
- I dag nåede vi at lave parser nogenlunde færdig. Vi er nu klar til at skrive noget controller kode, med det formål at for tegnet noget.
- Evaluering af produktivitet
 - Produktiviteten har været lav, men vi synes at kvaliteten af vores arbejde har været høj.
 - Hvis vi havde formået at skrive flere dele af koden samtidig, havde vi nok været produktive.
 - Ans er stadig ikke på toppen og da mark har manglet, og det har været en af faktorerne til det mindre produktive arbejde.
 - Man har godt kunne mærke at Mark ikke har været her idag, da han tit har godt overblik og tit kan se løsningen.
- Evaluering af samarbejde
 - Samarbejdet har været godt i dag, taget i betragtning af krisen i starten.
 - Stadig godt at der bliver meldt afbud.
- Vi har stadig ikke nået at nå den arbejdsfordeling hvor Ans og Mark skriver koden.

03.03.2014

- Ans er syg og har meldt afbud.
- Vi har kigget på Krak data og parsere, og er i gang med at implementere quadtree. Vi startede med at implementere det med ArrayList som datastruktur, og fortsatte derefter med quadtrees. Størrelsen på datakilden gjorde det meget svært at debugge. Vi havde et meget specifikt problem, der nu er løst, som blev forårsaget af at et datapunkt lå på grænsen mellem to quads. Der er også pt. et problem med at det kun er to tredjedele af dataen der bliver indlæst, resten ved vi ikke hvor er blevet af. Nu er de der allesammen, alle er glade.
- Vi er i gang med at lave noget kode der kan tegne, og har til dels løst problemet. Dog er der stadig nogle strukturelle overvejelser i forbindelse med skalering og andre funktioner der kræver at kortet opdateres med relativt hurtigt interval.
- Evaluering af produktivitet
 - Vi har fået lavet en del.
 - Mark har oplevet nogle gange at det er lidt svært at fokusere nogle gange, primært på grund af træthed.
 - Heraf kan vi lære, at vi altid skal drikke rigelige mængder af koffeinholdige substanser.
 - Vi har lavet en masse fejl, og det har selvfølgelig resulteret i at vores effektivitet ikke har været 100% optimal. På den anden side har vi lært en hel del om Swing, og lavet nogle designmæssige overvejelser vi ellers ikke ville have gjort.
- Evaluering af samarbejde

- Godt at der bliver meldt afbud.
- Godt humør i dag, også når vi laver noget lort som vi ikke kan bruge til noget, men bruger meget tid på.

26.02.2014

- Det vi lavede
 - Bookede en tid hos Søren
 - Git
 - Oprettede Github brugere
 - Fik inviteret alle til det rigtige Github repository
 - Leg med git generelt
 - Vi vælger Krak. Det er fedt og nemt fordi der allerede er skrevet en parser.
 - Indledende tanker og strategi
 - Vi vil lave systemet så modulært, at datakilden kan udskiftes.
 - Model view controller
 - View
 - Skal bare tegne
 - Ingen zoom eller pan, sker i controlleren
 - Statusbar
 - Model
 - Quad-struktur til kort data
 - Zoom
 - Plus minus
 - Via scroll
 - Tegne firkant ting
 - Reset
- Næste gang
 - Kigge på Krak data og parsere
 - Lave noget kode der kan tegne

24.02.2014

- Vi skrev samarbejds aftale.
- På onsdag:
 - Git
 - Indledende tanker/strategi
 - Valg af datakilde
 - Valg af samtaletidspunkt med Søren

11.6 Arbejdsblade

Arbejdsblade

Opgave	OSM til KRAK-konvertering
Hvem	Mark
Hvornår	igennem hele forløbet
Hvad skal der laves	begrænse OSM-datasættet og konvertere det til Krak-format
Hvordan har vi lavet det	Ved at benytte Saxparsing, og læse skrive fra filer, mens vi konverterer den ønskede data
Problemer	OSM-dataen er enorm, og følger ikke stringent dets eget struktur
Næste gang	

Opgave	Kør rendering på dedikeret tråd
Hvem	Malthe og Bjørn
Hvornår	17.05.2014
Hvad skal der laves	For at programmets brugergrænseflade ikke låser mens der renderes, er det nødvendigt at lægge renderingen på sin egen tråd.
Hvordan har vi lavet det	RenderThread Erstatter renderRectangle() med klasse, som udvider Javas Thread. Al rendering af tiles køres nu i denne klasses run()-metode. Hver gang der skal renderes oprettes nu en ny renderingstråd, som selv er ansvarlig for at rendere dens tiles. Enhver tile som gives til en tråd markeres som igangværende, og kan således ikke gives til andre tråde. Dette sikrer at en tile ikke dobbeltrenderes. Når en tråd er færdig med at hente de nødvendige edges, synkroniserer den med linepoolen, som genbruger lines og forhindrer constructor overheadet på disse, samt bufferen. Disse kan således anvendes og skrives til uden at der forekommer overlappning og dertilhørende renderingsfejl.
Problemer	

Næste gang	
------------	--

Opgave	Repræsenter en edge i samtlige quads, som den gennemskærer
Hvem	Bjørn og Malthe
Hvornår	17.05.2014
Hvad skal der laves	For at forbedre den nuværende løsning for linjeproblemet, hvor en linje indsættes i quadtræet efter dens centrumspunkt, og således ikke kan garanteres optræder i en hvilken som helst quad, som den gennemskærer, er det nødvendigt at ændre håndteringen af indsættelse og forespørgsel af edges.
Hvordan har vi lavet det	<p>Insert</p> <p>Der indsættes nu i det QuadTree, hvorpå metoden kaldes, ellers enhver af dennes børn. En edge kan således nu indsættes i mere end ét QuadTree.</p> <p>QueryRange</p> <p>For at håndtere, at en edge kan optræde flere gange, uden nødvendigvis at returnere disse dubletter, er et HashSet anvendt. Dette sikrer at en edge kun optræder én gang i resultatsættet.</p> <p>Køretid og morgen</p> <p>Fordi en quad nu garanterer, at en edge som passerer igennem denne også optræder i dens edge liste, er det ikke længere nødvendigt men en morgen på halvdelen af den længste edge. Således returneres (og derved tegnes) der ikke unødig mange edges, hvorfor køretiden ses forbedret.</p>
Problemer	
Næste gang	

Opgave	A*
Hvem	Malthe
Hvornår	13.05.2014
Hvad skal der laves	Udskift Dijkstras med A*
Hvordan har vi lavet det	Implementeret A* algoritmen som den kendes.

