Föreläsning 8

Prioritetsköer, och grafer

Innehåll

- Prioritetsköer
  - Modell, organisation, konstruktioner, implementationer
- Grafer
  - Modell, organisation, konstruktioner, implementationer

Prioritetskö

- **Modell**: Patienterna på en akutmottagning, man kommer in i en viss tidsordning men behandlas utifrån en annan ordning.
- **Organisation:** En mängd vars grundmängd är linjärt ordnad av en *prioritetsordning*.
  - Avläsningar och borttagningar görs endast på de element som har högst prioritet.
  - Andra mängdoperationer är inte aktuella

Specifikation av prioritetskö

- Gränsytan Pqueue(val, R):

Empty	() → Pqueue(val, R)
Insert	(v:val, p:Pqueue(val, R)) → Pqueue(val, R)
Isempty	(p:Pqueue(val, R)) → Bool
Inspect-first	(p:Pqueue(val, R)) → val
Delete-first	(p:Pqueue(val, R)) → Pqueue(val, R)

- R är relationen för prioritetsordningen.
- Ibland slås de två sista metoderna ihop.
- Ytan ovan förutsätter *statisk* prioritet. Vill man ha *dynamisk* prioritet måste en update-metod finnas.

Specifikation av prioritetskö

- Man kan också tänka sig att prioritetskön tar element (val) som består av ett värde och en prioritet.
  - Förra fallet antog man att värde = prioritet
- Gränsytan kan varieras på flera sätt
  - update har vi redan nämnt
  - Vi kan också vilja finna det högsta/minsta prioritetsvärdet i kön

Stack och kö är specialfall av prioritetskön!

- Om R är den totala relationen, dvs gäller för alla par av värden blir prioritetskön en stack.
- Om R är den tomma relationen, dvs inte gäller för några par av värden, blir det en kö.
- Dessutom:
  - Om R är en strikt partiell ordning, som ”>”, kommer lika element behandlas som en kö.
  - Om R är icke-strikt, som ”≥” behandlas lika element som en stack.

Konstruktioner av Prioritetskö

- Man utgår ofta från konstruktioner av Mängd, Lexikon eller Heap
  - Men de har vi inte stött på än…
- Lista, ej sorterad
  - Insert O(1), Delete-first O(n)
- Lista, sorterad
  - Insert O(n), Delete-first O(1)

Tillämpningar

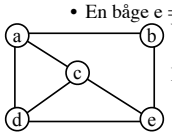
- Operativsystem som fördelar jobb mellan olika processer
- Enkelt sätt att sortera något.
  - Stoppa in allt i en prioritetskö och plocka ut det igen.
- Hjälpmedel vid traversering av graf
  - Jmfr att stack och kö används vid traversering av träd.

Graf

- Modell:
  - Vägkarta med enkelriktade gator utritade.
- Tillämpningar
  - Signaturdiagrammen
  - Elektroniska kretsar
  - Nätverk (gator, flygrutter, kommunikation)
  - Neurala nätverk
  - …

Mängorienterad specifikation (vanlig inom matematiken)

- En graf G = (V, E) består av
  - V : en mängd av *noder* (vertices)
  - E : en mängd av *bågar* (edges) som binder samman noderna i V.
    - En båge e = (u, v) är ett par av noder.



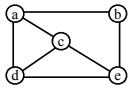
V = {a, b, c, d, e}

E = {(a,b), (a,c), (a,d), (b,e), (c,d), (c,e), (d,e)}

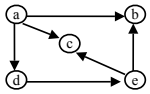
Navigeringsorienterad specifikation

- En graf är en mängd med *noder* där man till varje nod associerar en *grannskapsmängd* av noder som kallas *grannar*.
  - Alla noder tillhör samma typ
  - Alla ordnade par av en godtycklig nod och en av noderna i dess grannskapsmängd utgör en *båge*.
- Denna specifikation passar ofta bättre i algoritmer eftersom de förutsätter effektiva navigeringsoperationer.

