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)!

1. **Zoekbomen (4 pt)**

   - Voeg de volgende sleutels in de gegeven volgorde toe aan een initieel lege rood-zwart-boom. Gebruik voor het herbalanceren daarbij de in de les gezien geoptimaliseerde bewerkingen.

     De sleutels zijn: 6, 2, 3, 5, 1, 4 en 7

- Wat is de minimale en wat is de maximale diepte van een rood-zwart-boom met n sleutels? De $O()$-notatie is in beide gevallen voldoende en een bewijs is niet vereist.

- Geef een voorbeeld van een rood-zwart-boom en een 2-3 boom met dezelfde sleutels zodat er opzoekbewerkingen zijn waarvoor de rood-zwart-boom minder vergelijkingen nodig heeft en opzoekbewerkingen waarvoor de 2-3 boom minder vergelijkingen nodig heeft. Je mag ervanuit gaan dat in een 2-3 boom in een top met 2 sleutels altijd eerst met het linkerkind vergeleken wordt.

- Geef een voordeel van semi-splay bomen in vergelijking met rood-zwart bomen en een voordeel van rood-zwart bomen in vergelijking met semi-splay bomen. Geef elke keer ook een toepassing waar dit voordeel goed van pas komt.

2. **Dynamisch programmeren (2 pt)**

   Op een $n \times n$ schaakbord kan je de vakken met de coordinaten (x,y) labellen waarbij $0 \leq x,y < n$. Elke grens tussen twee buurvakken a en b heeft een waarde $w(a,b)$ die ook negatief kan zijn.

   Van een vakje (x,y) kan je naar een vakje (x',y') gaan waarbij $x' = x+1$ en $y' = y$ of $x' = x$ en $y' = y+1)$ (waarbij natuurlijk $0 \leq x',y' < n$).

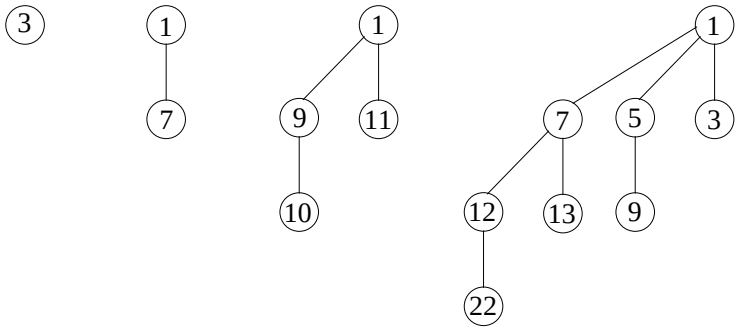
   Gezocht is nu een pad van $(0,0)$ naar $(n-1,n-1)$ zodat de som van de waarden van de grenzen die je hebt overgestoken maximaal is. Geef een dynamisch programmeren algoritme in pseudocode dat dit pad (niet de som van de waarden) als resultaat oplevert.

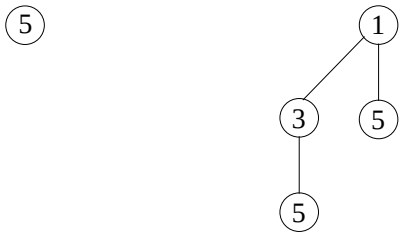
3. **Wachtlijnen (4 pt)**

- In een binomiale prioriteitswachtlijn is het belangrijk dat sleutels meerdere keren mogen opduiken. Waarom precies?

- Welke wachtlijnen heb je gezien waar het geen problemen zou opleveren als identieke sleutels verboden zouden zijn?

- Merge de volgende twee binomiale prioriteitswachtlijnen:





- Bewijs dat in een binomiale prioriteitswachtlijn met n toppen er een boom met diepte k aanwezig is als en slechts als in de binaire voorstelling van n de k-de bit 1 is (waarbij natuurlijk de 0-de bit die met de laagste waarde is). De gemakkelijkste manier is misschien met inductie, maar je kan zelf kiezen welke techniek je gebruikt.

