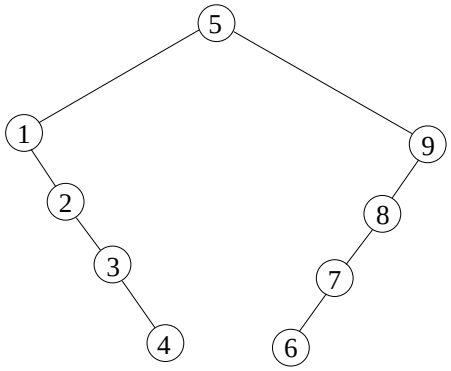
Examen Datastructuren en Algoritmen II

Naam : .....

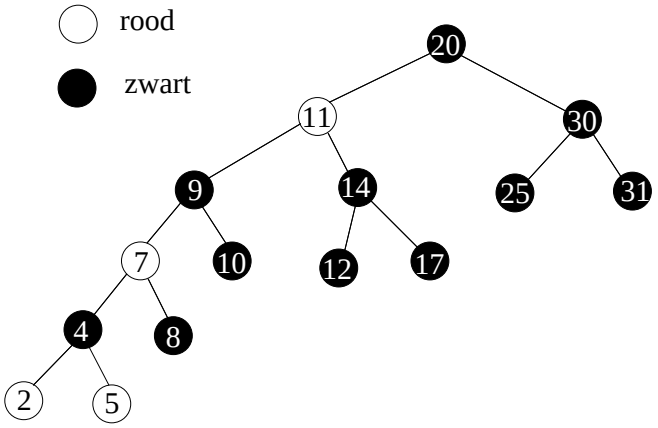
- Lees de hele oefening zorgvuldig voordat je begint ze op te lossen!
  Als je niet goed verstaat wat de vraag of taak is, vraag het aan de lesgever! Voor oefeningen die fout verstaan zijn, kunnen geen punten gegeven worden!
- Schrijf leesbaar. Oplossingen die niet leesbaar zijn, kunnen ook niet beoordeeld worden.
- Als technieken toegepast moeten worden, toon altijd voldoende tussenstappen om te kunnen zien wat er gebeurt en dat de technieken goed verstaan zijn.
- Stellingen uit de les mogen natuurlijk altijd gebruikt worden zonder dat het bewijs opnieuw gegeven moet worden (behalve in gevallen waar het expliciet anders staat)!
- Geef alleen dan een antwoord als je denkt dat je de oplossing kent. Verspil geen tijd met de poging gewoon lange teksten te schrijven waarin zekere sleutelwoorden opduiken – zoals dat vaak geprobeerd wordt. Dergelijke oplossingen halen nooit punten en voor bijzonder slechte oplossingen worden punten afgetrokken!

1. **Zoekbomen 2.5 pt**

   - Verwijder sleutel 5 uit de volgende semi-splay boom. Toon alle tussenstappen, maar je hoeft niet voor elke tussenstap de hele boom te tekenen.



- Voeg sleutel 3 toe aan de volgende rood-zwart boom. Gebruik de operaties die het toelaten soms vroeger met het herbalanceren te stoppen. Toon alle tussenstappen, maar je hoeft niet voor elke tussenstap de hele boom te tekenen.

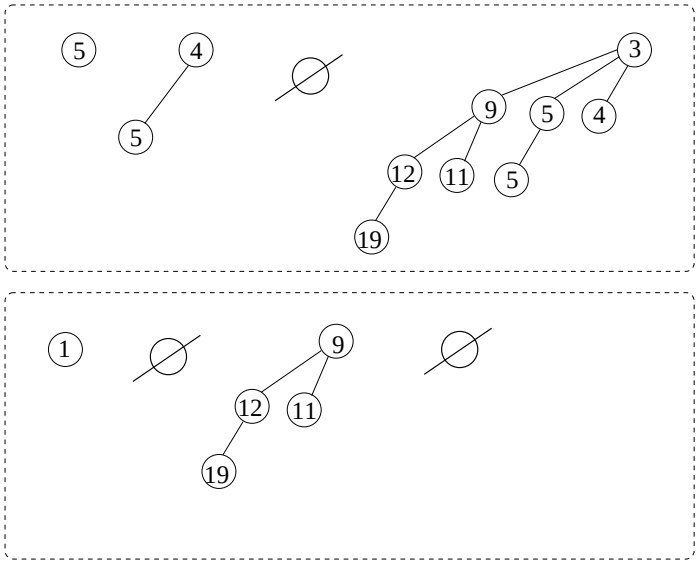


- Gegeven een 2-3 boom waarin één van de kinderbomen van de wortel m sleutels bevat.
  - Geef en bewijs een (goede) bovengrens voor het aantal sleutels in de andere kinderbomen van de wortel als functie van m.

