Tentamen Algoritmiek
Donderdag 3 juni 2021, 13.00 – 16.00 uur

Waneer er in een opgave gevraagd wordt om uitleg, toelichting of motivatie van je antwoord, is het belangrijk om die ook te geven.
De aantallen punten die bij het begin van elke opgave vermeld worden, zijn indicatief. Ze kunnen dus nog iets wijzigen.

---

1. [29 pt] De twee vrienden Maarten en Tim spelen een simpel stenenspelletje. In een rij van links naar rechts liggen $N \geq 1$ stenen, waarvan sommige van Maarten zijn en sommige van Tim. Bijvoorbeeld de rij MTTMTM: een steen van Maarten, een steen van Tim, een steen van Tim, enzovoort. Om de beurt kiest een van de spelers een van zijn eigen stenen uit de rij, waardoor er twee deelrijen overblijven: de stenen links van de gekozen steen en de stenen rechts van de gekozen steen. Een deelrij kan in principe leeg zijn, dus nul stenen bevatten. Als de ene deelrij langer is dan de ander, wordt er verder gespeeld met de *langste* deelrij. Als beide deelrijen even lang zijn, wordt er verder gespeeld met de *linker* deelrij. De stenen uit de deelrij waarmee niet wordt verder gespeeld, verdwijnen uit het spel.

   Vervolgens is de andere speler aan de beurt, die een van zijn stenen kiest uit de overgebleven deelrij, enzovoort, totdat er nog maar één soort steen over is: van Maarten of van Tim.

   Tim is als eerste aan de beurt. De speler van wie aan het eind de resterende steen is of de resterende stenen zijn, heeft gewonnen.

   (a) Hoe zien voor dit spel, bij een gegeven beginrij met stenen, de toestanden en de acties eruit? 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 complete toestand-actie-ruimte voor het geval dat de beginrij gelijk is aan TTMTM.

   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 T (Tim) of M (Maarten), te beginnen bij de eindtoestanden. Bepaal zo of het spel met bovenstaande beginrij winnend is voor T of voor M.

   Toestanden die (helemaal) gelijk zijn aan een al eerder uitgewerkte toestand hoef je niet nogmaals uit te werken. Zet daar wel bij wie er wint en verwijs naar de reeds uitgewerkte toestand.

   (c) Stel dat de beginrij bestaat uit eerst m stenen van Maarten, en dan n stenen van Tim, voor zekere $m, n \geq 1$

   Voor wie is het spel in deze situatie, afhankelijk van de waardes van m en n, winnend bij optimaal spel van beide spelers. Dit deelantwoord hoef je niet te motiveren.

   Wat is, in elke toestand die kan ontstaan in de loop van het spel in deze situatie, een zo goed mogelijke zet van de speler die aan de beurt is. Motiveer dit deelantwoord wel.

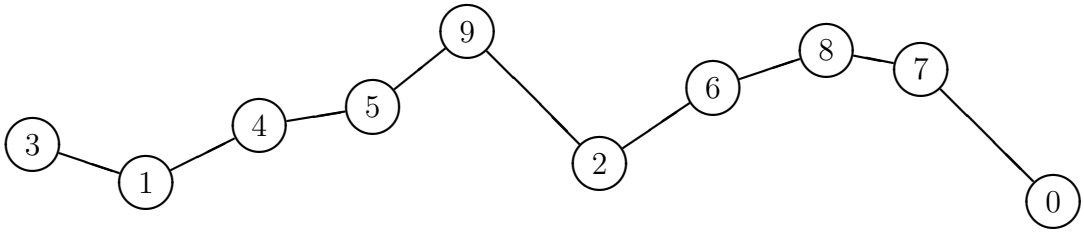
---

2. [22 pt] Gegeven een array $A = A[0], A[1], \ldots, A[N-1]$ dat $N \geq 2$ **verschillende** integers bevat. We noemen $A[i]$ een *piek-dal* (een piek of een dal), als (1) $i = 0$, of (2) $i = N-1$, of (3) $A[i-1]$ en $A[i+1]$ zijn allebei kleiner dan $A[i]$, of (4) $A[i-1]$ en $A[i+1]$ zijn allebei groter dan $A[i]$.