	<p>$g(x)$: kørselstid af kanten mellem to noder $h(x)$: tiden det tager at køre i fugleflugtslinje fra node til destination med 130 km/t. (valgt så h umuligt kan være pessimistisk)</p> <p>A* synes at være hurtigere end Dijkstras i alle brugstilfælde, på trods af at der ved Dijkstras kan ændres destination uden noget overhead.</p>
Problemer	Den valgte heuristik er ikke perfekt, da det sjældent (aldrig) vil være muligt at køre 130 km/t i lige linje til målet.
Næste gang	

Opgave	Kør adressesøgning på dedikeret tråd
Hvem	Malthe
Hvornår	01.05.2014
Hvad skal der laves	For at adressesøgningen kan foregå så responsivt som muligt, men uden at fryse brugergrænsefladen, er det nødvendigt at lægge levensthein-algoritmen i en anden tråd.
Hvordan har vi lavet det	<p>FindThread Tråd hvorpå edit-distance algoritmen kører. Denne tråd yielder til andre tråde hver gang en distance mellem to strenge er fundet (altså en gang per kørsel i hovedløkken). Derved har scheduleren fri mulighed for at inddrage CPU-tiden. Algoritmen kører som hidtil. Når et resultat af fundet notificeres det objekt, som blev givet af <code>getFindThread()</code>-metodens callback parameter. Derudover muliggøres kaldet af <code>getResult()</code>, som returnerer et array med de bedst matchende strenge.</p> <p>UpdateThread Denne tråd oprettes af <code>AddressFieldListener</code> og sørger for at modtage resultaterne fra <code>FindThread</code>. Tråden er nødvendig, da der skal ventes på resultater. Uden tråden ville swing således fryse indtil resultatet findes.</p> <p>Generelt Trådene vedligeholder en id ved oprettelse, og matcher denne ikke længere, dræbes tråden. Det vil sige at hvis mere end én <code>FindThread</code> eller mere end én <code>UpdateThread</code> dræber disse sig selv sikkert.</p>
Problemer	

Næste gang	
------------	--

Opgave	Fake zoom
Hvem	Bjørn
Hvornår	07.05.2014
Hvad skal der laves	Når der zoomes kræver det at en masse tiles tegnes. Det får programmet til at hakke.
Hvordan har vi lavet det	Der gemmes et snapshot når der zoomes som skaleres. Efter 200 millisekunder uden zoom renderes kortet.
Problemer	Der er opstået et mærkværdigt problem hvor programmet tager 50-60% cpu når der pannes på mac. Det samme tager 4-5% på windows og kører silk-smooth.
Næste gang	

Opgave	Optimering af tiler
Hvem	Bjørn
Hvornår	06.05.2014
Hvad skal der laves	Tiler skal tegne hurtigere, især når der skiftes zoomniveau.
Hvordan har vi lavet det	Tiles grupperes i sammenhængende rektangler og tegnes samlet. Dette giver en masse fordele, der får det til at gå meget hurtigere end at tegne dem individuelt.
Problemer	Lidt problemer med BufferedImage og kopiering af billeddata. Ble dog løst.
Næste gang	

Opgave	Adressesøgning GUI
Hvem	Malthe
Hvornår	06.05.2014

Hvad skal der laves	Drop-down gui til foreslåning af adresser
Hvordan har vi lavet det	<p>DropTextField</p> <p>Klasse som udvider funktionaliteten af et <i>JTextField</i>. Tekstfeltets originale funktionalitet bevares i store træk, men der anvendes en <i>JPopupMenu</i> som drop-down menu. Klassen bibeholder MVC-stilen og gør således intet andet end at repræsentere givne strenge. Det er således op til den nedenstående controller-kasse at afgøre hvilke strenge, der skal optræde i listen, samt hvornår listen vises og hvordan der skal handles når brugeren foretager en tekstinput.</p> <p>AddressFieldListener</p> <p>Controller-kasse til DropTextField, som muliggør adresseforeslag i denne. Ved at lytte til brugerinput i tekstfeltet, søges der efter de bedst passende adresser såfremt inputtet er længere end to tegn. Der ses en forsinkelse på 200 millisekunder før ovenstående beregnes, hvilket forhindrer inputlag.</p>
Problemer	Piletastinput i drop-down listen virker ikke.
Næste gang	Fix ovenstående.

Opgave	Ny mappestruktur
Hvem	Bjørn
Hvornår	05.05.2014
Hvad skal der laves	Krak og osm data skal flyttes ud i mapper, og det skal muliggøres at loade det ene eller det andet datasæt.
Hvordan har vi lavet det	Data flyttes i mapper. Loaderen modtager et parameter "dataSet" som definerer hvilken mappe den skal loade data fra.
Problemer	
Næste gang	

Opgave	Adressesøgning
Hvem	Malthe
Hvornår	05.05.2014

Hvad skal der laves	Adressesøgning med shortest-edit-distance algoritme
Hvordan har vi lavet det	<p>AddressFinder Klasse som, med en liste af adressenavne og tilsvarende Edge-objekter, kan finde de veje, hvis navne er tættest på en given tekststreng. Dette gøres ved hjælp af en algoritmisk implementering af Levenshtein distance. Udover de normale parametre i algoritmen, er der tilføjet gratis suffix-ændringer. Dvs at edit-distancen fra "rued" til "rued langgaards vej" er 0, fordi "langgaards vej" er et suffix til "rued". Klassens <i>find</i>-metode kan kaldes med en integer-værdi, som afgør hvor mange veje, metoden skal returnere.</p> <p>AddressFieldListener DocumentListener-klasse, som lytter til hvornår til- og frafelterne ændres. Når en ændring foretages, startes en nedtælling på 200 millisekunder, hvorefter de tætteste vejnavne findes. Tidsnedtællingen er til stede for at sikre, at indtastning i tekstfeltet kan foregå flydende, da de tager et øjeblik at udregne edit-distancerne.</p> <p>AddressButtonListener ActionListener-klasse, som lytter til hvornår rute-knappen trykkes på. Når den trykkes på rute-knappen beregnes et kortest-vej-træ fra den første Edge, hvis navn svarer til teksten i fra-feltet. Herefter findes den korteste vej til den første Edge, hvis navn svarer til teksten i til-feltet. Denne vej vises på kortet og rutevejledning opdateres i navigationsbjælken i venstre side.</p>
Problemer	Adressesøgningen kan ikke på nuværende tidspunkt tage højde for husnumre. Derudover er der små gui fejl i navigationsbjælken.
Næste gang	Ret gui fejl. Der stræbes ikke efter at implementere husnumre.

Opgave	Beregne manglende hastighedsdata
Hvem	Bjørn
Hvornår	05.05.2014
Hvad skal der laves	Der er manglende hastighedsdata i krak-datasættet, hvilket gør afgørende vejstykker ikke har nogen hastighed. Dette går selvfølgelig ikke. Derfor laves denne tilføjelse til DataPurger som retter op på ovenstående.