4. **Verzamelingen (2 pt)**

   Je kan union-by-depth definieren door vast te leggen dat elke keer als twee union-find-datastructuren met verschillende dieptes aan elkaar geplakt moeten worden de boom met kleinere diepte aan die met grotere diepte geplakt wordt.

   - Bepaal de maximale diepte van een union-by-depth datastructuur met n elementen.

- Geef een universum en een reeks van union bewerkingen zodat het resultaat van union-by-size een grotere diepte heeft dan het resultaat van union-by-depth – of bewijs dat een dergelijke reeks niet bestaat. Geef uitleg.

5. **Gerandomiseerde algoritmen (1 pt)**

   In een bedrijf dat software voor GPS toestellen schrijft, is er een probleem: blijkbaar werden sommige toestellen met een virus besmet zodat de software in gemiddeld 1 op 10 gevallen in plaats van de kortse weg te berekenen gewoon de mededeling *"blijf thuis"* uitvoert. De toestellen worden in dozen met 10 toestellen naar de winkels gestuurd.

   Je werkt in de afdeling kwaliteitscontrole en het is nu jouw taak een efficiënt algoritme te ontwikkelen dat bepaalt of dozen naar de winkel gestuurd mogen worden of niet. Daarbij kan je de software in elk GPS toestel testen door het routes te laten berekenen en de uitvoer te testen. De bedoeling is dat als er een doos met een besmet toestel is, de kans dat de doos niet als foutief herkend wordt ten hoogste 0.01% is.

   Beschrijf en analyseer een gerandomiseerd algoritme voor dit probleem. Welk soort algoritme is het?

6. **Het inpakproblem (1 pt)**

   Wij hebben gezien dat als je elk pakket zo plaatst dat er zo weinig mogelijk ruimte over is (dat is min of meer de best fit heuristiek) je niet noodzakelijk de beste oplossing vindt. Maar als een pakket **perfekt** past, is het wel een goed idee het daar te plaatsen. Bewijs:

   Gegeven een deeloplossing van het inpakprobleem waarin pakketten $g_1, \ldots, g_k$ al geplaatst zijn en waar er een mogelijkheid is de pakketten $g_{k+1}, \ldots, g_n$ zo te plaatsen dat het resultaat een optimale oplossing voor $g_1, \ldots, g_n$ is. Stel nu dat er een vrachwagen v is waarop op dit moment nog precies ruimte voor g_{k+1} is (het gewicht samen met g_{k+1} is dus 1). Dan is er een optimale oplossing met $g_1, \ldots, g_k$ op de vrachtwagens waarop ze nu zitten en g_{k+1} op v.

7. **Geamortiseerde complexiteitsanalyse (2 pt)**

   Je werkt met een gebalanceerde zoekboom (b.v. een rood-zwart boom) die ook de optie heeft de slechtste helft van de toppen (die met de kleinste sleutelwaarden) te verwijderen. Of precies: Een zoekboom waarvoor de volgende twee bewerkingen gedefinieerd zijn:

   - een element toevoegen (kost ten hoogste : $c \cdot \log n$ als er achteraf n elementen in de boom zitten)
   - als er n sleutels in de boom zitten kan je de $\lfloor n/2 \rfloor$ kleinste sleutels verwijderen (kost ten hoogste $c' \cdot n$)

   Hoe een dergelijke verwijder- en herbalanceerbewerking kan gebeuren hebben wij inderdaad in een oefening gezien, maar dat is hier niet belangrijk.

   Bewijs nu dat hoewel een enkele bewerking lineaire kost kan hebben, een reeks van m bewerkingen op een initieel lege boom geamortiseerde kost $O(\log m)$ per bewerking heeft. Kies zelf welke methode je toepast.

NOG NIET OMDRAAIEN !