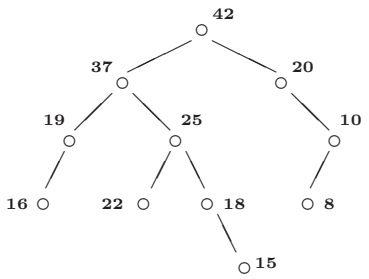
Tentamen Algoritmiek
Donderdag 7 juli 2016, 10.00 – 13.00 uur

Geef een **duidelijke toelichting** bij al je antwoorden.
Puntenverdeling: 1: 28; 2: 24; 3: 18; 4: 30. **Veel succes !**

Opgave 1. Gegeven een binaire boom met ingang (ofwel: wortel) `wortel`. Hierin is `wortel` een pointer naar een `knoop`, die er als volgt uitziet:

```
class knoop {
  public:
    knoop* links;
    knoop* rechts;
    int info;
    bool heap;
}; // knoop
```

Voorbeeld:



Bij aanvang van onderdeel **a** zijn alle `info`-velden gevuld (en verschillend). De `heap`-velden hebben nog geen waarde. De boom kan leeg zijn.

a. (8 punten)
Schrijf een *recursieve* C++-functie `void vullen(knoop* wortel)` die in elke knoop het `heap`-veld op `true` zet indien de (sub)boom met die knoop als ingang de heapeigenschap[1] heeft en anders `false`. Het `heap`-veld van een knoop zonder kinderen krijgt de waarde `true`.
In de voorbeeldboom krijgen alle knopen `heap`-waarde `true`.

De rest van deze opgave gaat over heaps en staat los van **a**.
Een *heap* (hoopstructuur) is een complete binaire boom die de heapeigenschap heeft. De voorbeeldboom is niet compleet, maar heeft wel de heapeigenschap.

b. (4 punten)
(i) Wat is de hoogte van een complete binaire boom met 8 knopen? En met 16 knopen?
(ii) Wat is in het algemeen de hoogte van een complete binaire boom met $n = 2^k$ knopen?
c. (8 punten)
Teken de met de rij 49 75 63 26 57 85 35 98 82 65 corresponderende complete binaire boom en breng deze met behulp van *heapify* in hoopstructuur. Licht duidelijk toe wat er gebeurt (wat doe je en in welke volgorde) en laat tussenstappen zien. Geef ten slotte het eindresultaat ook weer als een rij.
d. (8 punten)
Leg uit hoe het sorteeralgoritme Heapsort werkt. Voer ter illustratie daarbij de eerste twee stappen van Heapsort uit op de hoopstructuur die in **c** verkregen is en toon tussenstappen. Na afloop hiervan moeten dus de twee grootste getallen op hun goede plek achteraan staan in de met de boom corresponderende rij. Laat ook zien hoe deze rij er na de tweede stap uitziet.

[1] Dat wil zeggen dat in *elke* knoop geldt dat de `info`-waarde die in die knoop is opgeslagen groter of gelijk is aan de `info`-waarde die is opgeslagen in zijn kinderen.

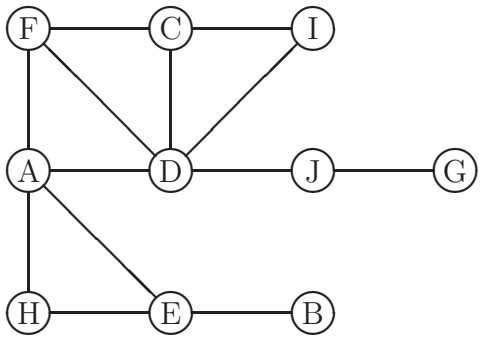
Opgave 2. Gegeven een array $A = A[0], A[1], \ldots, A[n-1]$ dat n (≥ 3) nullen en enen bevat. We willen weten of er ergens in het array drie enen direct naast elkaar voorkomen. We gaan dit probleem op drie manieren oplossen: met brute force, met decrease-and-conquer en met divide-and-conquer.