Hvordan har vi lavet det	Hvis der ikke findes hastighedsdata for et vejstykke beregnes det ud fra længden på vejstykket og den estimerede kørselstid.
Problemer	
Næste gang	

Opgave	Dynamisk vejtykkelse ved zoom
Hvem	Ans
Hvornår	28.04.2014
Hvad skal der laves	Når der zoomes skal vejbredderne for de forskellige vejtyper, bliv bredere afhængigt af zoom. Bredderne skal ikke blive for voldsomme, og skal derfor vokse med omkring 20 pct.
Hvordan har vi lavet det	Inde i klassen Group har vi lavet nogle faste bredder til de forskellige vejtyper. Bredderne for en linje kaldes i renderTile metoden i Tiler klassen. I Tiler klassen er der blevet lavet en metode lineWidth. Når zoom har en værdi der er højere end 0,15 divideres linjens bredde med zoom værdien, og 20 pct af den værdi lægges til den aktuelle linje bredde.
Problemer	Vi startede med at sige vejbredde divideret med zoom værdien. Det medførte at vejene fik nogle voldsomme bredder.
Næste gang	

Opgave	Zoom mod mus
Hvem	Bjørn
Hvornår	23.04.2014
Hvad skal der laves	Når der zoomes skal centrum gå mod mus, således at musen hele tiden peger på det samme punkt på kortet.
Hvordan har vi lavet det	Prøvet en hel del forskellige implementeringer uden rigtigt at vide om de var korrekte geometrisk. Et naivt forsøg.
Problemer	Ja, det lykkedes ikke at implementere det. Der var ikke nogen i gruppen der kunne regne ud hvordan centrum skulle forskydes.
Næste gang	Vi stopper med denne opgave indtil vi ved hvordan det kan

	beregnes i vores tilfælde.
--	----------------------------

Opgave	Rute beskrivelser
Hvem	Bjørn
Hvornår	23.04.2014
Hvad skal der laves	En liste af edges skal blive til en beskrivelse af en rute
Hvordan har vi lavet det	<p>Lavede en ny path klasse som indeholder listen af edges. Løber edges igennem og kigger på om vejnavnet ændrer sig. Hvis vejnavnet ændrer sig tilføjes linje til beskrivelse.</p> <p>Involverede også lidt arbejde med korrektioner til rækkefølgen af edges returneret af ShortestPath og et fix til TestShortestPath.</p> <p>Afsluttende lidt kode for at få vist rutebeskrivelsen i sidebaren.</p>
Problemer	Nej
Næste gang	

Opgave	FirstWindow
Hvem	Ans
Hvornår	23.04.2014
Hvad skal der laves	Der skal laves et nyt vindue hvor man kan vælge Krakdataset eller OSM.
Hvordan har vi lavet det	Der er lavet et nyt vindue som består af to knapper. Main klassen er blevet ændret til, at det er FirstWindow der kommer frem når vinduet køres. Det tidligere indhold i Main klassen er blevet rykkes til en metode i FirstWindow klassen, som kaldes når der trykkes på Krak knappen. Når der trykkes på en af knapperne lukkes vinduet.
Problemer	Tvivl omkring hvad der skulle ske med FirstWindow når der trykkes på en knap.
Næste gang	Tilføje evt progressbar og overveje om vinduet skal laves som et dialogvindue.

Opgave	Korrekt findClosest-metode
Hvem	Malthe
Hvornår	21.04.2014
Hvad skal der laves	findClosest-metoden skal rettes, så denne finder den faktisk tætteste Edge og ikke blot den Edge hvis centrum er tættest på.
Hvordan har vi lavet det	<p>Den nuværende løsning tager udgangspunkt i en 10x10 søgebox omkring musen som fordobles hvis ingen Edges findes. Når mindst én Edge findes i denne søgebox, gennemløbes samtlige fundne Edges og den hvis centrum er tættest returneres.</p> <p>Metoden hvorpå edges findes, ved fordobling af søgeboxen, ønskes bibeholdt, da denne har vist sig at være meget hurtig. Algoritmen bag metoden er således ændret så fremgangsmåden er som følger:</p> <ol style="list-style-type: none"> 1. Fordoble en query box indtil mindst en edge er fundet. (Om denne SKAL være med navn er bestemt af bool parameteren <code>withName</code>.) 2. Udvid nævnte query box endnu engang med den halve længd på den længste edge i quad træet. Dette sikrer at ingen edges bliver "snydt" grundet deres centrum koordinater. 3. Gennemløb alle fundne edges og beregn distance fra punkt til linje. Udvælg den korteste - muligvis betinget af et navn. (Distanceformlen finder det vinkelrette punkt på linjen til det søgte punkt. Under- eller overskridet dette linjesegmentet, benyttes henholdsvis start- eller slutpunktet.)
Problemer	Ved nærmere afprøvning har metoden vist sig at fungere i langt de fleste tilfælde. Der kan dog ske fejl når en Edge er meget lang (og musen således kan være langt fra dens centrum men tæt på selve linjen).
Næste gang	Implementer <code>findClosestNode</code> til at finde knudepunktet tættest musen.

Opgave	Bugfixing og optimering
Hvem	Bjørn

Hvornår	21.04.2014
Hvad skal der laves	En række små rettelser for at få programmet til at køre bedre.
Hvordan har vi lavet det	<p>Glasspane Glasspane virkede ikke. Fixet ved at rette translateToView metoden.</p> <p>Basicstroke Basicstroke objekter blev lavet hver gang der skulle tegnes en linje. Løst ved at cache basicstroke objekter, og reducere antallet af linjebredder til heltallene mellem 0 og 99.</p> <p>Tilesize En fejl ved tidligere commit gjorde at den tilesize der beregnes hver gang der ændres zoomniveau var alt for stor. Løst ved at reducere den til en fjerdedel af gennemsnittet af bredden og højden, hvilket resulterer i at der typisk skal tegnes $5 * 5$ tiles (én mere i bredden og højden i forhold til $4 * 4$ da section aldrig passer perfekt på tiles).</p>
Problemer	Nej
Næste gang	

Opgave	Kortest-vej algoritme
Hvem	Malthe
Hvornår	21.04.2014
Hvad skal der laves	Kortest-vej algoritme skal implementeres og kobles til musen
Hvordan har vi lavet det	<p>ShortestPath Dijkstra's algoritme er valgt som grafsøgealgoritme da denne eftersigende er den hurtigste når et kortest-vej træ skal dannes. Dette gør det muligt at udvælge et udgangspunkt og efterfølgend flytte destinationen uden nævneværdig computationstid.</p> <p>MouseHandler Der er tilføjet et kodesegment i MouseHandler-klassen, som aktiveres ved enkeltklik med musen. Her differentieres mellem højre- og venstrekliek. Ved et højreklik markeres startpunktet for kortest-vej søgningen, hvorefter et tilhørende kortest-vej træ genereres. Ved et venstrekliek markeres slutpunktet for kortest-vej</p>

	søgningen og brugergrænsefladen opdateres med den korteste rute.
Problemer	For at fremtidssikre algoritmen er denne udarbejdet med en orienteret graf. Dette er gjort da visse veje er ensrettede mm. Krak datasættet er ikke orienteret, hvorfor alle linjer tilføjes i grafen i begge retninger.
Næste gang	

Opgave	Data til fast format
Hvem	Bjørn
Hvornår	14.04.2014
Hvad skal der laves	Data skal indlæses til fast format i loader.
Hvordan har vi lavet det	Vektoren i øverste venstre hjørne og nederste højre hjørne bliver fundet, og hele datasættet bliver efterfølgende skaleret til værdier mellem 0 og 1000 på x og y aksen. En hel del andre småændringer var nødvendigt for at få altting til at spille. Før denne ændring var maksimum og minimum værdierne ”hardcodet”, og data blev ikke skaleret til fast format.
Problemer	Nej.
Næste gang	

Opgave	Sidepanel til GUI
Hvem	Ans og Bjørn
Hvornår	14.04.2014
Hvad skal der laves	Et sidepanel som skal indeholde søgetekstfelter, og udskrift fra rutebeskrivelse. Derudover skal der være en knap, der skal kunne skjule og tilføje panelet.
Hvordan har vi lavet det	Vi har lavet det ved at tilføje et JLayeredPane hvor vi har canvas og et sidepanel i. JLAYEREDPane har den funktion at det er sat til null layout. Sidepanelet består af JToggleButton, der viser og skjuler panelet og et indre panel hvor vi har tilføjet tekstfelter osv.