Bij het volgende array A

positie i	0	1	2	3	4	5	6	7	8	9
A[i]	3	1	4	5	9	2	6	8	7	0

zijn de piek-dallen de getallen 3, 1, 9, 2, 8 en 0. De naam piek-dal kan geïllustreerd worden met de volgende grafiek van de getallen in ons voorbeeldarray A:



In deze opgave moet je drie verschillende functies in C++ schrijven die het aantal piek-dallen bepalen in een algemeen array A met $N \geq 2$ verschillende integers: met brute force, met decrease-and-conquer en met divide-and-conquer.

(a) Geef een eenvoudig (brute force) algoritme dat het aantal piek-dallen in een array $A = A[0], A[1], \ldots, A[N-1]$ met $N \geq 2$ verschillende integers bepaalt. Schrijf hiervoor een *niet-recursieve* C++-functie `int piekdallen1 (int A[ ], int N)` die het gevraagde aantal bepaalt en retourneert.

Hint: Het kan gemakkelijk zijn (ook bij de onderdelen hierna) om hulpbooleans `stijg1` en `stijg2` te gebruiken, die aangegeven of $A[i-1] < A[i]$, respectievelijk $A[i] < A[i+1]$.

(b) Geef nu een decrease-by-one-and-conquer algoritme voor bovenstaand probleem. Schrijf hiervoor een *recursieve* C++-functie `int piekdallen2 (int A[ ], int i)` die het probleem oplost en het gevraagde aantal retourneert voor het (deel)array $A[0], A[1], \ldots, A[i-1]$ ($i \geq 2$).

De aanroep `piekdallen2 (A, N);` geeft dan uiteraard het antwoord voor ons hele array A.

(c) Neem aan dat N een 2-macht is (minstens 2). Geef nu een divide-and-conquer algoritme voor het probleem. Het array dient hiervoor in twee even grote delen te worden verdeeld. Schrijf een *recursieve* C++-functie `int piekdallen3 (int A[ ], int links, int rechts)` die het probleem oplost voor het deelarray A[`links`], …, A[`rechts`] ter lengte een 2-macht (minstens 2 dus).

De aanroep `piekdallen3 (A, 0, N-1);` geeft dan uiteraard het antwoord voor ons hele array A.

---

3. [29 pt] De stelling van Pythagoras luidt dat $a^2 + b^2 = c^2$.[1] Een Pythagorees drietal is een drietal positieve, gehele getallen (a, b, c) waarvoor $a < b < c$ en inderdaad $a^2 + b^2 = c^2$. Bijvoorbeeld (3,4,5) of (8,15,17).

   Laat nu $A = A[0], A[1], \ldots, A[N-1]$ een array zijn met oplopende, positieve gehele getallen, dus $1 \leq A[0] < A[1] < \ldots < A[N-1]$. De vraag is of we de getallen in A in twee groepen kunnen verdelen (de rode getallen en de blauwe getallen), zódat er geen Pythagorees drietal te vormen is met getallen uit A waarbij alle drie de getallen dezelfde kleur hebben. Dus bijvoorbeeld als 3, 4 en 5 in A voorkomen, en 3 en 4 zijn rood, dan moet 5 blauw zijn. Of als 8, 15 en 17 in A voorkomen en 8 en 17 zijn blauw, dan moet 15 rood zijn. Als dit lukt, dan spreken we van een *geschikte kleuring*.

   (a) Stel dat A het volgende array is:

positie i	0	1	2	3	4	5	6	7
A[i]	3	4	5	6	8	10	12	13

   Geef een geschikte kleuring van de acht getallen in A. Geef van elk getal aan of het rood of blauw moet worden.

