Examen Datastructuren en Algoritmen II

Naam : .....

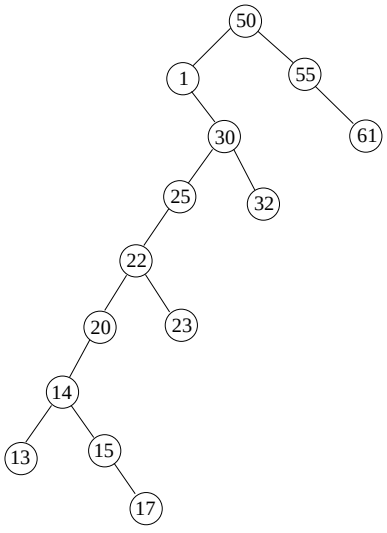
1. **Splaybomen** (2 pt)

   In deel a.) en b.) is de O() notatie voldoende.

   a.) Hoe duur is één toevoegbewerking op een splay-boom met *n* toppen in het slechtste geval ? (Geen bewijs vereist.)

   b.) Hoe duur is een reeks van *n* toevoegbewerkingen op een initieel lege splayboom in het slechtste geval ? (Geen bewijs vereist.)

   c.) Voeg de sleutel 18 op een splay-manier toe aan de volgende boom. Toon voldoende tussenstappen om te laten zien wat er gebeurt, maar voor de tussenstappen moet je niet elke keer de **hele** boom tekenen.

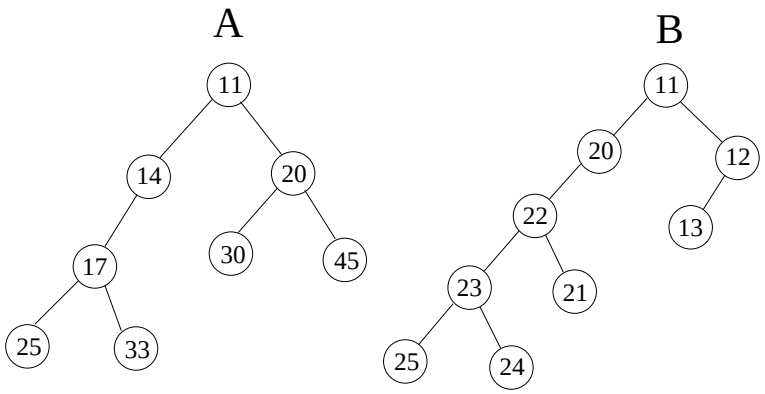


2. **Leftist heaps:** (2 pt)

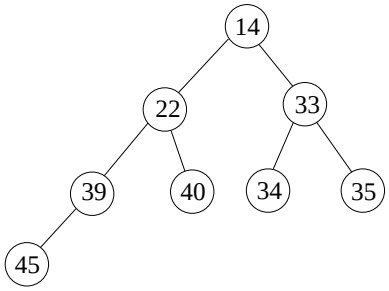
a.) Mag een leftist heap dezelfde sleutel meerdere keren bevatten ?

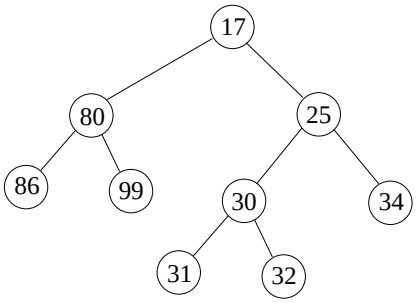
b.) Wat is het effect op de efficiëntie of juistheid van de in de les geziene bewerkingen voor leftist heaps als je de voorwaarde uit a.) verandert ?

c.) Schrijf voor elk van de twee bomen of hij een leftist heap is en in het geval van niet waarom niet.



d.) Merge de twee volgende leftist heaps en toon voldoende tussenstappen om te laten zien wat er gebeurt.





3. **Gebalanceerde zoekbomen:** (2 pt)

In deel a.) en b.) is de O() notatie voldoende.

a.) Wat zijn in het slechtste geval de kosten van n toevoegingen op een initieel lege boom die nooit geherbalanceerd wordt? (Geen bewijs vereist.)

b.) Wat zijn in het slechtste geval de kosten van n toevoegingen op een initieel lege AVL-boom? (Geen bewijs vereist.)

c.) 2-3-bomen hebben in het slechtste geval de diepte van een perfect gebalanceerde binaire boom – dus is de diepte in bijna alle gevallen veel beter dan die van AVL-bomen en rood-zwart-bomen. Waarom zijn AVL-bomen en rood-zwart-bomen toch al interessant om te gebruiken?

d.) Gegeven een AVL-boom T met wortel x en toppen y, z op afstand d van x. Toon aan dat de hoogten van y en z ten hoogste d verschillen.

4. **Gerandomiseerde algoritmen:** (1 pt)

   Gegeven een constante r.