Problemer	JLayeredPane gav det problem at komponenterne i panelet ikke resizede. Vi lavede derfor en resize metode som bliver kaldt i en ComponentListener og også når der bliver trykket på toggle knappen.
Næste gang	Indsætte scrollpane. Man skal kunne scrolle ned, da rutebeskrivelsen kan blive meget lang. Rykke knappen ned i bunden og se på layout..

Opgave	Grafstruktur
Hvem	Malthe
Hvornår	14.04.2014
Hvad skal der laves	Implementering af en grafstruktur over noder og kanter, således at kortest-vej algoritmer kan benyttes.
Hvordan har vi lavet det	Der er implementeret en orienteret grafklasse Graph som repræsenterer forbindelser mellem knuder. Disse forbindelser er denne implementation givet som Edge-objekter, da disse således let kan tegnes senere. Grafen indlæses sideløbende med det resterende data under programopstart.
Problemer	Da Kraks data ikke er retningsorienteret kan der opstå problemer med implementeringen af kortest vej algoritmen.
Næste gang	Implementer kortest vej algoritme.

Opgave	Tiler
Hvem	Bjørn
Hvornår	14.04.2014
Hvad skal der laves	Renderingen skal effektiviseres.
Hvordan har vi lavet det	Translator skal skrottes, og Tiler skal skrives. Tiler sørger for at dele kortet op i små bitmaps, rendrere dem individuelt, og samle dem til skærbilleder.
Problemer	Ja. Mange. Det tog meget længere tid end forventet, og masser af svære bugs. Unit testing hjalp en hel del.

Næste gang	Fix bugs, gør gode pæn og læselig.
------------	------------------------------------

Opgave	Selection zoom
Hvem	Bjørn
Hvornår	19.03.2014
Hvad skal der laves	Selection zoom skal implementeres. Funktionaliteten fungerer således at man markerer et område ved at holde højreklik inde og flytte markøren, og når man slipper zoomes der ind på det markerede område.
Hvordan har vi lavet det	Brugt den MouseHandler klasse der allerede var implementeret, og brugt metoder fra Translator og klasserne Vector og Box. Implementerede først at der blev tegnet en firkant på kortet når man markede området, dernæst at der blev recentreret, og dernæst at der blev zoomet, hvilket lettede arbejdet.
Problemer	Nej.
Næste gang	

Opgave	Tilpasning af Window
Hvem	Ans
Hvornår	19-03-2014
Hvad skal der laves	Opdele Window i to paneler, så der er plads til at få fremvist "nærmeste vejnavn" nederst i vinduet.
Hvordan har vi lavet det	Opdelt Window i to paneler. Det ene panel er canvas, det andet panel er bottomPanel. Vi har brugt borderlayout til at opsætte Window. I bottomPanel er der et label, hvor vejnavnet skal fremvises.
Problemer	Ikke rigtigt nogle problemer. Lidt omkring valg af layout.
Næste gang	Få fremvist vejnavn i label.

Opgave	Paning
--------	--------

Hvem	Malthe
Hvornår	19.03.2014
Hvad skal der laves	Paning med musen skal implementeres, så der kan trækkes rundt på kortet intuitivt.
Hvordan har vi lavet det	<p>MouseHandler</p> <p>Metoderne <i>mouseDragged</i> og <i>mousePressed</i> samt <i>-Released</i>, som implementeres fra henholdsvis <i>MouseListener</i> og <i>MouseMotionListener</i> interfacene, anvendes til at observere musens opførsel.</p> <p>Når musen trækkes sendes hyppige kald til <i>mouseDragged</i> af Swing biblioteket. Disse anvendes til at udregne afstanden, som musen har bevæget sig mellem to drag-events. Denne afstand oversættes til en addering af center-koordinatet i translatoren, hvorefter kortet tegnes efter det nye center.</p>
Problemer	Første implementering tog udgangspunkt i et kvadratisk quadtræ men da dette ikke er tilfældet for modellen, var paningen skæv på den ene axe - kortet løb fra musen. For at udbedre dette anvendes nu musekoordinater, som er relative til modellens størrelse. Dette har dog haft den beklagelige bivirkning, at kortets bevægelse nu accelererer.
Næste gang	Udbedring af accelerering samt refaktorering af uoverskuelig kode.

Opgave	Flyt controller-dele ud af viewpakken
Hvem	Malthe og Ans
Hvornår	17.03.2014
Hvad skal der laves	View klassen Canvas indeholder listeners, som bør findes i controller-pakken. Disse skal derfor omskrives og flyttes.
Hvordan har vi lavet det	Klassen ResizeHandler i controller-pakken erstatter den indre klasse ResizeListener i Canvas. Denne anvender translatoren til at opdatere linjerne og derefter tegne dem. Herudover kører en timer i et halvt sekundt efter at programmet er blevet resized og aktiverer anti aliasing.
Problemer	Graphics objektet synes at bibeholde de gamle linjer, men dette blev løst ved at rydde objektet før der tegnes.

Næste gang	Flyt de resterende controller-dele ud af viewet.
------------	--

Opgave	Translatorpusning
Hvem	Bjørn, Malthe, Mark
Hvornår	Mandag d. 17/3 -
Hvad skal der laves	Vi skal restrukturere koden, og forkorte unødig lange eller unødig komplikerede kodestykker. Sikrer at der ikke er redundant kode
Hvordan har vi lavet det	Ved skrivning på 1-2 datamater. 2 par vågne øjne på hele forløbene
Problemer	Vores data er nu strukket ud i en høj slank form, der ikke passer til buttede Danmark
Næste gang	find out why

Opgave	Musen skal aktiveres
Hvem	Malthe
Hvornår	Mandag d. 17
Hvad skal der laves	ikke så meget, men der er forsøgt en del
Hvordan har vi lavet det	Vi har indset at det nok var smartere at få lagt vores listeners i control
Problemer	kom ikke så meget videre med resten af opgaven
Næste gang	Musen skal aktiveres med pan, og zoomfunktioner

