Tentamen Algoritmiek
Maandag 9 juli 2018, 14.00 – 17.00 uur

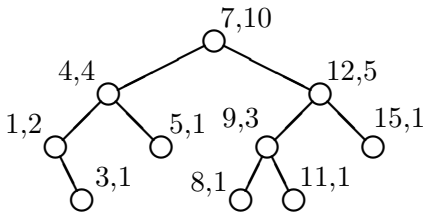
Als er om uitleg, toelichting of motivatie gevraagd wordt bij een opgave, is het belangrijk om die ook te geven.
Als je het antwoord op een onderdeel niet weet, en je hebt dat antwoord nodig bij een later onderdeel, dan kun je het antwoord 'kopen' bij de docent.
Globale puntenverdeling: 1: 22 pt; 2: 24 pt; 3: 10 pt; 4: 12 pt; 5: 32 pt. **Veel succes!**

---

1. Deze opgave gaat over binaire zoekbomen, bestaande uit knopen uit de klasse `knoop` die er als volgt uitziet:

```
class knoop
{ public:
    knoop* links;
    knoop* rechts;
    int waarde;
    int gewicht;
}; // knoop
```

Voorbeeld:
Bij de knopen staat de waarde van de velden `waarde` en `gewicht` vermeld ná het aanroepen van de functie uit onderdeel (c).



De binaire zoekbomen zijn geordend op de velden `waarde`. Bij aanvang hebben de velden `gewicht` in de boom nog geen zinvolle waarde. Die gaan gevuld worden bij onderdeel (c).

(a) Als je de waardes in een binaire zoekboom in oplopende volgorde wilt aflopen, moet je dan een WLR-wandeling, een LWR-wandeling of een LRW-wandeling uitvoeren? Motiveer je antwoord.

(b) Schrijf een *recursieve* C++-functie `void wandel (knoop *w)` die de velden `waarde` in de binaire zoekboom (of subboom) met wortel `w` in oplopende volgorde op het scherm afdrukt.

(c) Het veld `gewicht` in een knoop is bedoeld voor het gewicht van die knoop: het aantal knopen in de subboom met die knoop als wortel. Zo heeft elk blad gewicht 1, want de subboom met een blad als wortel bevat alleen dat blad. Interne knopen hebben een hoger gewicht. Zie de voorbeeldboom hierboven.

Schrijf een *recursieve* C++-functie `void vulgewicht (knoop *w)` die in elke knoop in de (sub)boom met wortel `w` het veld `gewicht` vult met de juiste waarde.

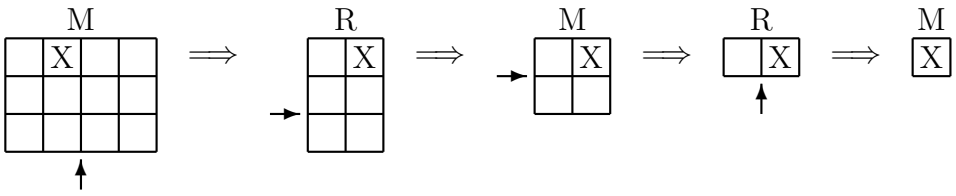
(d) Ga er nu vanuit dat alle velden `waarde` in de boom verschillend zijn. Ga er verder vanuit dat van alle knopen in een binaire zoekboom de velden `gewicht` correcte waardes hebben. Schrijf een *niet-recursieve* functie `int aantalkleinergelijk (knoop *w, int grens)`, die het aantal knopen in een binaire zoekboom boom met wortel `w` retourneert, die een waarde $\leq$ `grens` hebben.

Hint: kies (voor jezelf) als voorbeeld een bepaald getal `grens`, en bepaal welke knopen in de voorbeeldboom hierboven een waarde $\leq$ `grens` hebben.

