Oppgave 1a

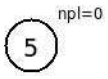
Tegn treet vi får ved sette inn følgende elementer i en vanlig, tom venstre ventrevridd heap; 5, 3, 4, 1, 2 (altså en hvor vi snur om på barna på vanlig måte). Tegn så opp alle mellomstadiene. Som vanlig betyr lav key høy prioritet.

Svar

Vil bruke nodeP[0-5] for å angi nodene som blir lagd.

Steg 1

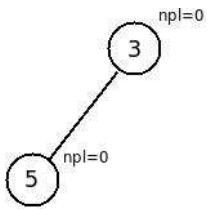
Lager en ny tom venstrevridd heap og setter inn en node med prioritet 5. Etter den er satt inn i treet har rootnoden nullsti lengde på 0.



Etter vi har lagd den nodeP5

Steg 2

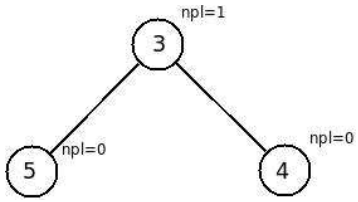
Lager som over en tom venstrecridd heap og setter inn en node med proioritet 3. Etter dette tar man en merge på de to lagde venstrevridde heapene. Siden nodeP3 ikke har noen vestre barn setter vi venstre pekeren til å peke på nodeP5.



Etter vi har satt in nodeP3

Steg 3

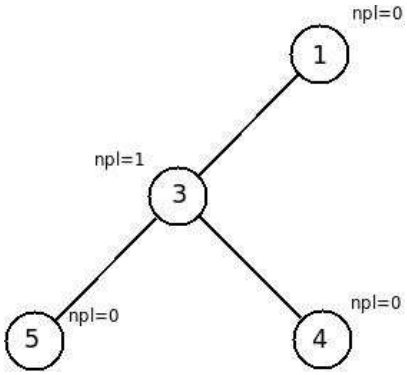
Lager som over en tom venstrecridd heap og setter inn en node med proioritet 4. Siden nodeP3 har en venstre peker setter vi inn nodeP4 som høyre peker. Når dette er gjort oppdaterer vi nodeP3 sin nullsti lengde til 1. Siden nodeP3 sitt venstre barn ikke har minde nullsti lengde en høyrebarnet blir det ikke gjort bytte av pekerene til nodeP3



Etter vi har satt inn nodeP4

Steg 4

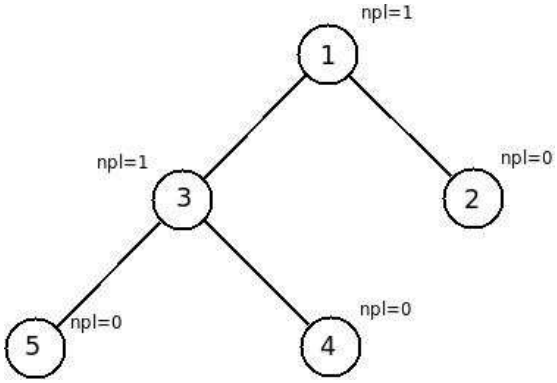
Lager som over en tom venstrecridd heap og setter inn en node med proioritet 1. Siden denne noden har lavere prioritet og ingen venstre peker setter vi nodeP1 sin venstre peker til å peke på node3.



Etter vi har inn nodeP1

Steg 5

Lager som over en tom venstrecridd heap og setter inn en node med proioritet 2. Siden nodeP1 har lavere prioritet og ingen høyre peker blir denne satt til å være nodeP2. Siden nodeP1 sitt venstre barn ikke har lavere nullsti verdi en nodeP1 sitt høyre baren blir det heller ikke her gjort noe bytte av pekerene til nodeP1.



Etter insert av node med prioritet 2

Oppgave 1b

Finnes det en rekkefølge (m.h.t. prioritet) slik at om vi setter elementer inn i denne rekkefølgen i en vanlig, tom ventrevridd heap (uten å ta ut noe), så vil vi med jevne mellomrom (når antallet er $2^n - 1$) få et fult balansert tre? Forklar.

Svar

Det finnes en slik rekkefølge som gjør at vi kan få et fult balansert tre. Hvis man har en tall sekvens som alltid er økende vil man hele tiden fylle opp hver rad før man går videre til neste rad (prioriteten til noden man setter inn er alltid lavere en prioriteten til den neste). Ved å ha en slik tall sekvens vil få et fult balansert tre når vi har $2^n - 1$ noder i treet (hvor n er antall noder i treet). Eksemel på en slik tall sekvens vil være {1,2,3,4,5,6,7}.

Grunnen til at dette skjer er at vi hele tiden fyller up barnene til noden som ligger helt til høyre. Når denne noden er fylt blir den byttet ut ved hjelp av nullsti lengdene som blir generert. Så på denne måten vil man alltid fylle ut et helt nivå før man får videre til å legge til noder i et nivå som er høyre. På denne måten vil man gå et balansert tre når det er $2^n - 1$ noder.

Oppgave 1c

Finnes det en (annen) rekkefølge som, igjen for en vanlig venstrevridd heap, gir os en lang (venstre)sti uten forgreninger? Forklar hvorfor dette er en heldig situasjon for venstrevridde heaper?

Svar

Det finnes en rekkefølge av tall som for vanlig venstrevridde heaper gir en slik sti. I motsetning til i oppgave to hvor rekkefølgen er økende vil her rekkefølgen være synkende (første element har alltid høyerer prioritet en den neste i rekkefølgen). Eksempelvis vil en slik rekkefølge være {5, 4, 3, 2, 1}

Denne typen situasjon vil være veldig heldig for venstrevridde heaper. Grunnen til dette er at når vi skal sette inn en ny node, kan vi alltid sette det allerede eksisterende treet til å være ventrepekeren til den nye noden. Dette medfører at man bare vil trenge å kalle merge en gang for å sette inn en ny node.

Det samme gjelder når vi skal hente ut data fra heapen ved kall på deleteMin. Dette medfører da at vi bare trenger å bytte ut rooten med roten sitt første venstrebarn. Så bare et kall på merge her også.