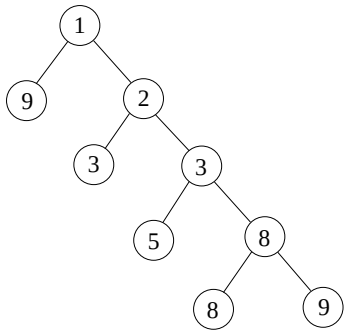
  - Geef en bewijs een (goede) benedengrens voor het aantal sleutels in de andere kinderbomen van de wortel als functie van m.

2. **Heaps 2.5 pt**

- Merge de twee volgende binomiale wachtlijnen.



- Voeg sleutel 4 toe aan de volgende skew heap. Gebruik de niet recursieve manier om heaps te mergen.



- Geef een gewijzigde vorm van de recursieve skew merge bewerking uit de les die als resultaat dezelfde heap oplevert als de niet recursieve skew merge bewerking (die dus eerst de rechterpaden merget) uit de les. Je hoeft geen rekening te houden met het probleem dat voor identieke sleutels de volgorde in de gemergde rechterpaden niet uniek is, maar mag voor deze oefening veronderstellen dat alle sleutels verschillend zijn.

3. **Geamortiseerde complexiteit 2.25 pt**

Je kan een trinomiale wachtlijn analoog aan een binomiale wachtlijn definiëren:

- Een trinomiale boom met hoogte 0 is gewoon een enkele top.
- Een trinomiale boom T met hoogte k bestaat uit 3 trinomiale bomen T_1, T_2, T_3 met hoogte $k-1$. De wortel van T is de wortel van T_1 en de wortels van T_2 en T_3 zijn nieuwe kinderen van de wortel van T_1. De kinderen hebben hier geen bepaalde volgorde.

Een trinomiale wachtlijn is een verzameling W van trinomiale bomen met sleutels in de toppen waarvoor geldt:

- Voor elke k zitten ten hoogste twee bomen met hoogte k in W. De diepte van de grootste aanwezige boom in W noemen we de diepte $d(W)$ van de wachtlijn.
- De sleutels in de kinderen van een top in één van de bomen zijn ten minste even groot als de sleutel in de ouder.

Het mergen van een trinomiale wachtlijn gebeurt gelijkaardig aan een binomiale wachtlijn, alleen dat je nu pas een overdracht hebt als je ten minste drie bomen met dezelfde diepte hebt. Dan worden drie bomen samengevat tot één boom met een grotere diepte. Als de wachtlijnen W, W' gemerged worden en $d(W) > d(W')$ mergen wij W' in W zodat we in sommige gevallen bomen met grote diepte niet hoeven te beschouwen. Als $d(W) = d(W')$ kiezen we om het even welke wachtlijn om de andere erin te mergen.

Een toevoegbewerking op deze wachtlijn is volledig analoog met een toevoegbewerking op een binomiale wachtlijn: je plaatst de nieuwe sleutel in een nieuwe boom met diepte 0 die een eigen wachtlijn vormt en merget deze wachtlijn met de al bestaande.

Als twee wachtlijnen worden gemerged dan rekenen wij het wijzigen van een boom of het plaatsen van een boom in de wachtlijn als kost 1. Als dus bv. 3 bomen uit de twee wachtlijnen één nieuwe boom vormen en die wordt in de nieuwe wachtlijn geplaatst, dan heeft dat kost 4.