---

2. Als troost voor een mislukt WK hebben Messi en Ronaldo (samen) een grote plak chocolade gekregen, bestaande uit $m \times n$ stukjes, waarbij m het aantal rijen is en n het aantal kolommen. Een van de stukjes is echter bedorven. Om de beurt breekt een van de twee vrienden de (resterende) plak langs een horizontale of verticale breuklijn in tweeën. Hij mag dan het deel zonder bedorven stukje opeten. Het andere deel (met het bedorven stukje) geeft hij aan de ander, die dan aan de beurt is. Dit gaat zo door tot alleen het bedorven stukje over is. Degene die op dat moment aan de beurt is, heeft verloren: hij moet het bedorven stukje opeten. Als jongste van de twee mag Messi beginnen.

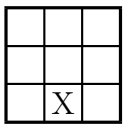
   Bij een plak van 3×4 stukjes kan dit als volgt verlopen:



   Hierin geeft de X het bedorven stukje chocolade aan. In dit geval verliest Messi dus.

   (a) Wat zijn voor dit spel de toestanden en de acties (voor algemene $m, n \geq 1$)? Wanneer is een toestand een eindtoestand?

   (b) We noemen een toestand *winnend* voor een speler, als die speler gaat winnen bij optimaal spel van beide spelers vanaf die toestand. Teken de toestand-actie-ruimte voor het geval de oorspronkelijke plak chocolade er als volgt uitziet:



   Geef bij elke actie (= tak) in je toestand-actie-ruimte duidelijk aan om welke actie het gaat. Geef bij *elke* toestand aan of deze winnend is voor M (Messi) of R (Ronaldo), te beginnen bij de eindtoestanden. Bepaal zo of het spel met bovenstaande plak chocolade winnend is voor M of voor R.

   Toestanden die je al hebt uitgewerkt (of die door rotatie en/of spiegeling overeenkomen met een toestand die je al hebt uitgewerkt) hoef je niet nogmaals uit te werken. Zet daar wel bij wie er wint en verwijs naar de reeds uitgewerkte toestand.

   (c) Beredeneer dat als de oorspronkelijke plak chocolade vierkant is (dus $m = n$) en het bedorven stukje op een van de twee hoofddiagonalen ligt (van linksboven naar rechtsonder of van linksonder naar rechtsboven) het spel verliezend is voor de speler die begint.

   Hint: kijk of je een dergelijke toestand ook in je toestand-actie-ruimte van onderdeel (b) hebt.

3. Stel dat a_1, a_0, b_1, b_0 vier cijfers zijn, dan kun je het product $a_1a_0 \times b_1b_0$ als volgt rechtstreeks uitrekenen:

$$\begin{aligned} a_1a_0 \times b_1b_0 &= (a_1 \cdot 10^1 + a_0 \cdot 10^0) \times (b_1 \cdot 10^1 + b_0 \cdot 10^0) \\ &= a_1 \times b_1 \cdot 10^2 + (a_1 \times b_0 + a_0 \times b_1) \cdot 10^1 + a_0 \times b_0 \cdot 10^0 \end{aligned}$$

Dit kost in totaal vier vermenigvuldigingen, één voor elke combinatie van cijfers uit a_1a_0 en b_1b_0.

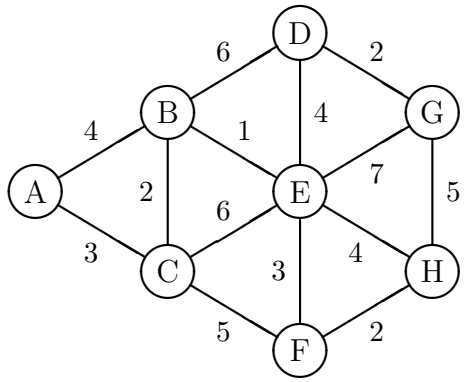
(a) Laat zien hoe we het product $a_1a_0 \times b_1b_0$ ook met maar drie vermenigvuldigingen kunnen uitrekenen.

(b) Pas de methode uit het vorige onderdeel toe op het product 13×42. Wat is de resulterende waarde van het product?

---

4. Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen. Pas het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop A. Geef voor elke stap van het algoritme duidelijk aan welke knoop erbij wordt gekozen in U (= verzameling knopen waarvan de kortste afstand vanaf A bekend is) en welke labels door die keuze veranderen en hoe. Licht toe hoe je uitwerking gelezen moet worden (legenda).

