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

1) **a.** Geef de definities van een binomial tree en een binomial queue.

b. i. Construeer een binomial queue door, uitgaande van een lege queue, achtereenvolgens de volgende getallen toe te voegen: 85, 20, 79, 80, 92, 40 en 34. Uiteraard dient ook het tussenresultaat na iedere toevoeging een binomial queue te zijn. Geef via tussenresultaten en een korte toelichting duidelijk aan hoe je aan je antwoord komt.

ii. Voer op het resultaat van het vorige onderdeel één maal de operatie DeleteMin (VerwijderMin) uit.
Geef ook hier via tussenresultaten en een korte toelichting duidelijk aan hoe je aan je antwoord komt.

c. Wat zijn de twee belangrijkste (mogelijke) verschillen tussen de vorm van een binomial queue en die van een Fibonacci queue?

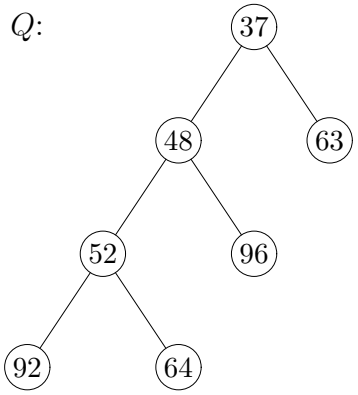
Waardoor ontstaan die verschillen?

d. De rij

Insert (2), Insert (9), Insert (7), DeleteMin ()

is een voorbeeld van een serie operaties, inclusief eventuele argumenten, die je kunt uitvoeren op een Fibonacci queue.

Beschouw nu de volgende Fibonacci queue Q:



Geef een serie operaties, inclusief eventuele argumenten, die uitgaande van een lege queue de Fibonacci queue Q oplevert. Laat ook zien *hoe* je met behulp van die operaties op Q uitkomt, d.w.z. geef tussenresultaten en een korte toelichting.

N.B.: In bovenstaande tekening van Q zijn knopen die al een subboom hebben verloren niet gemarkeerd. Afhankelijk van de serie operaties kunnen er echter wel knopen gemarkeerd zijn. Geef in je eigen tekeningen wél steeds de markeringen aan.

2) **a.** Een Fibonacci boom (niet te verwarren met een Fibonacci queue) van hoogte h is een AVL-boom van hoogte h met zo weinig mogelijk knopen.

i. Geef voor $h = 1, 2, 3, 4, 5$ een Fibonacci boom van hoogte h (een boom bestaande uit één knoop heeft hoogte $h = 1$). Het gaat alleen om de vorm van de boom; het is niet nodig om waardes voor de sleutels te geven.

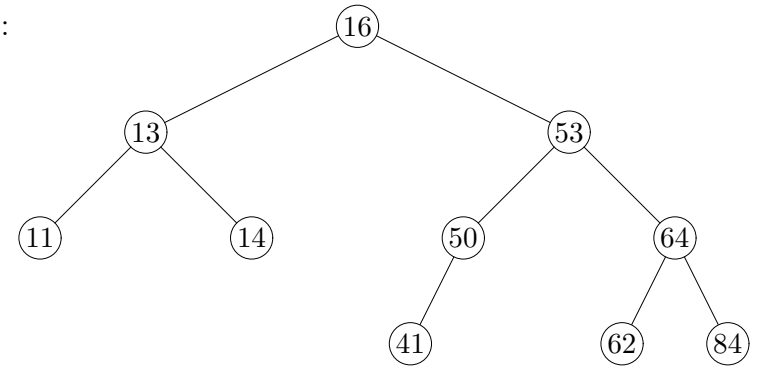
ii. Beredeneer hoeveel niveaus met knopen volledig gevuld zijn in een Fibonacci boom van hoogte h ($h \geq 1$ willekeurig).

iii. Hoeveel knopen bevinden zich op deze volledig gevulde niveaus samen? (Indien je bij onderdeel ii. geen antwoord hebt, gebruik dan als antwoord van dat onderdeel: $f(h)$.)

iv. Leid uit het vorige onderdeel een (grove) bovengrens af voor de hoogte h van een AVL-boom met n knopen.

b. Beschouw de volgende AVL-boom B_1:

B_1:



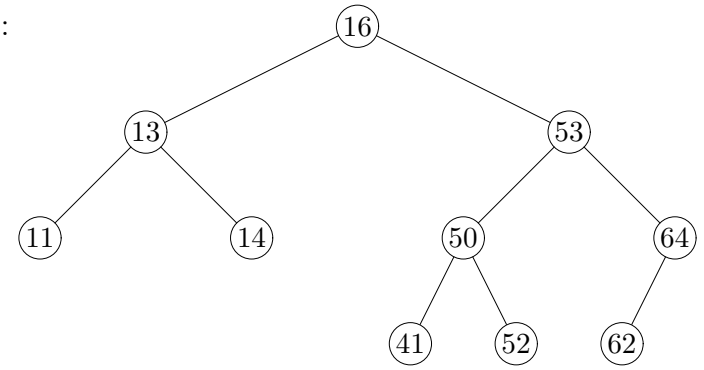
i. Wat is het resultaat na toevoeging van de sleutel 81 aan B_1?

ii. We beginnen opnieuw met de originele boom B_1. Wat is het resultaat na toevoeging van achtereenvolgens de sleutels 30 en 22 aan B_1?