   N.B.: hiervoor bestaan vele verschillende oplossingen.
   Hint: er zijn met de getallen in A drie verschillende Pythagorese drietallen te vormen. En wellicht ten overvloede: $12^2 = 144$ en $13^2 = 169$.

   We gaan nu met behulp van backtracking voor een willekeurige array $A = A[0], A[1], \ldots, A[N-1]$ met oplopende, positieve gehele getallen bepalen hoeveel verschillende geschikte kleuringen er zijn. Daarvoor moet je bij onderdeel (c) een recursieve C++-functie `int aantalKleuringen (int A[ ], int N, bool rood[ ], int k)` schrijven. De parameters `rood` en `k` geven aan dat er al een kleuring is bepaald voor de getallen $A[0], A[1], \ldots, A[k-1]$, opgeslagen in array `rood`, waarbij `rood[j]` is true, dan en slechts dan als het getal $A[j]$ rood gekleurd is. Je mag ervan uitgaan dat er bij deze gedeeltelijke kleuring nog geen Pythagorese drietallen met dezelfde kleur gekleurd zijn.

   De functie moet het aantal complete, geschikte kleuringen van $A[0], A[1], \ldots, A[N-1]$ bepalen (en retourneren), waarnaar je de gedeeltelijke kleuring in parameter `rood` kunt uitbreiden. De eerste aanroep zal dan van de vorm `aantalKleuringen (A, N, rood, 0)` zijn.

   (b) De recursieve functie moet gebruik maken van een niet-recursieve hulpfunctie `bool uitbreidingOK (int A[ ], bool rood[ ], int k)`. In de parameter `rood` van deze functie heeft nu ook `rood[k]` een waarde true of false gekregen. De functie moet true retourneren als de waarde van `rood[k]` niet tot problemen leidt, en false anders. Met andere woorden: de functie moet false retourneren, dan en slechts dan als er i en j zijn, met $0 \leq i < j < k$, zódat $A[i], A[j], A[k]$ een Pythagorees drietal vormen met dezelfde kleur.

   Schrijf deze hulpfunctie, ook in C++. Je mag gebruik maken van een functie `int square (int x)`, die x^2 retourneert, maar je mag geen gebruik maken van een functie `sqrt` (voor de wortel van een getal).

   Probeer ervoor te zorgen dat de tijdcomplexiteit van de functie lineair is in k. Als dit niet lukt, kun je nog wel een deel van de punten verdienen.

   Hint: Begin met $i = 0$ en $j = k - 1$ en loop naar elkaar toe.

[1] Hierin zijn a en b de lengtes van de twee rechthoekszijden en is c de lengte van de schuine zijde van een rechthoekige driehoek.

(c) Schrijf nu de recursieve C++-functie `aantalKleuringen` zoals die hierboven gespecificeerd is.

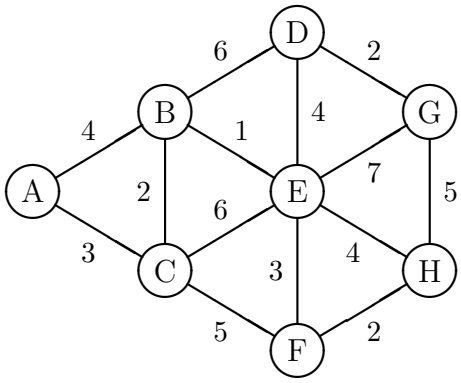
(d) Leg uit waarom er evenveel geschikte kleuringen van de getallen $A[0], A[1], \ldots, A[N-1]$ zijn waarbij $A[0]$ rood is, als waarbij $A[0]$ blauw is. Hoe kunnen we dit gebruiken om tijd te besparen bij het berekenen van het aantal geschikte kleuringen?

