

# Heap / Prioritetskø:

- Se igjen fra forelesning 1a: **AbstrakteDatatyper.pdf** (*aller* nederst om Heap)
- **En heap:**
  - ligner noe på stakk/kø.
  - er grei å bruke når stadig skal operere på det største/(minste) elementet.
  - *er en array ordnet etter visse (heap)betingelser.*
  - **Poenget:** *Delvis* sortert, raskt å få tak i det neste og største/(minste) elementet
- **Prinsippene for en heap (hvordan man bygger/lager/vedlikeholder den):**
  1. Dersom alle verdiene legges inn i et binært tre, så skal treet tilfredsstillende **heap-betingelsen:** *Enhver nodes verdi/key/ID er større (eller lik) verdien i barna* (om den har noen). Dette medfører at den største verdien til enhver tid er/ligger i treet's rot.
  2. Treet som bygges er et *komplett tre* (se definisjon her: **TrerGrunnleggende.pdf** )
  3. Vi traverserer/leser dette treet *level-order*, og legger verdiene *fortløpende inn i en array*. Dermed har vi fått en array som tilfredsstillende heap-betingelsen! Og den største verdien ligger først i arrayen (med indeks nr.1. I element nr.0 (null) ligger det stort sett en *sentinel key*).
  4. Foreldren/mora til element nr.  $j$ , ligger i element nr.  $j/2$  (oddtall avrundes nedover) Og evt. barnene til element nr.  $i$ , ligger i element nr.  $i*2$  og  $(i*2)+1$  !
- **Algoritmer som opererer på heap:**

(Disse endrer noe på strukturen, slik at heap-betingelse brytes midlertidig, men de omordner slik at den raskt stemmer igjen.)

  - **insert:** Plasserer nytt element bakerst. Bytter m/foreldre til ikke større lengre.
  - **remove:** Tar vare på, og til slutt returnerer nåværende element nr.1. Flytter den aller bakerste inn som nytt element nr.1. Bytter nedover med største barn inntil ikke mindre lengre.
  - **replace:** Bytter ut den største verdien (i element nr.1) med en annen, og returnerer den nå største verdien. **NB:** Det *kan* hende at den nye verdien faktisk er den største, og da er det denne som skal returneres! Dette ivaretas ved også å bruke element nr.0 (null) – se kode på **EKS\_25\_Heap**).
  - **change:** Se oppgave nr.15
  - **delete/extract:** Se oppgave nr.15

*Heap-algortimene kan brukes til sortering vha. insert alle, og deretter remove alle (får da tak i alle elementene i fallende/stigende rekkefølge).*