Opgave	Fjern unødig data fra datafiler
Hvem	Malthe
Hvornår	16.03.2014
Hvad skal der laves	Da de originale datafiler indeholder meget unødig data, som aldrig tages i brug, skal disse omskrives.
Hvordan har vi lavet det	DataPurger Læser en tekstfil med kommaseparererede punkter (både tal og

	<p>strenge), og skriver dele af disse til en ny fil. Udvælgelsen af data afgøres med et indeksarray, hvori de ønskede kolonner optræder. Denne klasse kan både køres med argumenter (<input file> <output file> <index1,index2,index3...>) og uden argumenter. Køres klassen uden argumenter udvælges in- og outputfiler som standard datafilerne.</p> <p>EdgeData og NodeData De konstanter som indeholdte det unødige data er fjernet fra klasserne.</p>
Problemer	Der redigeres i det originale data.
Næste gang	

Opgave	Gruppering af data
Hvem	Malthe, Bjørn og Mark
Hvornår	12.03.2014
Hvad skal der laves	Undgå at samtlige edges tegnes ved ethvert zoom-niveau.
Hvordan har vi lavet det	<p>Edges er nu grupperet efter deres trafikale prioritet. Disse grupper er i skrivende stund defineret som følger:</p> <ul style="list-style-type: none"> • Highways (rød) • Main roads (blå) • Paths (grøn) • Pedestrian (gul) • Naval (cyan) • Other (sort) <p>Grupperingen er i datastrukturen foretaget ved at holde et quadtree array, hvis indeksering er ens med edge grupperne. Derudover tegnes forskellige edge grupper nu med forskellige farver som vist ovenfor.</p>
Problemer	Vi fandt to edge-typer, som ikke er defineret i Kraks dokumentation (type 0 og type 95.) Der er dog kun 8 af disse udefinerede edges. Derudover kan det ses som et problem at grupperne skal vedligeholdes i kildekoden..
Næste gang	

Opgave	QuadTree søgning
Hvem	Malthe
Hvornår	12.03.2014
Hvad skal der laves	QuadTree klassen skal have en metode som finder den nærmeste edge til et punkt.
Hvordan har vi lavet det	<p>QuadTree</p> <p>Metoden search(double[] point) er implementeret således, at den nærmeste edge til punktet returneres. Metoden benytter en begyndelsesrækkevidde (prædefineret som 20x20 pixels med centrum i det givne punkt), og findes edges inden for denne rækkevidde, gennemløbes disse og den tætteste returneres. Hvis ingen edges findes inden for rækkevidden fordobles denne, og der startes forfra.</p>
Problemer	Da det er svært at teste QuadTree-klassen, da det kræver initialisering af Edge objekter. Derfor er det endnu usikkert om metodens køretid er acceptabel.
Næste gang	Implementer søgningen således at vejen nærmest markøren vises i applikationen.

Opgave	Controller
Hvem	Malthe, Ans og Bjørn
Hvornår	10.03.2014
Hvad skal der laves	Translator klasse og yderligere arbejde på view klasser
Hvordan har vi lavet det	<p>Translator</p> <p>Henter edges fra model og skalerer dem til størrelsen på vinduet, og sender derefter linjer til view. Sørger også for at spejle y-koordinaterne og indstille farven på linjerne.</p> <p>View klasser</p> <p>En række mindre tilpasninger og tilføjelse af en metode der tegner linjerne.</p>
Problemer	Skagen mangler.
Næste gang	Bedre skalering efter bedre akser. Lige nu skaleres der kun efter x-aksen.

Opgave	Færdiggørelse af datamodel
Hvem	Malthe, Ans og Bjørn
Hvornår	05.03.2014
Hvad skal der laves	Datamodellen skal færdiggøres, således view delen af programmet kan tegne kortet ud fra dataet.
Hvordan har vi lavet det	<p>Flere dele af modellens implementation var ringe eller utilstrækkelig, hvorfor flere klasser er blevet omskrevet. Derudover ønskede gruppen at bibeholde krak-kit koden i dens oprindelige form, hvilket den forhenværende implementation ikke opnåede.</p> <p>Edge og Node For at repræsentere krak dataet på en håndterbar måde, skrev gruppen en decorator klasse til EdgeData og NodeData. Disse klasser udbyggede de oprindelige klasser med funktionalitet, som var nødvendigt for at kunne indlæses i quad træet. Blandt andet har en edge i denne implementation en reference til sin egen start- og slutnode.</p> <p>Loader Færdiggjorde klassen, som før hed MappedKrakLoader, således at denne instansierer et QuadTree og populerer det med edges, som kender til deres start- og slutnode.</p> <p>QuadTree I quad træet er der ikke længere gjort brug af den indre klasse Boundary, men i stedet arrays (se Generelt.) Derudover er metoden queryRange, som returnerer samtlige edges indenfor to koordinatsæt, færdigimplementeret og testet.</p> <p>Generelt I modellen er der nu gjort brug en ensartet metode at referere til koordinater på. Der anvendes et dobbeltarray, hvor [0] indeholder x1 og y2, mens [1] indeholder x2 og y2.</p>
Problemer	Køretiden på instansiering af modellen er ganske ringe, men da dette kun sker ved programmets start, kan problemets omfang diskuteres.
Næste gang	Da datamodellen er færdig, kan arbejdet på at translatere data fra modellen til viewet påbegyndes.

Opgave	Implementation af kort-vinduet
Hvem	Bjørn og Mark
Hvornår	03.03.2014
Hvad skal der laves	Vi skal have implementeret en række klasser der tager sig af at lave et vindue og tegne kortet på det.
Hvordan har vi lavet det	<p>Vi har lavet 3 klasser: Window, Canvas, Painter. Derudover har vi lavet en klasse ViewTest til at agere controller og teste koden undervejs.</p> <ul style="list-style-type: none"> - Window er en extended JFrame der er vinduet. - Canvas er en extended JPanel der er tegnepladen som pt. fylder hele vinduet. - Painter tegner på tegnepladens Graphics object.
Problemer	Der er langt mere resourcekrævende at tegne stregerne med antialiasing end uden. Dette gør at vi skal tage en beslutning omkring hvorvidt dette er nødvendigt. Kortet vil blive meget mere letlæseligt, men der vil komme en del mere ventetid, hvor kortet opdaterer.
Næste gang	Et tekstpanel kan tilføjes, der noterer stregernes navne når musen holder over.

