ALGORITMIEK: opgaven werkcollege 6

Backtracking (1, 2, 3);
Divide and conquer (4, 5, 6);
Partitie (7,8)

Opgave 1. Gegeven een rij van n positieve (> 0) gehele getallen, opgeslagen in een array `A`. Gevraagd: de lengte van het/een langste stijgende deelrijtje. (Zie ook college.) Een voorbeeld: twee stijgende deelrijen van de rij $R = 7, 4, 3, 2, 7, 5, 3, 6, 4, 5$ zijn bijvoorbeeld $4, 5, 6$ en $2, 3, 4, 5$. Deze laatste is de langst mogelijke stijgende deelrij. De rij $5, 6, 7$ is geen deelrij van R.
Een backtracking algoritme bouwt deelrijtjes bijvoorbeeld stap voor stap op door van links naar rechts telkens een nieuw element van `A` toe te voegen aan het reeds opgebouwde stijgende deelrijtje. De toe te voegen elementen zijn die waarden die in A allemaal rechts van de als laatste toegevoegde waarde staan. Voorbeeld: de deelrij $4, 5, 6$ van R kan worden uitgebreid met 4 resp. 5 tot een nieuwe deelrij van R. In elke stap wordt dan gecontroleerd of de tot dusver geconstrueerde stijgende deelrij met de toe te voegen waarde nog wel stijgend is. Zo ja, dan wordt de waarde toegevoegd en wordt het deelrijtje op dezelfde manier verder uitgebreid. Zo nee, dan wordt de volgende waarde uit `A` geprobeerd.
Schrijf een recursieve C++-functie `void langste(A, n, vanaf, deelrij, lengte, max)` die de lengte van zo'n langste stijgende deelrij oplevert. Hierin is de tot dusver opgebouwde stijgende deelrij opgeslagen in `deelrij[1]` t/m `deelrij[lengte]`, en ga je die uitbreiden met de elementen uit `A[vanaf]` t/m `A[n-1]`. Laat `deelrij[0] = 0`. Verder geeft `max` de lengte van de tot dusver gevonden langste deelrij aan.

Opgave 2. (zie college 6)
Het Knapzakprobleem luidt als volgt. Gegeven n objecten met gewicht $w_1, \ldots, w_n$ en waarde $v_1, \ldots, v_n$, en een knapzak met capaciteit W. Gevraagd: de meest waardevolle deelverzameling der objecten die in de knapzak past (dus met totaalgewicht $\leq W$).

Voorbeeld:

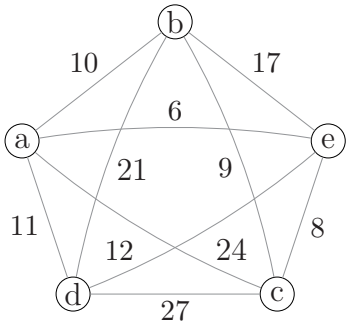
object	gewicht	waarde
1	8	42
2	3	14
3	4	40
4	5	27

knapzakcapaciteit 12

We bekijken het *backtracking* algoritme zoals dat in de collegesheets staat gegeven. Teken de state space tree die de werking van dit algoritme weergeeft op bovenstaand voorbeeld, en geef daarin de volgorde aan waarin de knopen (=deeloplossingen) bekeken worden. Doe dit voor het geval waarin deeloplossingen telkens stapsgewijs worden uitgebreid met een nieuw object in (a) de volgorde $1, 2, 3, 4$ en (b) de volgorde $1, 4, 3, 2$ en bespreek het verschil.

Opgave 3. (onderdeel oude tentamenopgave)
Het handelsreizigersprobleem (Traveling Salesman Problem) luidt: gegeven een complete, ongerichte graaf met n knopen en met gewichten op de takken. Geef een Hamiltonkring met minimaal totaalgewicht.