---

4. [20 pt] Het algoritme van Dijkstra bepaalt voor samenhangende, gewogen grafen G met knooppuntenverzameling V de (lengtes van) kortste paden vanuit een gegeven knoop s naar alle andere knopen. Het algoritme wordt beschreven door de volgende pseudo-code:

```
for v ∈ V do
        pad[v] := ∞;
od
pad[s] := 0;
U := ∅;
while ( U ≠ V ) do
        vind knoop v* ∈ V \ U met pad[v*] minimaal;
        U := U ∪ {v*};
        for alle knopen v aangrenzend aan v* do
            if pad[v*]+ gewicht(v*, v) < pad[v] then
                pad[v] := pad[v*]+ gewicht(v*, v);
                nieuwe kandidaattak voor v: (v*, v)
            fi
        od
od
```

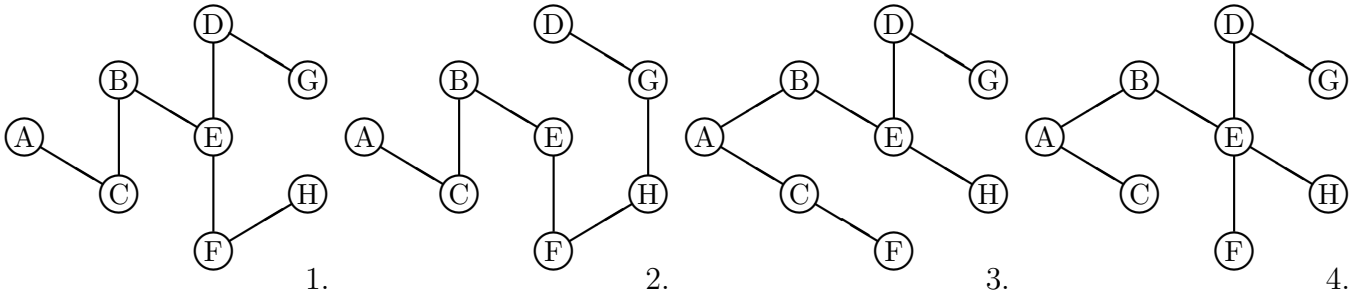
(a) Pas (op kladpapier, dus niet inleveren) het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop A.



i. In welke volgorde kunnen de knopen in de loop van het algoritme van Dijkstra (bij bovenstaande pseudocode) aan de boom van kortste paden worden toegevoegd? Nul of meer van de volgende volgordes kunnen goed zijn. Geef de nummers van alle goede volgordes.

1. A, C, B, E, D, G, F, H
2. A, C, B, E, F, D, H, G
3. A, C, B, E, F, H, D, G
4. A, C, B, E, F, H, G, D

ii. Hoe kan de resulterende boom van kortste paden (bij bovenstaande pseudocode) eruit zien? Nul of meer van de volgende bomen kunnen goed zijn. Geef de nummers van alle goede bomen.



(b) In het algemeen kan de voortgang van het algoritme van Dijkstra (bij bovenstaande pseudocode) worden weergegeven met een tabel met getallen. Elke rij van de tabel correspondeert met een iteratie van de while-lus in het algoritme, en bevat de waarden van pad[v] voor de knopen v die nog niet zijn gekozen aan het begin van die iteratie.

Voor een andere graaf dan bovenstaande graaf ziet deze tabel er als volgt uit:

A	B	C	D	E	F	G	H	Actie
0	∞	∞	∞	∞	∞	∞	∞	...
–	6	∞	∞	4	∞	6	∞	...
–	6	7	∞	–	∞	5	12	...
–	6	7	∞	–	∞	–	10	...
–	–	7	∞	–	∞	–	10	...
–	–	–	14	–	∞	–	9	...
–	–	–	13	–	10	–	–	...
–	–	–	12	–	–	–	–	...

(De kolom Actie zou je normaal gesproken ook in moeten vullen.) Leid uit deze tabel de resulterende boom van kortste paden vanaf knoop A af. Geef als je antwoord deze boom, met daarin bij elke knoop de lengte van het kortste pad vanaf knoop A.