Opgave	Implementation af KrakLoader
Hvem	Malthe
Hvornår	03.03.2014
Hvad skal der laves	<p>Implementering af KrakLoaders to abstrakte metoder processNode og processEdge.</p> <p>Udformning af egen datastruktur til effektivt at opbevare og tilgå vejsegmenter indenfor et rektangulært koordinatsæt.</p>
Hvordan har vi lavet det	<p>MappedKrakLoader</p> <p>KrakLoader klassen extendes af klassen MappedKrakLoader. Denne implementerer processNode og processEdge.</p> <p>Når disse metoder kaldes ved indlæsningen af kortdata, tilføjes disse til et quad tree.</p> <p>QuadTree</p> <p>Klasse som opretholder en quad tree datastruktur. Indeholder objekter af typen Locatable. Træets nodekapacitet er på</p>

	<p>nuværende tidspunkt fastsat til 500, men dette bør senere finjusteres.</p> <p>Locatable Interface som muliggører returnering af X- og Y-koordinater med metoderne getX() og getY().</p>
Problemer	<p>Vejsegmenter indeholder ikke umiddelbart et koordinatsæt for deres start- og slutsted. Dette kan dog løses ved at sammenholde et vejsegment med dets start- og slutkryds (node).</p> <p>Et vejsegment bør optræde i samtlige quads som dette beskærer. Dette er ikke let løseligt med de to koordinatsæt, som kan findes for segmentet. En mulig løsning kunne være at betragte vejsegmentets midpunkt som dets quad-koordinat. Implementeres denne løsning må tegnefunktionaliteten omgå quadtræets data med omhu, da et segments godt kan gennemskære en specifik quad, uden at have midtpunkt deri.</p>
Næste gang	<p>Færdiggør quadtræet således at det indeholder samtlige vejsegmenter i krakdataet opdelt i rimelige quads.</p> <p>Implementer metoden queryRange i QuadTree klassen. Denne bør kunne finde samtlige vejsegmenter mellem to koordinatsæt.</p> <p>Skab dataklasser for vejsegmenter (edges) og vejkryds (nodes) som et let at repræsentere grafisk, efter datakilden er indlæst.</p>

11.7 Samarbejdsaftale

Samarbejds aftale

Ambitionsniveau

- Vi vil gerne have 7

Prioriteter

- Bjørn: Ikke mit super-primære fokus lige nu, synes de andre kurser er sværere/vigtigere.
- Mark og Ans: Vi synes det her kursus er vigtigt, og er første prioritet

Preferænecer

- Mark og Ans vil gerne skrive skrive kode, vil gerne blive bedre
- Malthe og Bjørn er okay tilfredse med deres kode niveau

Process/estimeringer

- Aftale om hvilke moduler der skal være færdige til næste uge
- Prøve at ramme en “weekly build” hvis muligt

Konflikthåndtering

- Evalueringer ved hvert møde
 - Samarbejde
 - Produktivitet

Mødetider

- Mandag: Klokken 10
- Onsdag: Klokken 13

Afbud

- Vi starter på klokkeslettet, venter ikke
- Melde afbud så tidligt som muligt
- Send en besked hvis man kommer for sent

Kommunikationsmidler

- Mark laver en mailing list
- Telefon:
 - Mark: 30648630
 - Malthe: 51883445
 - Ans: 30276607
 - Bjørn: 50872800

Versionskontrol

- Mark vil gerne stå for en github repository.
- Vi finder ud af hvordan det virker sammen

11.8 Grene i Prioritetskø

```

public class PriorityQueue<Key extends Comparable<Key>, Value> {
    private Entry<Key, Value>[] a;
    private int n;

    @SuppressWarnings( "unchecked" )
    public PriorityQueue( int initSize ) {
        a = (PriorityEntry<Key, Value>[][]) new PriorityEntry[initSize];
        n = 0;
    }

    public PriorityQueue() {
        this(1);
    }

    @SuppressWarnings( "unchecked" )
    public void push( Key key, Value value ) {
        /*1*/if( key == null ) throw new NullPointerException();
        /*2*/if( n == a.length - 1) resize();
        a[++n] = new PriorityEntry<Key, Value>(key, value);
        swim(n);
    }

    public void push( Key[] keys, Value[] values ) {
        /*3*/if( keys.length != values.length )
            throw new IllegalArgumentException(
                "Mismatch between length of key and value arrays");
        /*4*/for( int i = 0; i < keys.length; i++ ) {
            push( keys[i], values[i] );
        }
    }

    public Entry pop() {
        /*1*/if( isEmpty() )
            throw new NoSuchElementException( "PriorityQueue underflow" );

        Entry ret = a[1];
        exch(1, n--);
        sink(1);

        a[n+1] = null; // gc
    }
}

```

```

/*2*/if(n > 0 && n == (a.length -1)/4) resize();
return ret;
}

public Entry peek() {
/*1*/if(isEmpty()) throw new
    NoSuchElementException("PriorityQueue\underflow");
return a[1];
}

public int size() {
    return n;
}

public boolean isEmpty() {
    return n == 0;
}

private void swim(int i) {
/*1*/while(i > 1 && greater(i/2, i)) {
    exch(i/2, i);
    i = i/2;
}
}

private void sink(int i) {
/*1*/while(i*2 <= n) {
    int j = 2*i;
/*2*/if(j < n && greater(j, j+1)) j++;
/*3*/if(!greater(i, j)) break;
    exch(i, j);
    i = j;
}
}

private boolean greater(int i, int j) {
    return a[i].getKey().compareTo(a[j].getKey()) == 1;
}

@SuppressWarnings("unchecked")
private void exch(int i, int j) {
    PriorityEntry temp = (PriorityEntry) a[i];
    a[i] = a[j];
}

```

```
a[ j ] = temp;  
}  
  
@SuppressWarnings( "unchecked" )  
private void resize() {  
    Entry<Key, Value>[] temp  
        = (PriorityEntry<Key, Value>[]) new PriorityEntry[(n+1)*2];  
    /*1*/for(int i = 0; i <= n; i++) {  
        temp[ i ] = a[ i ];  
    }  
    a = temp;  
}  
  
@Override  
public String toString() {  
    // Will NOT return pop()-order, but heap-order.  
    String ret = super.toString() + ":[";  
    if(!isEmpty()) {  
        for(int i = 1; i <= n; i++) {  
            ret += a[ i ].toString() + ",[";  
        }  
        ret = ret.substring(0, ret.length() - 2);  
    }  
    return ret;  
}  
}
```

11.9 Forventningstabeller

PriorityQueue

Tabel 11.1 – PriorityQueue:Constructor

ID	Input data	Forventet output	Egentligt output
A1	Intet	Intet	Ingen fejl

Tabel 11.2 – PriorityQueue:Push (B1-5 pushes i samme liste med ét metodekald.)