Voorbeeld: in het rijtje $0, 1, 1, 0, 1, 1, 1, 0$ komen drie enen naast elkaar voor; in $0, 1, 1, 0, 1, 1, 0, 1$ is dit niet het geval.

a. (6 punten)
Geef een eenvoudig (brute force) iteratief algoritme dat `true` oplevert als ergens in het array A drie enen naast elkaar staan, en `false` als dat niet het geval is. Schrijf hiervoor een C++-functie `bool bevat111(int` A`[ ], int` n`)`.
b. (8 punten)
Geef een decrease-by-one algoritme voor bovenstaand probleem. Schrijf daartoe een *recursieve* C++-functie `bool trio111(int` A`[ ], int` i`)`, die het probleem oplost voor het (deel)array $A[0], A[1], \ldots, A[i]$ ($i \geq 1$). De aanroep `return trio111(`$A, n-1$`);` geeft dan uiteraard het antwoord voor het hele array.
c. (10 punten)
Neem aan dat n een 2-macht is. Geef nu een divide-and-conquer algoritme dat het probleem oplost. Het array dient hiervoor in twee even grote delen te worden verdeeld.
Schrijf een *recursieve* C++-functie `bool eeneeneen(int` A`[ ], int links, int rechts)` die het probleem oplost voor het deelarray A[`links`], $\ldots$, A[`rechts`] ter lengte een 2-macht.

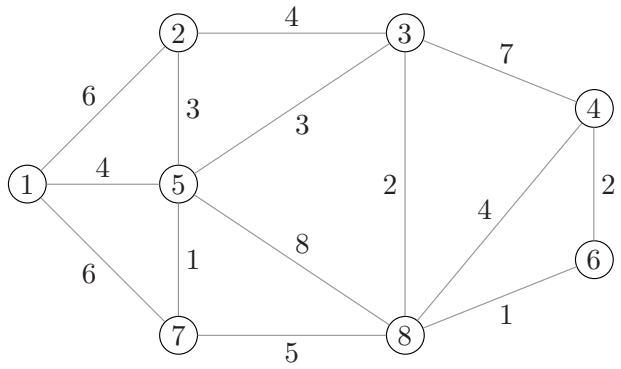
Opgave 3. Deze opgave bevat twee vragen over grafen. Ze staan echter los van elkaar.

a. (8 punten)
Beschouw de volgende ongerichte graaf G:



Maak een DFS wandeling door G, startend in knoop A. Wanneer een knoop meerdere buren heeft, handel die dan in alfabetische volgorde af.
Geef de resulterende DFS boom. Geef daarin aan in welke volgorde de knopen voor de eerste keer worden bereikt (en op de stapel gezet), en in welke volgorde ze helemaal zijn afgehandeld (van de stapel worden gehaald).
b. (10 punten)
Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen.
(i) Pas het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop 1. Geef voor elke stap van het algoritme duidelijk aan welke knoop erbij wordt gekozen in U (= verzameling

knopen waarvan de kortste afstand vanaf 1 bekend is) en welke labels door die keuze veranderen en hoe. Licht toe hoe je uitwerking gelezen moet worden (legenda).
(ii) Geef ook de uiteindelijke boom van kortste paden met daarin de bijbehorende lengtes van de kortste paden vanaf 1.



Opgave 4. Gegeven is een rij van $n \geq 2$ (met n even) gehele getallen en twee spelers, in dit geval David en Boris. Zij mogen om de beurt een getal van een van de uiteinden van de rij pakken. Aan het eind hebben beide spelers zo $n/2$ getallen in hun bezit. Degene met de hoogste totaalsom heeft gewonnen. Er wordt afgesproken dat David begint.
De uiteindelijke totale waarde van ieders verzameling getallen hangt in sterke mate af van de gemaakte keuzes in elke stap.

Voorbeeld: $n = 8$ en we beginnen met een rij bestaande uit de volgende getallen $W[i]$:

i	0	1	2	3	4	5	6	7
$W[i]$	9	3	5	15	7	10	17	11

Een mogelijke opeenvolging van keuzes door David en Boris is:
David: getal 7, 5, 1, 4 met totale waarde 31; Boris: getal 6, 0, 2, 3 met totale waarde 46.
Een andere mogelijke serie keuzes getallen is:
David: getal 0, 6, 5, 2 met totale waarde 41; Boris: getal 7, 1, 4, 3 met totale waarde 36.