   In een computernetwerk met ten hoogste $|V| = \frac{1}{4} * (\frac{3}{2})^r$ computers moeten kopieën van drie verschillende databanken A, B, C zo op de computers geplaatst worden dat voor elke databank $d \in \{A, B, C\}$ elke knoop d bevat of een directe buur heeft die d bevat. Elke computer mag maar één databank bevatten. Je weet dat elke knoop tenminste r directe buren heeft. Geef een Las Vegas Algoritme dat met kans ten minste $\frac{1}{2}$ na één poging een oplossing heeft gevonden. Eén poging moet in lineaire tijd uitgevoerd kunnen worden (lineair in het aantal computers plus het aantal verbindingen tussen de computers).

5. **De waarde van een spel en dynamisch programmeren:** (3 pt)

   Een spelletje: Er liggen n 1-cent-munten op tafel. Elke speler kan 1 tot 4 munten nemen. De speler die de laatste munt neemt is verloren. De speler die wint, mag zijn munten houden en de speler die verliest moet de genomen munten terugleggen en de munten op tafel bovendien opvullen tot er opnieuw n munten zijn. De ene speler verliest dus altijd even veel als de andere wint.

   a.) Teken de spelboom voor $n = 6$. Pas alleen het minimax principe en α-β-snoeien toe en bepaal de waarde van het spel. Takken die door α-β-snoeien gesnoeied kunnen worden moeten niet getekend worden.

   b.) Geef een algoritme met dynamisch programmeren voor dit probleem (geef b.v. de pseudocode). Pas jouw algoritme toe voor $n = 8$.

6. **Geamortiseerde complexiteitsanalyse: Een stapel met een extra bewerking:** (2 pt)

   In een bedrijf worden taken op een stapel bijgehouden omdat de nieuwste taak altijd de meest belangrijke is. Bovendien zijn heel oude taken zo weinig belangrijk dat ze, als er voldoende taken zijn, gewoon verwijderd kunnen worden. Voor deze taak bestaat er – naast de gewone push- en pop-bewerkingen nog de bewerking *verwijder-oudste-helft* die de oudste helft van de stapel verwijdert.In het geval van een oneven aantal n van elementen worden $(n-1)/2$ elementen verwijderd. Het aantal n van elementen op de stapel wordt altijd bijgehouden. De verwijder-oudste-helft-bewerking gebeurt door de eerste $n/2$ elementen door middel van een pop-bewerking te lezen en door middel van een push-bewerking op een nieuwe stapel te plaatsen. De tweede $n/2$ elementen worden door middel van een pop-bewerking verwijderd. Dan worden de $n/2$ elementen op de nieuwe stapel door middel van een push- en een pop-bewerking terug op de stapel geplaatst. Als wij push- en pop-bewerkingen als één stap tellen, vraagt zo'n verwijder-oudste-helft-bewerking dus $(5/2)n$ stappen.

   Toon aan dat een reeks van m bewerkingen (push, pop of verwijder-oudste-helft) op een initieel lege stapel $O(m)$ stappen vraagt. Gebruik de potentiaalmethode.

   Tip: Een potentiaal die niet sterk verschilt van de potentiaal die wij in de les voor de multipop bewerking gezien hebben, kan ook hier gebruikt worden.

7. Dit is de vervangoefening voor de huiswerkoefeningen. Ze staat op 4 punten. Voor het examen wordt het maximum van de in deze oefening verworven punten en de punten uit de huiswerkoefening geteld. Wie in de huiswerkoefeningen al goed presteerd heeft zou deze oefening dus het best ofwel niet ofwel alleen maar als hij/zij met alle andere oefeningen klaar is bekijken.

   Tonen dat iets slecht presteert: Het inpak probleem (4 pt)

   Definieer een algoritme voor het inpak probleem als *zwak online* als voor een reeks $g_1, \ldots, g_n$ van gewichten elk gewicht g_i onmiddellijk geplaatst wordt – zonder rekening te houden met de gewichten $g_{i+1}, \ldots, g_n$ (dat is zoals met online algoritmen) maar elke keer dat een nieuw gewicht g_i wordt geplaatst mag één van de gewichten $g_1, \ldots, g_{i-1}$ op een nieuwe plaats gezet worden (dat mag een online algoritme niet).

   Schrijf voor elk van de twee volgende zinnen of ze juist zijn en toon aan dat jouw antwoord juist is:

   a.) Er bestaat een oneindige reeks van inputsets $i_1, \ldots, i_m$ voor het inpakprobleem zodanig dat voor elke i_k in de reeks ten minste k vrachtwagens nodig zijn en elk zwak online algoritme voor elke input i_k uit deze reeks ten minste $4/3$ keer het optimum aantal vrachtwagens gebruikt.

   b.) Voor elk zwak online algoritme A bestaat een oneindige reeks van inputsets $i_1, \ldots, i_m$ zodanig dat voor elke i_k in de reeks ten minste k vrachtwagens nodig zijn en A voor elke input uit deze reeks ten minste $4/3$ keer het optimum vrachtwagens gebruikt.

   Tip: Denk aan de reeks uit de les, maar splits de kleine stukken in zo veel nog kleinere stukken op dat herverdelen niet meer mogelijk is als de grote stukken worden geplaatst.

NOG NIET OMDRAAIEN !