ID	Input data	Forventet output	Egentligt output
B1	(0,1)	size++	size++
B2	(1,2)	size++	size++
B3	(1,3)	size++	size++
B4	(-1,4)	size++	size++
B5	(2,5)	size++	size++
C1	(0, 1, 1, -1, 2), (1, 2, 3, 4)	Fejl	Fejl
C2	null	Fejl	Fejl
C3	Tom liste	Intet	Ingen fejl
C4	Liste med ét element	size++	size++

Tabel 11.3 – PriorityQueue:Pop

ID	Input data	Forventet output	Egentligt output
D1	Efter B1	1	1
D2	Efter B2	2	2
D3	Efter B3	3	3
D4	Efter B4	4	4
D5	Efter B5	5	5
E1	Intet (tom)	Fejl	Fejl
E2	(1,0), pop, pop	Fejl	Fejl

Tabel 11.4 – PriorityQueue:Peek

ID	Input data	Forventet output	Egentligt output
F1	push(1,0), push(-1,0)	(-1)	(-1)
F2	Intet (tom)	Fejl	Fejl

Tabel 11.5 – PriorityQueue:Size

ID	Input data	Forventet output	Egentligt output
G1	Intet (tom)	0	0
G2	push(1,0)	1	1
G3	push(1,0), pop	0	0

Tabel 11.6 – PriorityQueue:isEmpty

ID	Input data	Forventet output	Egentligt output
G1	Intet (tom)	true	true
G2	push(1,0)	false	false
G3	push(1,0), pop	true	true

Tabel 11.7 – PriorityQueue:Push

Gren	Input egenskab	Dækket af test
(1)true	null	C2
(1>false	not null	B1
(2)true	n == a	B1
(2>false	n < a	B2
(3)true	længde af keys != values	C1
(3>false	længde af keys == values	B1-5
(4)Kører ikke	Tom liste	C3
(4)Kørers én gang	Liste med ét element	C4
(4)Kørers flere gang	Liste med flere elementer	B1-5

Tabel 11.8 – PriorityQueue:Pop

Gren	Input egenskab	Dækket af test
(1)true	PQ tom	E1
(1>false	PQ populéret	D1-5
(2)true	n == a/4	B1-5 D1-5
(2>false	n > a/4	D1

Tabel 11.9 – PriorityQueue:Swim

Gren	Input egenskab	Dækket af test
(1)Kører ikke	i < 1 [i/2] < [i]	B1
(1)Kørers én gang	i > 1 && [i/2] > [i], men ikke i/2 > 1 && [i/4] > [i/2]	E1
(1)Kørers flere gange	i/2x > 1 && [i/2x] > [i/2x]	B4

Tabel 11.10 – PriorityQueue:Sink

Gren	Input egenskab	Dækket af test
(1)Kører ikke	i*2>n	D5
(1)Kørers én gang	i*2>n, men ikke i*4>n	D2
(1)Kørers flere gange	i*2*x>n	D1
(2)true	j < n && [j] > [j+1]	D1
(2>false	j > n [j] < [j+1]	D4
(3)true	[i] <= [j]	D1
(3>false	[i] > [j]	D3

Tabel 11.11 – PriorityQueue:Resize

Gren	Input egenskab	Dækket af test
(1)Kører ikke	Ikke muligt	-
(1)Kørers én gang	$n = 0$	B1
(1)Kørers flere gange	$n > 0$	B3

Vector

Tabel 11.12 – Vector:Constructor

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)$	$\text{vec}(1,2)$	$\text{vec}(1,2)$

Tabel 11.13 – Vector:Set

Input data	Forventet output	Egentligt output
$\text{vec}(2,3)$	$\text{vec}(2,3)$	$\text{vec}(2,3)$

Tabel 11.14 – Vector:Copy

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)$	$\text{vec}(1,2)$	$\text{vec}(1,2)$

Tabel 11.15 – Vector:Add

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)+\text{vec}(1,1)$	$\text{vec}(2,3)$	$\text{vec}(2,3)$
$\text{vec}(1,2)+\text{vec}(-1,-1)$	$\text{vec}(0,1)$	$\text{vec}(0,1)$
$\text{vec}(1,2)+\text{vec}(0,0)$	$\text{vec}(1,2)$	$\text{vec}(1,2)$

Tabel 11.16 – Vector:Subtract

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)-\text{vec}(1,1)$	$\text{vec}(0,1)$	$\text{vec}(0,1)$
$\text{vec}(1,2)-\text{vec}(-1,-1)$	$\text{vec}(-1,-1)$	$\text{vec}(-1,-1)$
$\text{vec}(1,2)-\text{vec}(0,0)$	$\text{vec}(1,2)$	$\text{vec}(1,2)$

Tabel 11.17 – Vector:Multiplication

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)*2$	$\text{vec}(2,4)$	$\text{vec}(2,4)$
$\text{vec}(1,2)*0.5$	$\text{vec}(0.5,1)$	$\text{vec}(0.5,1)$
$\text{vec}(1,2)*1$	$\text{vec}(1,2)$	$\text{vec}(1,2)$

Tabel 11.18 – Vector:Division

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)/0.5$	$\text{vec}(2,4)$	$\text{vec}(2,4)$
$\text{vec}(1,2)/2$	$\text{vec}(0.5,1)$	$\text{vec}(0.5,1)$
$\text{vec}(1,2)/1$	$\text{vec}(1,2)$	$\text{vec}(1,2)$

Tabel 11.19 – Vector:Distance

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)$ to $\text{vec}(2,3)$	$\sqrt{2}$	$\sqrt{2}$
$\text{vec}(1,2)$ to $\text{vec}(0,1)$	$\sqrt{2}$	$\sqrt{2}$
$\text{vec}(1,2)$ to $\text{vec}(1,2)$	0	0

Tabel 11.20 – Vector:Translate

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)$ from box(0,0;2,4) to box(0,0;4,8)	$\text{vec}(2,4)$	$\text{vec}(2,4)$

Tabel 11.21 – Vector:isInside (i box(0,0; 1,1))

Input data	Forventet output	Egentligt output
$\text{vec}(0,0)$	true	true
$\text{vec}(0.5,0.5)$	true	true
$\text{vec}(1,1)$	true	true
$\text{vec}(0,1.01)$	false	false
$\text{vec}(1.01,0)$	false	false
$\text{vec}(1.01,1.01)$	false	false

Tabel 11.22 – Vector:MirrorY

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)$ in box(0,0;1,3)	$\text{vec}(1,1)$	$\text{vec}(1,1)$

Tabel 11.23 – Vector:Abs

Input data	Forventet output	Egentligt output
$\text{vec}(-1,-1)$	$\text{vec}(1,1)$	$\text{vec}(1,1)$

Tabel 11.24 – Vector:Dot

Input data	Forventet output	Egentligt output
$\text{vec}(0,0) \cdot \text{vec}(0,0)$	0	0
$\text{vec}(0,0) \cdot \text{vec}(1,3)$	0	0
$\text{vec}(0,0) \cdot \text{vec}(-2,-5)$	0	0
$\text{vec}(0,0) \cdot \text{vec}(-3,4)$	0	0
$\text{vec}(1,3) \cdot \text{vec}(0,0)$	0	0
$\text{vec}(1,3) \cdot \text{vec}(1,3)$	10	10
$\text{vec}(1,3) \cdot \text{vec}(-2,-5)$	-17	-17
$\text{vec}(1,3) \cdot \text{vec}(-3,4)$	9	9
$\text{vec}(-2,-5) \cdot \text{vec}(0,0)$	0	0
$\text{vec}(-2,-5) \cdot \text{vec}(1,3)$	-17	-17
$\text{vec}(-2,-5) \cdot \text{vec}(-2,-5)$	29	29
$\text{vec}(-2,-5) \cdot \text{vec}(-3,4)$	-14	-14
$\text{vec}(-3,4) \cdot \text{vec}(0,0)$	0	0
$\text{vec}(-3,4) \cdot \text{vec}(1,3)$	9	9
$\text{vec}(-3,4) \cdot \text{vec}(-2,-5)$	-14	-14
$\text{vec}(-3,4) \cdot \text{vec}(-3,4)$	25	25