Uiteraard willen David en Boris allebei een zo groot mogelijke totaalsom bereiken. Ze kiezen daartoe beiden optimaal. Dat betekent dat elke speler als hij aan de beurt is een getal kiest dat hem het beste eindresultaat oplevert. Merk op dat wat goed is voor David juist niet goed is voor Boris: een hogere eindwaarde voor David geeft immers een lagere waarde voor Boris.
We willen berekenen wat de maximale eindwaarde is die David kan verkrijgen, onder de aanname dat zowel hij als Boris optimaal kiest (onderdelen **b** t/m **e**).

a. (5 punten)
De Fibonaccigetallen zijn als volgt gedefinieerd: $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ als $n > 1$, $\text{fib}(0) = 0$ en $\text{fib}(1) = 1$. Het n-de Fibonaccigetal kan recursief worden berekend, maar dat is in dit geval niet efficiënt. Leg uit waarom niet, en geef aan hoe dynamisch programmeren de efficiëntie aanzienlijk kan vergroten.

We gaan ons probleem met behulp van *bottom-up dynamisch programmeren* aanpakken. Daartoe gebruiken we een tweedimensionaal hulparray D, waarbij $D[i][j]$ (met $j > i$) de maximale

totaalwaarde is die **David** kan bereiken als de rij bestaat uit de getallen $W[i]$ tot en met $W[j]$. Dit onder de aanname dat beide spelers optimaal kiezen. Merk op: als $j - i + 1$ (het aantal getallen in de rij) even is, is David aan de beurt, en anders Boris.

b. (5 punten)
Leg uit waarom $D[i][j]$ met $0 \leq i < j \leq n - 1$ voldoet aan:

$$D[i][j] = \begin{cases} \mathbf{max}(W[i], W[i+1]) & \text{als } j = i+1 \text{ (ofwel } j-i+1 = 2) \\ \mathbf{min}(D[i+1][j], D[i][j-1]) & \text{als } j-i+1 \text{ oneven en } \geq 3 \\ \mathbf{max}(W[i] + D[i+1][j], W[j] + D[i][j-1]) & \text{als } j-i+1 \text{ even en } \geq 4 \end{cases}$$

Hierin geeft $\mathbf{min}(a, b)$ het minimum van a en b en $\mathbf{max}(a, b)$ het maximum van a en b.

c. (4 punten)
Ga aan de hand van de recurrente betrekking uit **b** na welke array-elementen reeds bekend moeten zijn voordat $D[i][j]$ kan worden berekend en leid daaruit af hoe (in welke volgorde) het array gevuld moet/kan worden voor bottom-up dynamisch programmeren.
d. (8 punten)
Schrijf een C++-functie `void David(int` W`[ ], int` D`[ ][`n`])` die de rechterbovendriehoek van het array D vult op de in **c** aangegeven manier. Je mag aannemen dat voor aanroep alle $D[i][j]$ op nul zijn geïnitialiseerd en dat de functies `int max(int, int)` en `int min(int, int)` reeds bestaan.

Wanneer we voor het voorbeeld de rechterbovendriehoek van het array D invullen krijgen we:

0	9	5	20	18	28	28	44
-	0	5	5	18	18	35	35
-	-	0	15	15	25	25	36
-	-	-	0	15	10	27	24
-	-	-	-	0	10	10	24
-	-	-	-	-	0	17	17
-	-	-	-	-	-	0	17
-	-	-	-	-	-	-	0

e. (8 punten)
Bepaal uit deze D de maximale eindwaarde die David kan bereiken (en tevens wat Boris' eindwaarde dan wordt) en een oplossing van het probleem, dat wil zeggen de door elk van beide spelers achtereenvolgens gepakte getallen. Deze oplossing hoeft overigens niet uniek te zijn. Leg daarbij in woorden uit hoe je deze oplossing uit het array D (en W) bepaalt.
Tip: begin daartoe bij de maximale eindwaarde en bepaal met behulp van de recurrente betrekking of getal $W[i]$ of getal $W[j]$ gepakt moet worden voor de optimale oplossing. Vervolgens vanuit de aldus verkregen positie hetzelfde doen.