Beschrijf een *backtracking* algoritme voor het probleem met algemene n en licht dit toe aan de hand van onderstaand voorbeeld. Gebruik ook de lengte van de tot dusver gevonden minimale Hamiltonkring om te snoeien.
Teken een gedeelte (ongeveer de helft) van de bijbehorende state space tree die de werking van je algoritme weergeeft voor het voorbeeld, en geef daarin de volgorde aan waarin de knopen (=deeloplossingen) bekeken worden.



De kring abdcea is een Hamiltonkring met totaalgewicht $10+21+27+8+6 = 72$. Dit is *niet* minimaal.

Opgave 4. (naar Levitin, 5.1.1)
a. Geef een verdeel-en-heers algoritme (in pseudocode) voor het vinden van de index van het kleinste arrayelement uit een array van n elementen. Hierbij moet het array in twee ongeveer gelijke delen worden verdeeld.
b. (*) Laat zien dat voor $n = 2^k$ het aantal vergelijkingen dat dit algoritme doet $n-1$ is.
c. Een brute force algoritme voor dit probleem loopt van links naar rechts door het array en houdt de tot dusver gevonden minimale waarde bij. Hoeveel vergelijkingen doet dit algoritme en welke van de twee algoritmen voor het probleem is te prefereren?

Opgave 5. (naar Levitin, 5.2.11)
We hebben een collectie van n moeren van verschillende diameter en n bijbehorende schroeven. Je mag een schroef en een moer "proberen" om te kijken of het past: je kunt zo te weten komen of de schroef te groot is voor de moer, of te klein, of dat hij precies past. Je kunt echter nooit twee moeren of twee schroeven vergelijken. Het probleem is nu om bij elke moer die ene passende schroef te vinden.
Geef een verdeel en heers algoritme (in woorden) dat het probleem oplost. Hint: stop eerst wat werk in het verdelen van het probleem in twee (kleinere) versies van het probleem.

Opgave 6. (oude tentamenopgave)
Gegeven een array A dat n (≥ 2) gehele getallen $A[0], A[1], \ldots, A[n-1]$ bevat. Verder is gegeven dat op de even posities positieve getallen (> 0) staan, en op de oneven posities negatieve (< 0) getallen. Gevraagd wordt het aantal paren (i, j) met $i < j$ waarvoor $A[i] + A[j] = 0$.

a. Van welke paren (i, j) weet je al zeker dat $A[i] + A[j] \neq 0$?

Om nodeloze vergelijkingen te vermijden mogen de bij **a.** genoemde paren bij **b.** en **c.** niet bekeken worden.

b. Geef een eenvoudig (brute force) algoritme in C++, dat het gevraagde aantal oplevert. Wat is de complexiteit van je algoritme (als functie van n)?
c. Geef een divide-and-conquer algoritme in C++ voor bovenstaand probleem. Verdeel

hiertoe het array in twee gelijke delen. Neem aan dat n (≥ 2) een 2-macht is. Hier moet dus een recursieve C++-functie `int aantal2(A,links,rechts)` worden geschreven die het probleem oplost voor het deelarray A[`links`], ..., A[`rechts`] ter lengte een 2-macht.

Opgave 7. (Levitin, 5.2.8)
Reorganiseer de elementen van een gegeven array A zodanig dat alle negatieve waarden voorafgaan aan de positieve. Het algoritme moet lineair zijn (en slechts één doorgang door het array maken) en in situ (alleen interne verwisselingen). Vergelijk het partitioneren (Partitie) van het array bij Quicksort.

Opgave 8. (Levitin, 5.2.9.a: Dutch National Flag Problem)
Gegeven een array gevuld met 'R', 'W', en 'B'. Reorganiseer dit array zo dat van links naar rechts eerst alle 'R', dan de 'W' en dan de 'B' komen te staan. Het algoritme moet lineair zijn en in situ (alleen interne verwisselingen). Het mag maar één doorgang door het array maken.