Tabel 11.25 – Vector:Cross

Input data	Forventet output	Egentligt output
$\text{vec}(0,0) \times \text{vec}(0,0)$	0	0
$\text{vec}(0,0) \times \text{vec}(1,3)$	0	0
$\text{vec}(0,0) \times \text{vec}(-2,-5)$	0	0
$\text{vec}(0,0) \times \text{vec}(-3,4)$	0	0
$\text{vec}(1,3) \times \text{vec}(0,0)$	0	0
$\text{vec}(1,3) \times \text{vec}(1,3)$	0	0
$\text{vec}(1,3) \times \text{vec}(-2,-5)$	1	1
$\text{vec}(1,3) \times \text{vec}(-3,4)$	13	13
$\text{vec}(-2,-5) \times \text{vec}(0,0)$	0	0
$\text{vec}(-2,-5) \times \text{vec}(1,3)$	-1	-1
$\text{vec}(-2,-5) \times \text{vec}(-2,-5)$	0	0
$\text{vec}(-2,-5) \times \text{vec}(-3,4)$	-23	-23
$\text{vec}(-3,4) \times \text{vec}(0,0)$	0	0
$\text{vec}(-3,4) \times \text{vec}(1,3)$	-13	-13
$\text{vec}(-3,4) \times \text{vec}(-2,-5)$	23	23
$\text{vec}(-3,4) \times \text{vec}(-3,4)$	0	0

Tabel 11.26 – Vector:Mag

Input data	Forventet output	Egentligt output
$\text{vec}(0,0)$	0	0
$\text{vec}(1,3)$	$\sqrt{5}$	$\sqrt{5}$
$\text{vec}(-2,-5)$	$\sqrt{5}$	$\sqrt{5}$
$\text{vec}(-3,4)$	$\sqrt{5}$	$\sqrt{5}$

Tabel 11.27 – Vector:Norm

Input data	Forventet output	Egentligt output
$\text{vec}(10,0)$	$\text{vec}(0,1)$	$\text{vec}(0,1)$

Tabel 11.28 – Vector:AngleBetween

Input data	Forventet output	Egentligt output
$\text{vec}(1,2) \angle \text{vec}(1,2)$	0	0
$\text{vec}(1,2) \angle \text{vec}(2,-1)$	$-\pi/2$	$-\pi/2$
$\text{vec}(1,2) \angle \text{vec}(-1,-2)$	π	π
$\text{vec}(1,2) \angle \text{vec}(-2,1)$	$\pi/2$	$\pi/2$

Tabel 11.29 – Vector:Equals

Input data	Forventet output	Egentligt output
$\text{vec}(7,21)$ og $\text{vec}(7,21)$	true	true
$\text{vec}(7,21)$ og $\text{vec}(21,7)$	false	false
$\text{vec}(7,21)$ og $\text{vec}(0,-90)$	false	false

Box**Tabel 11.30** – Box:Constructor

Input data	Forventet output	Egentligt output
$\text{vec}(1,2)$ og $\text{vec}(3,4)$	$\text{box}(1,2;3,4)$	$\text{box}(1,2;3,4)$

Tabel 11.31 – Box:Dimensions

Input data	Forventet output	Egentligt output
$\text{box}(1,2;3,4)$	$\text{vec}(2,2)$	$\text{vec}(2,2)$

Tabel 11.32 – Box:RelativeToAbsolute

Input data	Forventet output	Egentligt output
$\text{vec}(0.5,0.5)$ i $\text{box}(1,2;3,4)$	$\text{vec}(2,3)$	$\text{vec}(2,3)$

Tabel 11.33 – Box:AbsoluteToRelative

Input data	Forventet output	Egentligt output
$\text{vec}(2,3)$ i $\text{box}(1,2;3,4)$	$\text{vec}(0,5)$	$\text{vec}(0,5)$

Tabel 11.34 – Box:Ratio

Input data	Forventet output	Egentligt output
$\text{box}(1,2;3,4)$	$\text{vec}(1,1)$	$\text{vec}(1,1)$
$\text{box}(0,0;2,1)$	$\text{vec}(1,0,5)$	$\text{vec}(1,0,5)$
$\text{box}(0,0;1,2)$	$\text{vec}(0,5,1)$	$\text{vec}(0,5,1)$

Tabel 11.35 – Box:Overlapping

Input data	Forventet output	Egentligt output
$\text{box}(0,5,1,5; 1,5;2,5)$ og $\text{box}(1,2;3,4)$	true	true
$\text{box}(0,5,1,5; 0,99;1,99)$ og $\text{box}(1,2;3,4)$	false	false

Tabel 11.36 – Box:OverlappingLine (i box(1,4;3,2))

Input data	Forventet output	Egentligt output
vec(2,5)→vec(2,3)	true	true
vec(4,3)→vec(2,3)	true	true
vec(2,1)→vec(2,3)	true	true
vec(0,3)→vec(2,3)	true	true
vec(2,3)→vec(2,3)	true	true
vec(0,5)→vec(1,4)	true	true
vec(0,2)→vec(4,4)	true	true
vec(3,1)→vec(1,5)	true	true
vec(1,1)→vec(1,5)	true	true
vec(2,1)→vec(4,3)	true	true
vec(3,1)→vec(4,2)	false	false
vec(4,1)→vec(4,5)	false	false
vec(4,2)→vec(4,4)	false	false
vec(4,3)→vec(4,3)	false	false

Tabel 11.37 – Box:getCenter

Input data	Forventet output	Egentligt output
box(1,2;3,4)	vec(2,3)	vec(2,3)

Tabel 11.38 – Box:Scale

Input data	Forventet output	Egentligt output
box(1,2;3,4)*2	box(0,1;4,5)	box(0,1;4,5)

Tabel 11.39 – Box:ProperCorners

Input data	Forventet output	Egentligt output
box(51,-10;0,80)	box(0,-10;51,80)	box(0,-10;51,80)

Tabel 11.40 – Box:FlipX

Input data	Forventet output	Egentligt output
box(11,0;-3,0)	box(-3,0;11,0)	box(-3,0;11,0)

Tabel 11.41 – Box:FlipY

Input data	Forventet output	Egentligt output
box(0,11;0,-3)	box(0,-3;0,11)	box(0,-3;0,11)

Tabel 11.42 – Box:Grow

Input data	Forventet output	Egentligt output
box(1,2;3,4) grow(1)	box(0,1; 4,5)	box(0,1; 4,5)

Tabel 11.43 – Box:isInside (i box(1,2;3,4))

Input data	Forventet output	Egentligt output
box(0.5,1.5; 3.5,4.5)	true	true
box(1.5,2.5; 2.5,3.5)	false	false