Geef ook de resulterende boom van kortste paden met daarin de bijbehorende lengtes van de kortste paden vanaf A.



Z.O.Z.

5. We noemen een verzameling positieve getallen *delervrij*, als voor elk tweetal verschillende getallen in de verzameling geldt dat hun grootste gemene deler gelijk is aan 1. Bijvoorbeeld: de verzameling $\{19, 20, 21, 23\}$ is delervrij, maar de verzameling $\{19, 20, 21, 22\}$ is niet delervrij, omdat de grootste gemene deler van 20 en 22 gelijk is aan 2. De lege verzameling en een verzameling met één getal zijn per definitie delervrij.

   (a) De verzameling $\{2, 5, 6, 9, 10\}$ is zelf niet delervrij, maar je kunt wel deelverzamelingen kiezen die wél delervrij zijn. Geef alle delervrije deelverzamelingen *met minstens twee elementen* van deze verzameling.

   (b) Ga er nu vanuit dat een verzameling met N positieve getallen is opgeslagen in een array `verz`, op posities `verz[0]`, ..., `verz[N-1]`. [1] Ten overvloede: alle N getallen zijn verschillend.

   Schrijf een *recursieve* C++-functie `void bepaaldelervrij (int verz[], int N, int i, bool gekozen[], int &aantal)` die met behulp van backtracking alle delervrije deelverzamelingen van de verzameling `verz` opbouwt. De parameters `i` en `gekozen` betekenen dat we tot nu toe uit uit de getallen `verz[0]`, ..., `verz[i-1]` een aantal getallen hebben gekozen, namelijk die getallen `verz[j]` waarvoor `gekozen[j]` `true` is. We doen (in ieder geval) een recursieve aanroep, iedere keer dat we een getal toevoegen aan de deelverzameling. De parameter `aantal` houdt het aantal delervrije deelverzamelingen bij. Die wordt met 1 opgehoogd, iedere keer dat we een nieuwe delervrije deelverzameling hebben gevonden.

   De eerste aanroep zal van de vorm `bepaaldelervrij (verz, N, 0, gekozen, aantal);` zijn, waarbij `aantal` is geïnitialiseerd op 0, en het array `gekozen` is geïnitialiseerd op `false`. Je mag ervan uitgaan dat er een functie `int ggd (int m, int n)` is, die de grootste gemene deler van twee positieve getallen `m` en `n` bepaalt.

   (c) i. Beschrijf voor algemene $N \geq 1$ een 'zo slecht mogelijk geval' voor dit probleem: een verzameling `verz` met N positieve getallen, die **zo veel mogelijk** delervrije deelverzamelingen heeft.

   ii. Hoeveel delervrije deelverzamelingen kent `verz` in dit 'zo slecht mogelijke geval'? Wat zegt dit dus over de worst-case tijdcomplexiteit van de functie `bepaaldelervrij`? Motiveer je antwoorden.

   (d) Op welke wijze zou een brute-force algoritme voor het bepalen van alle delervrije deelverzamelingen van een gegeven verzameling positieve getallen, afwijken van het backtracking algoritme van onderdeel (b)?

   (e) Je kunt je ook afvragen wat de *grootste* delervrije deelverzameling (d.w.z.: een deelverzameling met de meeste elementen) van een gegeven verzameling positieve getallen is. Een gretig algoritme daarvoor zou als volgt kunnen werken:

   - Sorteer de getallen van klein naar groot.
   - Begin met de lege deelverzameling.
   - Loop de getallen van klein naar groot af, en voeg een getal toe aan de deelverzameling als het een grootste gemene deler 1 heeft met elk van de getallen die al in de deelverzameling zitten.

   Geef een voorbeeld van een verzameling positieve getallen waarvoor dit gretige algoritme geen correct resultaat geeft. Wat is bij dit voorbeeld het resultaat, en wat zou een correct resultaat zijn?

---

[1] Inderdaad, we zouden ook met de klasse `set` kunnen werken, maar dat doen we dus niet.