Riktade/oriktade grafer

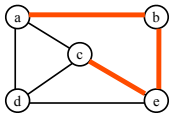


- Oriktade grafer
  - Bågen är en mängd av två noder. Noderna är grannar till varandra.
  - Gradtalet = Antalet bågar till grannar (eller sig själv)
- Riktade grafer
  - Bågen är ordnade par av noder.
  - Gradtalet indelas i
    - Ingradtalet = antalet bågar som går till noden
    - Utgradtalet = antalet bågar som startar i noden och går till en annan nod.



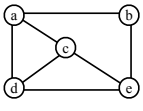
Tänkbar informell specifikation:

Empty	– konstruerar en tom graf utan noder och bågar
Insert-node(v, g)	– sätter in noden v i grafen g
Insert-edge(e, g)	– sätter in en båge e i grafen g. Det förutsätts att noderna finns i grafen
Isempty(g)	– testar om grafen g är tom, dvs utan noder
Has-no-edges(g)	– testar om grafen g saknar bågar
Choose-node(g)	– väljer ut en nod ur grafen g
Neighbours(v, g)	– mängden av alla grannar till v i grafen g
Delete-node(v, g)	– tar bort noden v ur grafen g, förutsatt att v inte ingår i någon båge
Delete-edge(e, g)	– tar bort bågen e ur grafen g

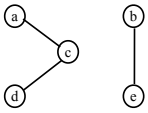


Terminologi

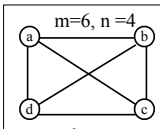
- *Väg*/*stig* (path): En sekvens av noder $v_1, v_2, \ldots, v_n$ så att v_i och v_{i+1} är grannar.
  - I figuren ovan, a, b, e, c
- *Enkel väg* (simple path): Inga noder förekommer två gånger i vägen.
- *Cykel* (cycle): En enkel väg där den sista noden i sekvensen är densamma som den första.



Terminologi

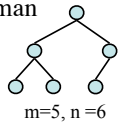


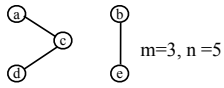
- Sammanhängande (connected) graf
  - Varje nod har en väg till varje annan nod.
- Delgraf (subgraf) en delmängd av noderna och kanterna som formar en graf.
- Sammanhängande komponenter
  - En sammanhängande subgraf
  - Grafen uppe till höger har två sammanhängande komponenter.



Connectivity

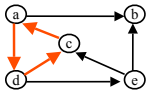
- Låt *n* = antalet noder och *m* = antalet bågar.
- En komplett graf (complete graph) får man när alla noder är grannar till alla andra.
  - I en komplett oriktad graf är m = n(n-1)/2
  - För ett träd gäller m = n-1
  - Om m < n-1 så är grafen inte sammanhängande

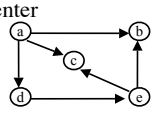




Digraph och DAGs

- DiGraph = Directed graph
  - dvs riktad graf
  - kan vara sammanhängande (dvs vägar finns från alla till alla noder)
  - kan ha sammanhängande komponenter
- DAG = Directed Acyclic Graf
  - dvs, en riktad graf utan cykler

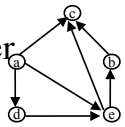




Mer grafer…

- Viktad graf
  - En graf där bågarna har vikter
- Multigraf
  - Tillåtet med flera bågar mellan två noder. Dessa bågar har då olika egenskaper som måste lagras.
- Ordnad graf har inbördes ordning mellan grannarna till en nod.

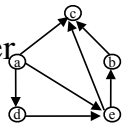
Konstruktion av grafer



- Förbindelsematris
  - Bågarna representeras av ettor i en matris.
  - Rad 1 visar vilka bågar man kan nå från a.
  - Kolumn 3 visar från vilka noder det kommer bågar till c.
  - \+ Enkel att implementera och passar när man har siffror på noder och bågar.
  - \- Matrisen kan bli stor och gles och kräva specialtrick.

	1	2	3	4	5
1			1	1	1
2			1		
3					
4					1
5		1	1		

Konstruktion av grafer



- Graf som fält av lista.
  - Listan är grannskapslistan.
  - Man utgår att det finns minst en båge från varje nod (Fält) men inte att går en båge från varje nod till varje annan nod (därför Lista).
  - \+ Inte lika utrymmeskrävande som en gles matris.
    Utrymmet = O(n+m)
  - \- Fixt antal noder