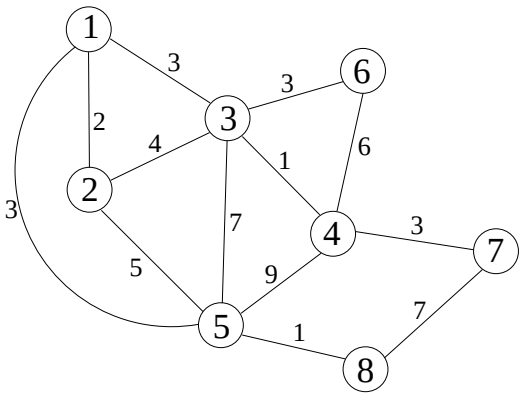
- Wat is de maximale diepte $d(W)$ (exact – geen $O()$-notatie) van een trinomiale wachtlijn W met n sleutels? Geef een afleiding van jouw formule.

- Wat is de maximale kost ($O()$-notatie) van een toevoegbewerking in een reeks van n toevoegbewerkingen op een initieel lege trinomiale wachtlijn? Geef uitleg.

- Wat is de geamortiseerde kost ($O()$-notatie) van een toevoegbewerking in een reeks van n toevoegbewerkingen op een initieel lege trinomiale wachtlijn? Je mag vrij kiezen welke methode je gebruikt om jouw formule af te leiden.

4. **Dynamisch programmeren 1.75 pt**

Gegeven een graaf $G = (V,E)$ met toppenverzameling $V = \{1,2,\ldots,n\}$ en gewichten $w(e) \geq 0$ voor de bogen $e \in E$. Een spel begint met top 1 als actuele top. Er zijn spelers X en O – speler X begint. In elke zet kiest een speler een buur van de actuele top die groter is dan de actuele top. Als er geen dergelijke buur is, is het spel afgelopen en de speler die zou moeten kiezen is verloren. Dat is dus ten laatste na $n-1$ stappen het geval omdat top n geen grotere buren kan hebben. Als het spel gedaan is, vormen de gekozen bogen een pad in de graaf. De som van de gewichten van het pad is het bedrag dat de verliezende speler aan de winnaar moet betalen.



In dit voorbeeld zouden de keuzes dus kunnen zijn:

X kiest top 3, O kiest top 4, X kiest top 6, O kan niet meer kiezen. Dan is O verloren en moet $w(\{1,3\}) + w(\{3,4\}) + w(\{4,6\}) = 3+1+6 = 10$ aan X betalen.

