

# GIB2 - Stikkord

Alexander Nossum  
I-IKT  
alexanno@stud.ntnu.no  
41293632

19. desember 2006

## Innhold

<b>1</b>	<b>Kartografi</b>	<b>2</b>
<b>2</b>	<b>Datastrukturer og Algoritmer</b>	<b>2</b>
2.1	Kvadtrær og trær generelt . . . . .	2
2.2	Hashstrukturer . . . . .	4
2.3	Representasjon . . . . .	4
<b>3</b>	<b>WWW</b>	<b>4</b>
3.1	Dynamiske variabler . . . . .	4

## 1 Kartografi

**Metrisk rom** er et “vanlig” rom der lengder er like og fast definerte. Må oppfylle kravene:

$$\text{Avstand } a \rightarrow b = b \rightarrow a$$

$$\text{Avstand } a \rightarrow t \rightarrow b \geq a \rightarrow b$$

$$\text{Avstand } a \rightarrow b = 0 \text{ hvis } a = b. \text{ Ellers } \geq 0.$$

**Ikke-metrisk rom** er det motsatte av metrisk rom og omfatter ofte tid. For eksempel kan “avstanden” (tiden) mellom  $a \rightarrow b \leq a \rightarrow t \rightarrow b$ . Hvis man tar tog istedenfor å gå ...

**Presisjon** - oppløsning i måldomenet. Hvor presist har vi gjengitt kildedomenet.

**Nøyaktighet** - Hvor bra måldomenet er *tilpasset* kildedomenet.

## 2 Datastrukturer og Algoritmer

### 2.1 Kvadtrær og trær generelt

**Lineært kvadtre** er når vi kan representere hele treet i et *lineært* array. Dette forutsetter at data er lagret i løvnoder (på nederste nivå?) og som regel sortert ved en eller annen logikk.

**Mortonindeksering** er en type *bitfletting* som bruker *4-tallssystemet* til representasjon. Hensikten er å representere to verdier med en verdi. Eksempelvis. Vi har 010 og 101, ved mortonindeksering velger vi annehver verdi og fletter sammen til 011001. Mer matematisk kan vi kalle verdiene for I og J. Utvalget blir da  $I[i]J[i]I[i+1]J[i+1]$  osv.. Mortonindeksen sier noe om hvor punktet ligger, men *ikke* i hvilken “dybde”.

**Posisjonskode** er mortonindeksen *og* en *nivåkode*.

**Peanokurve** er en kurve som viser sammenhengen i en mortonindeksering. Den er som speilvendt  $N$ . Kan representere en matrise lineært.

**Regulær firedeling** er matematisk balansert oppdeling av planet. Splitter i halvparten av sidekanten for hver gang  $\rightarrow$  kvadrater.

**Irregulær firedeling** velger vi selv oppsplittingspunktet. Kan selv velge hvilke områder vi vil konsentrere oss om.  $\rightarrow$  rektangler.

**Punktkvadtre** bruker firedeling av planet. Splitter i punkter. Kan balanse-res ved bruk av Quicksort i planet. Punkter lagres i nodene (*inter- og løvnoder*).

**PR-tre** bruker firedeling av planet. Lagrer punkter i *løvnodene*. Splitter ikke nødvendigvis i punkter men deler opp til *kun ett punkt pr region* som så lagres i noden (løvnoden). Søk for å finne nærmeste nabopunkt til et tilfeldig punkt baserer seg på *søkesirkel*. Søkesirkelen defineres først ut i fra *naboregioner* (noder) - tar vekk alle punkter utenfor sirkelen - finner forslag til nærmeste punkt - er det et nærmere punkt innenfor søkesirkelen? Gjør sirkelen mindre fra det nærmeste punktet.

**PM(R)-tre** bruker firedeling av planet. Lagrer *informasjon* i noder. En node inneholder *forskjellige felt (attributter)*. Inneholder *ikke topologi*.

**KD-tre** er et binærtre. Splitter i x- og y-retning *annehver gang i ett punkt*. Splittpunktet bestemmer vi selv, median er å foretrekke - tilpasset tyngdepunkt går også. Data lagres i *alle noder* - punktet det splittes i lagres. Trenger ikke informasjon om splittakse.

**Adaptivt KD-tre** er som et KD-tre men vi kan *velge* hvilken akse vi vil *splitte* på for hver gang. Data lagres i alle noder *med informasjon om splittakse*.

**Strip-tre** bruker omskrivende rektangler som oppsplitting. Rektangelet de-fineres ut i fra et sett regler. Dette gjøres til en viss grense er oppnådd.

**MX(-cif)-tre** lagrer blokker som representerer punkt. Forskjellig oppdeling → forskjellig oppløsning. Fin oppdeling der punktene varierer mye - liten oppdeling der punktene er like.

**$R^+$ -tre** bruker omskrivende rektangler. Som B-tre - data lagres i løvnoder på samme og nederste nivå. Intern-noder bygges opp av rektangler som omskriver flere rektangler (noder). Løvnodene er *minste omskrivende rektangel* til et objekt.  $R^+$ -tre har tilleggsregel som sier: *ingen overlappende rektangler*. Dette medfører at et objekt kan bli representert i flere forskjellige løvnoder.

**Okt-tre** er en videreføring av kvadtre. Splitter i *åtte deler*. Godt egnet til representasjon av kubepunkter → kan brukes i *3-dimensjonale rom*.

## 2.2 Hashstrukturer

**Cellemetoden** deler planet i celler fra 1-n. Plasserer punktene i disse cellene. Celleverdien kommer fra en matematisk kompleks hashfunksjon. Ved endring i antall celler må vi *reindeksere*.

**EXCELL** deler planet inn i *celler* med nummerering 1-n. Har en matematisk kompleks hashfunksjon som sørger for at selv om vi deler opp planet i mindre celler blir punktene plassert i de samme cellene som før. Oppdelingen av planet skifter mellom horisontal og vertikal deling.

**GRIDFILE** deler egenskaper med EXCELL men *deler ulikt*. Hashfunksjonen blir betydelig mer matematisk kompleks. Må vite skalaen for delingen vi står på.

## 2.3 Representasjon

**Spagettidata** er når kantene er representert distinkt uten tilknytning til hverandre. Fks. representasjon mellom noder: (2,3),(4,5),(3,4)

**Dobbellenket kantliste (DCEL)** lagrer kanter og sammenhenger mellom de. Data som lagres er: *startpunkt, endepunkt, neste kant, forrige kant, venstre areal og høyre areal*. Lett å finne linjer rundt areal.

**Winged-edge** er som DCEL men lagrer *flere nabokanter*. Lagrer neste-kant, *n, med klokka* og forrige kant, *p, med klokka*.

## 3 WWW

**WMS** står for Web Map Service og gir ut et *ferdig kart* ut ifra brukerens input.

**WFS** står for Web Feature Service og er mer generelt enn WMS. Gir ut data på vektorform og kan støtte transaksjoner.

### 3.1 Dynamiske variabler

**Frekvens** Frekvensen forteller oss hvor fort et objekt forandrer seg pr. tidsenhet

**Synkronisering** Sier noe om hvordan objekter sammenfaller. (rødt-gult  $\rightarrow$  gult-rødt VS. rødt-rødt  $\rightarrow$  gult-gult)

**Varighet** Hvor lenge vises objektet.

**Endringsgrad** I hvor stor grad endres objektet

**Visningstidspunkt** Når vises objektet på tidslinjen.

**Rekkefølge** I hvilken rekkefølge kommer objektene i forhold til hverandre.