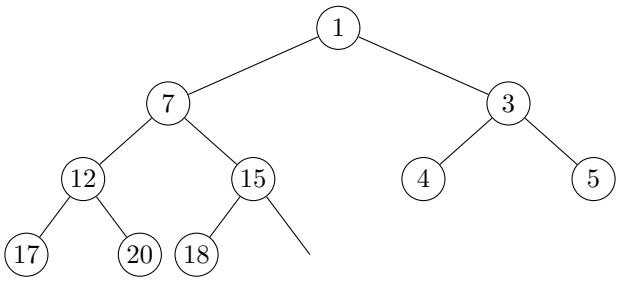
Eksamen 25. mai 2021 **IDATA2302 Algoritmer og Datastrukturer**
Eksamenstid $4\frac{1}{2}$ timar *Nynorsk*

1. Me har to *array*ar A og B som kvar inneheld n heiltal.
   (a) Lag ein effektiv algoritme for å sjekka om der finst eit element e som er innehalde i den eine *array*en og ikkje i den andre (dvs. $e \in A - B$ eller $e \in B - A$). Algoritmen skal ha køyretidskompleksitet $O(n \cdot \log n)$. Forklar kort kvifor algoritmen er korrekt.
   (b) Lag ein annan effektiv algoritme som finn ein *array* C som omfattar m distinkte element som utgjer unionen av A og B. Forklar kort kvifor algoritmen er korrekt.
   (c) Finn køyretidskompleksiteten åt algoritmen.
   (d) Drøft kort køyretidskompleksiteten. Er det råd å løysa problemet asymptotisk raskare, anten i verste fall eller i gjennomsnitt?



Figur 1: A heap for Question 2.

2. Figur 1 viser ei dyngje (*heap*), visualisert som eit binærtre.
   (a) Forklar kva dyngjeeigenskapen er, og vis at datastrukturen i figuren tilfredsstiller denne eigenskapen.
   (b) Forklar kort kvifor dyngjeeigenskapen kan vera nyttig i praktisk bruk, gjerne med døme.
   (c) Gå ut frå at me skal henta ut det minste elementet frå dyngja. Vis, steg for steg, korleis dyngja vert endra for å gjenoppretta dyngjeeigenskapen når det minste elementet vert fjerna. Bruk ein generell og effektiv algoritme.
   (d) Gjev pseudokode for ExtractMin-algoritmen som du brukte i førre oppgåve (inklusive hjelperutinar).
   (e) Kva er køyretidskompleksiteten åt ExtractMin()?

3. Sett at me har ei usortert liste a med n element. Me skal finna det $i$te største elementet. Dersom me sorterer lista i synkande orden fyrst, kan me sjølvsagt henta ut det $i$te elementet relativt enkelt.
   (a). Kva er køyretidskompleksiteten på denne naïve løysinga inklusive sortering? Grunngje svaret kort.
   (b). Der er fleire moglege løysingar som gjev lågare kompleksitet (anten i verste fall eller i gjennomsnitt). Gje pseudo-kode for minst éi og forklar kvifor ho gjev korrekt svar.
   (c). Vurder kompleksiteten på løysinga som du har vald.

4. Gje eit grovdesign til ein søkjemotor basert på søkjetre, dvs. sokalla *tries*. Søkjemotoren vil indeksera sidene på førehand. Når brukaren søkjer på eit ord, skal systemet returnera ei liste med sider sortert etter relevans. Ta med fylgjande:
   (a). Ein kort drøfting av moglege søkjeheuristikkar. Korleis måler me relevansen for ei side?
   (b). Skildring av datastrukturen, gjerne vha. klasse- og objektdiagram.
   (c). Algoritmen for å byggja indeksen, med køyretidskompleksitet og argumentasjon.
   (d). Algoritmen for å finna ei side, køyretidskompleksitet og argumentasjon.