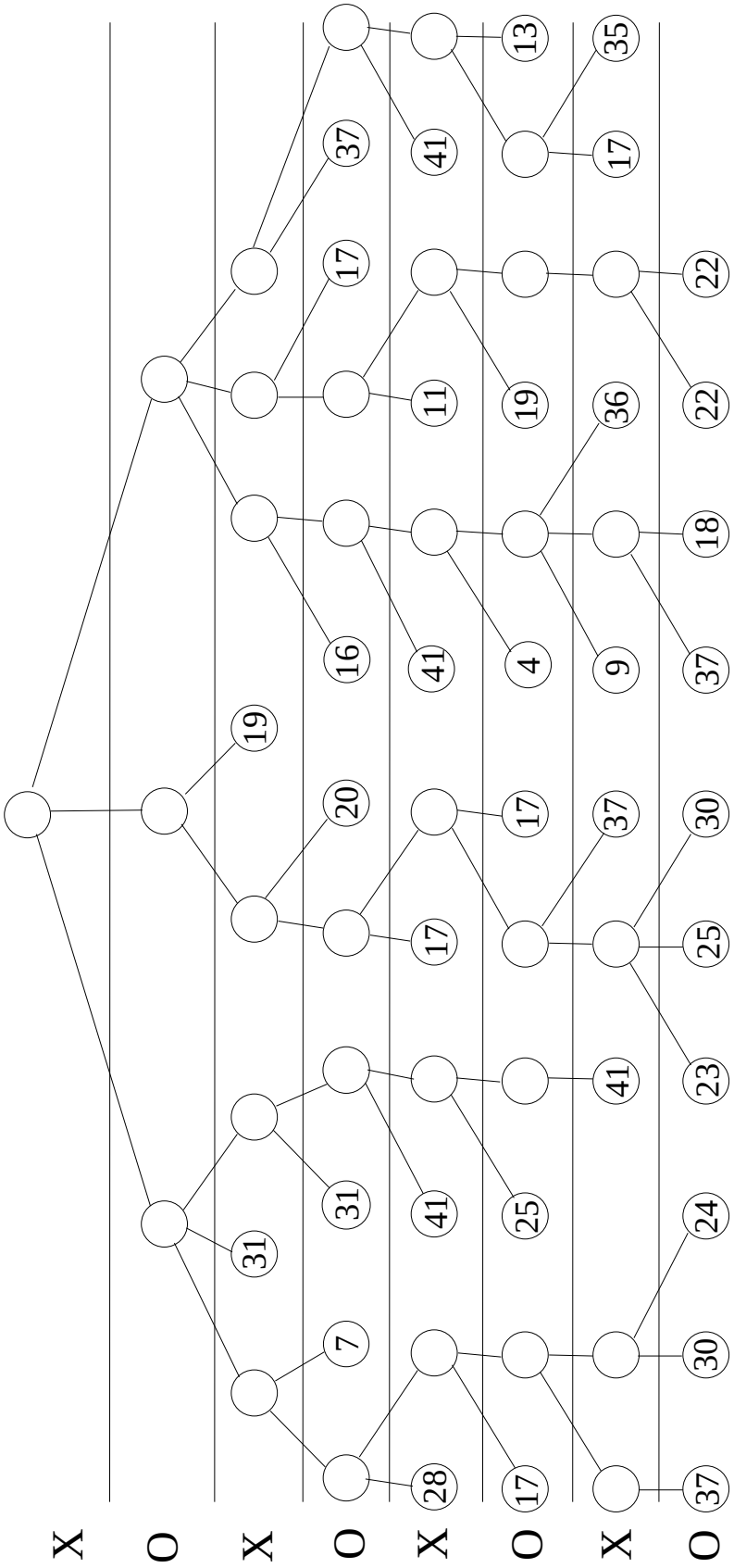
- Geef de pseudocode van een algoritme met dynamisch programmeren dat in tijd $O(n^2)$ voor elke gewogen graaf $G = (V,E)$ met toppen $\{1,2,\ldots,n\}$ en gewichten $w(e)$ voor $e \in E$ de waarde van het spel berekent. Het is voldoende de waarde te berekenen – een optimale strategie hoeft niet bijgehouden te worden.

5. **α–β–snoeien 2 pt**

- Als je het α–β–snoeien algoritme uit de les toepast dan wordt soms ook op constante paden gesnoeid. Het kan dus gebeuren dat je het constante pad niet helemaal doorloopt – de functie dus niet voor elke top op het pad oproept. Bewijs dat als er maar één constant pad in de spelboom zit dat pad wel helemaal doorlopen wordt.

  Tip: Kijk eens naar de returnwaarden van de functie als de meegegeven grenzen niet de waarde van het spel zijn.

- Pas α–β–snoeien op de volgende spelboom toe om de waarde van het spel te berekenen. Evalueer altijd eerst de linkertakken. Schrijf gebruikte grenzen **aan** de toppen en de teruggegeven waarden **in** de toppen. Plaats streepjes door de bogen die naar deelgrafen leiden die je niet evalueert omdat je snoeit.



6. **Gretige en online algoritmen 2 pt**

- Een buis van lengte l mm moet in kleine stukken gesneden worden. De lengten $l_1, \ldots, l_k \leq l$ kunnen doorverkocht worden – andere lengten zijn niet bruikbaar. Het is ook mogelijk meerdere stukken met dezelfde lengte te verkopen, dus bv. alleen maar stukken van lengte l_3 te snijden. De lengten $l_1, \ldots, l_k$ (in mm) en l zijn allemaal gehele getallen. Natuurlijk eist ook hier elke doorsnede een beetje ruimte (1 mm) – dat is dus een deel van de buis dat verloren gaat – net als een mogelijke rest waaruit geen van de stukken meer gesneden kan worden.

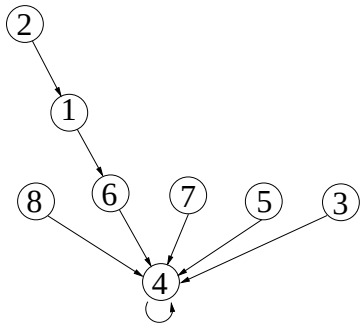
  Het doel is nu een opdeling te vinden zodat zo weinig mogelijk verloren gaat – dus waar de som van de sneden en de rest zo klein mogelijk is.