Geef steeds duidelijk aan hoe je aan je antwoord komt.

c. Beschouw de volgende AVL-boom B_2:

B_2:



i. Wat is het resultaat na verwijdering van de sleutel 16 uit B_2?

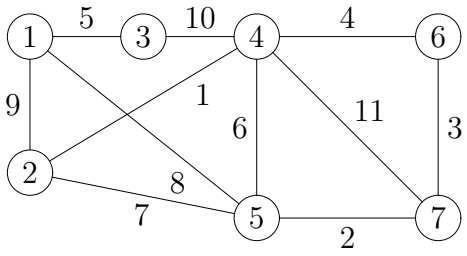
ii. We beginnen opnieuw met de originele boom B_2. Wat is het resultaat na verwijdering van achtereenvolgens de sleutels 11 en 13 uit B_2?

Geef steeds duidelijk aan hoe je aan je antwoord komt.

3) Wanneer volgens het algoritme van Prim een opspannende boom bepaald wordt, groeit deze boom vanuit de gekozen startknoop. Het *algoritme van Kruskal* dat hetzelfde probleem oplost laat geen boom groeien, maar kiest steeds een tak van minimaal gewicht die geen kring vormt met de reeds gekozen takken.

```
gegeven graaf G=(V,E)                                          0
begin met lege boom T                                          1
while T verbindt niet alle knopen                              2
do    kies tak (u,v) uit E met minimaal gewicht                3
      verwijder (u,v) uit E                                    4
      if (u,v) geen kring vormt met takken uit T               5
      then voeg (u,v) aan T toe                                6
      fi                                                       7
od                                                             8
```

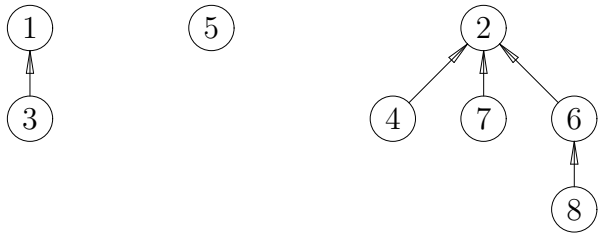
a. Voer het algoritme uit op de volgende graaf:



Regels 5 en 6 van het algoritme zijn cruciaal. Er moet een representatie gebruikt worden om op efficiente wijze kringen te kunnen detecteren.

De datastructuur *union-find* houdt binnen N objecten (zeg de getallen $1, \ldots, N$) een aantal disjuncte deelverzamelingen bij. De operatie union(i, j) voegt de deelverzamelingen waarin i en j zich bevinden samen; de operatie find(i) levert een unieke representant van de verzameling van i.

Bij een mogelijke representatie worden bomen gebruikt, waarbij we letten op de hoogte van de bomen. Bij het samenvoegen van bomen wordt steeds de wortel van de laagste boom als kind aan de wortel van de hoogste boom gekoppeld. (Nogmaals: hierbij is de hoogte van de boom maatgevend, en niet de waarde opgeslagen in de knoop.)
Hieronder tekenen we het resultaaat na union(6,8); union(3,1); union(7,2); union(4,7); union(7,8); voor de getallen 1,...,8 toegepast op de beginsituatie waar elke deelverzameling één element heeft.



b. Werk bovenstaande representatie uit. Geef declaraties en implementeer de functies. (Dit hoeft *niet* als een volledig werkende klasse gepresenteerd te worden.)

c. Voeg de juiste union-find instructies toe aan het gegeven algoritme van Kruskal.

4) De methode van *Knuth-Morris-Pratt* wordt gebruikt om een patroon P[1..m] in een tekst T[1..Lengte] te zoeken. Daartoe worden *failure-links* opgesteld.

a. Stel de failure link bij positie i wijst naar positie k in P: Flink[i] = k. Wat geldt dan voor de woorden P[1]...P[i] en P[1]...P[k] ?
In het bijzonder, wanneer geldt Flink[i] = 0 ?

b. Geef een efficient algoritme om de failure links op te stellen.

c. Bepaal de *failure-links* voor het patroon P = `ABCABABC`.

d. We zoeken naar P in de tekst T = `ABCA BCAB AABC ABAB BABC` (hier staan de spaties voor de leesbaarheid). Geef nauwkeurig aan hoe het zoeken volgens de KMP-methode gebeurt. Welke letters worden telkens met elkaar vergeleken?