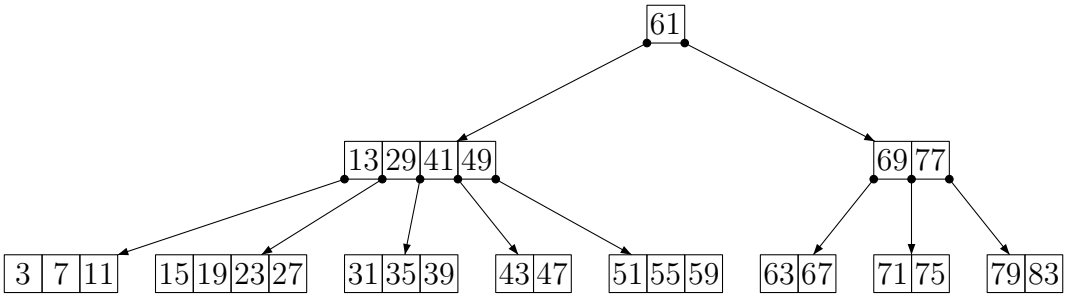
Gevraagde functies en programma's mogen in pseudo-code gegeven worden. Geef steeds voldoende uitleg. Succes.

1. **a)** Wat is een abstracte datastructuur (ADT) ?

b) Beschrijf de ADT *stapel* (=*stack*) en schets twee geschikte implementaties.

2. Beschouw de volgende B-boom B van orde 5:



a) **(i)** Verwijder de sleutel 69 uit B. **(ii)** Voeg vervolgens 69 weer toe.
Zorg dat we steeds een B-boom van orde 5 behouden. Geef uitleg en tussenstappen; delen van de boom die niet veranderen kun je schematisch weergeven.

b) Met één geschikte sleutel erbij kun je zorgen dat de wortel van B twee sleutels krijgt. Hoeveel sleutels erbij zijn er nodig om de boom een nivo hoger te laten krijgen? (Je hoeft niet precies de sleutels te geven: schetsmatig de aantallen sleutels per knoop is genoeg.)

c) We kijken naar B-bomen van orde m. Na het verwijderen van een sleutel uit een knoop wordt die knoop soms samengevoegd met een buur/broer.
(i) Wanneer gebeurt dat?
(ii) Laat zien dat na samenvoegen het aantal sleutels van de resulterende knoop dan altijd weer klopt met de eisen van de B-boom.

3. **a)** Beschrijf *leftist trees*, met hun basisoperatie 'ritsen' (*Zip*).

b) Leg uit hoe leftist trees gebruikt kunnen worden als implementatie van de abstracte datastructuur Priority Queue.

c) **(i)** Bewijs dat een leftist tree waarvan de wortel *nil path length* k heeft, ten minste $2^k - 1$ knopen heeft.
(ii) Wat kun je zeggen over het maximaal aantal knopen in die boom?

4. Het algoritme van Floyd berekent alle de afstanden tussen alle paren knopen in een graag, als die graaf geen cykels met negatief gewicht bevat.

a) Geef het algoritme van Floyd.

We voeren het algoritme uit op een specifieke graaf. De afstanden tussen ieder paar knopen van een graaf zijn achtereenvolgens (onderstreept zijn de waardes die veranderd zijn):

$k = 0$

$$\begin{pmatrix} 0 & 3 & 1 & \infty \\ 3 & 0 & 1 & 4 \\ \infty & 1 & 0 & 6 \\ -2 & 4 & 6 & 0 \end{pmatrix}$$

$k = 1$

$$\begin{pmatrix} 0 & 3 & 1 & \infty \\ 3 & 0 & 1 & 4 \\ \infty & 1 & 0 & 6 \\ -2 & \underline{\mathbf{1}} & \underline{\mathbf{-1}} & 0 \end{pmatrix}$$

$k = 2$

$$\begin{pmatrix} 0 & 3 & 1 & \underline{\mathbf{7}} \\ 3 & 0 & 1 & 4 \\ \underline{\mathbf{4}} & 1 & 0 & \underline{\mathbf{5}} \\ -2 & 1 & -1 & 0 \end{pmatrix}$$

$k = 3$

$$\begin{pmatrix} 0 & \underline{\mathbf{2}} & 1 & \underline{\mathbf{6}} \\ 3 & 0 & 1 & 4 \\ 4 & 1 & 0 & 5 \\ -2 & \underline{\mathbf{0}} & -1 & 0 \end{pmatrix}$$

b) Bepaal de uiteindelijke afstandsmatrix voor de graaf.

c) Welke informatie is nog nodig om de kortste paden zelf te reconstrueren? Bepaal die informatie voor deze graaf.

5. Beschouw hashen in een zgn. 'open' hashtabel met twee hash-functies h en p. Het $i+1$-ste bezochte adres $h(K, i)$ is zoals gewoonlijk $h(K) - i \cdot p(K)$ (modulo M).

a) Welke twee soorten clustering onderscheiden we bij hashen met open adressering? Geef een korte beschrijving.

b) In een tabel `T[0..10]`, dus $M = 11$, worden achtereenvolgens de sleutels 16, 29, 36, 40, 9, 20, 28 geplaatst, met adresfunctie $h(K) = K \bmod 11$ en lineair hashen (stapgrootte 1).

(i) Laat zien welke tabel ontstaat, maar geef op een overzichtelijke manier ook alle plekken waar de sleutels geprobeerd worden.

(ii) Hoeveel stappen kost het om te zien of 18 in T opgeslagen is?

c) Idem, nu met een stapfunctie $p(K) = 1 + (K \bmod 10)$.