  Geef de pseudocode van een gretig algoritme waarvan je denkt dat het goed zal presteren. Geef uitleg waarom dat een *gretig* algoritme is en waarom je denkt dat het algoritme goed zal presteren.

- Wij definiëren hier een inpakalgoritme als *zwak online* als het elke keer als het een gewicht plaatst ten hoogste één van de al geplaatste gewichten mag verplaatsen. Bewijs één van de volgende twee uitspraken:
  - Voor elk zwak online algoritme A bestaat er een reeks van gewichten zodat A ten minste $\frac{5}{4}m$ vrachtwagens gebruikt als m het optimale aantal vrachtwagens voor de reeks is. (0.5 pt)
  - Voor elk zwak online algoritme A en elk getal n bestaat er een reeks van gewichten waar een optimale plaatsing $m \geq n$ vrachtwagens vraagt en A ten minste $\frac{5}{4}m$ vrachtwagens gebruikt. (1 pt)

7. **Verzamelingen 1.5 pt**

- Kan de volgende union-find datastructuur het resultaat zijn van union-by-size bewerkingen met path-compression toegepast op een verzameling van oorspronkelijk allemaal geïsoleerde toppen? Geef ofwel een reeks van bewerkingen die dit resultaat oplevert of bewijs dat het niet kan.



- Je werkt op een computer met woorden van 64 bit. Het universum waarmee je bezig bent, bevat de elementen $0, \ldots, 100$. Je hebt verzamelingen $M_1, \ldots, M_k$ die als bitvectoren zijn voorgesteld en wilt de unie van alle verzamelingen vormen die een zeker element i met $0 \leq i \leq 100$ bevatten. Geef de pseudocode van een routine `vorm_unie(i)` die dat doet. Gebruik daarbij de bit-operatoren uit de les (`<<`, `|`, `~`, etc) die je nodig hebt. Beschrijf eerst hoe die M_i precies zijn voorgesteld.

8. **Gerandomiseerde algoritmen 1.5 pt**

In een examen met 80 studenten mogen lesnota's met 100 pagina's gebruikt worden maar geen andere teksten. De vraag is nu of een zekere student X de lesnota's heeft gewijzigd en delen door andere teksten heeft vervangen. Wij gaan ervanuit dat geen 2 studenten dezelfde wijzigingen hebben gedaan. Natuurlijk zijn de studenten slim genoeg ervoor te zorgen dat het nog altijd 100 pagina's zijn en dat de meeste pagina's ongewijzigd zijn. De enige manier om te toetsen of student X sjoemelt, is pagina's van zijn kopie van de lesnota's te vergelijken met dezelfde pagina uit andere lesnota's. Jammer genoeg is de lesgever zijn eigen kopie vergeten, zodat hij de lesnota's van student X alleen met andere lesnota's van studenten kan vergelijken. Wij gaan ervanuit dat 4 studenten sjoemelen en dat studenten die sjoemelen 20 pagina's gewijzigd hebben. Gezocht is een efficiënt Monte Carlo algoritme om student X te toetsen dat als resultaat ofwel `eerlijk` ofwel `oneerlijk` teruggeeft. Er zijn wel eisen aan de kans op een fout antwoord:

Het antwoord `eerlijk` moet met kans ten minste 99% juist zijn. Het antwoord `oneerlijk` mag met kans ten hoogste 5% fout zijn – in dit tweede geval zouden nog meer toetsen volgen, maar dat is geen deel van deze oefening.

- Het volgende algoritme voldoet niet aan de vereisten – waarom niet (behalve dat het zeker niet efficiënt is)?
  - Vergelijk alle pagina's van de lesnota's van X met alle pagina's van de lesnota's van alle andere studenten.
  - Geef `eerlijk` terug als ze allemaal identiek zijn en anders `oneerlijk`.

- Geef een efficiënt algoritme dat wel aan de eisen voldoet. Niet alle waarden van parameters van jouw algoritme moeten expliciet gegeven worden – als je bv. een zekere formule $f(n)$ hebt, die voor grote *n* tegen 0 gaat, is het in orde als je gewoon zegt *"Kies n zo groot dat* $f(n) \leq \ldots$*"*.

NOG NIET OMDRAAIEN !