ALGORITMIEK: opgaven werkcollege 8

Partitie (1,2);
Divide and conquer (3 t/m 7, 15);
Decrease and conquer (8 t/m 12);
Decrease by half (13 t/m 15)

Opgave 1. (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 2. (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.

Opgave 3. (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 4. (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 5. (uit een oud tentamen)
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 $A[\texttt{links}], \ldots, A[\texttt{rechts}]$ ter lengte een 2-macht.

Opgave 6. (Levitin, 5.4.2)
Bereken $1201 * 2430$ door het op college en Levitin 5.4 beschreven verdeel & heers (divide & conquer) algoritme toe te passen.

Opgave 7. (naar Levitin, 5.5.1)
a. We bekijken het eendimensionale closest-pair probleem: gegeven een verzameling van n reële getallen, vind de twee getallen die het dichtst bij elkaar liggen. Geef een verdeel & heers algoritme en sorteer de punten eerst.
b. Geef ook een niet-recursief algoritme.

Opgave 8. (Levitin, 4.1.1)
Een groep van n soldaten moet een rivier oversteken. Er is geen brug, maar wel twee jongens met een roeiboot. De boot kan alleen de 2 jongens (of 1 van hen) dragen zonder te zinken, of één soldaat. *Vraag:* hoe kunnen alle soldaten aan de overkant komen zodat telkens minstens één van beide jongens op dezelfde oever is waar de boot is? Geef een decrease and conquer algoritme. Hoe vaak moet de boot overvaren? Hint: breng het probleem terug van n tot $n - 1$.

Opgave 9. (Levitin, 4.1.2.a)
Er staan $2n$ glazen naast elkaar in een rij. De eerste n daarvan zijn gevuld met limonade, de andere zijn leeg. Geef een decrease and conquer algoritme dat er door overgieten (een vol glas leeggieten in een leeg glas) voor zorgt dat de rij glazen afwisselend vol, leeg, vol, leeg, etc. wordt.

Opgave 10.
Schrijf een decrease by one algoritme (in pseudocode of C++) dat alle 2^n bitstrings van lengte n genereert.

Opgave 11.
Lees op bladzijde 173-174 van Levitin wat Gray-codes zijn en maak vervolgens de volgende opgave.
a. (Levitin, 4.3.9.a) Gebruik de decrease-by-one techniek om de Gray-code voor $n = 4$ te genereren.
b. (Levitin, 4.3.9.b) Pas het volgende niet-recursieve algoritme om een Gray-code te genereren toe voor $n = 4$:
Begin met een string met n 0'en.
Voor $i = 1, 2, \ldots, 2^n - 1$, genereer de i-de bitstring door in de vorige bitstring het b-de bit te 'flippen', waarbij b de positie is van de minst significante (= achterste) 1 in de binaire representatie van i.

Opgave 12. (uit een oud tentamen)
Gegeven een array A ($A[1], \ldots, A[n]$, met $n \geq 2$), dat evenveel oneven als even getallen bevat. De oneven getallen staan op de oneven posities en de even getallen op de even posities. Het array moet —via verwisselingen— zo gereorganiseerd worden dat alle oneven getallen vooraan komen te staan, en alle even getallen achteraan.

a. Geef een eenvoudig iteratief algoritme voor dit probleem dat slechts één for-loop gebruikt. Hoeveel verwisselingen doet je algoritme?
b. Geef een decrease by four algoritme (in pseudocode of C++) voor dit probleem. Neem hierbij aan dat n een 2-voud is. Er moet dus een recursieve functie `hussel(i,j)` worden geschreven die het probleem oplost voor het deelarray $A[i], \ldots, A[j]$ ter lengte een 2-voud.

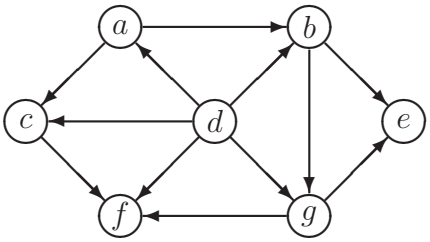
Opgave 13. (naar Levitin, opgaven 4.4.4 en 4.4.5)
Lineair zoeken heeft dezelfde complexiteit wanneer je de te doorzoeken lijst implementeert via een enkelverbonden pointerlijst als wanneer je die implementeert via een array. Hoe zit dat met binair zoeken? (Uiteraard nemen we daarbij aan dat de lijst al gesorteerd is.)

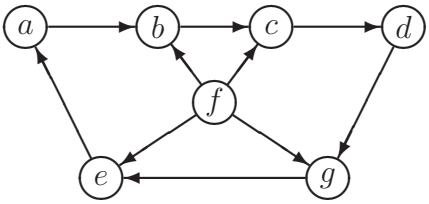
Opgave 14. (uit een oud tentamen)
Gegeven een array A met n (≥ 1) verschillende gehele getallen $A[0], A[1], \ldots, A[n-1]$. Verder is gegeven dat er een index p met $0 \leq p \leq n-1$ bestaat zodat A stijgend is tot index p, en daarna dalend.
Voorbeeld: als $A = 3\ \ 6\ \ 9\ \ 11\ \ 8\ \ 2$, dan is $p = 3$. *Randgevallen*: als $A = 7\ \ 5\ \ 3\ \ 2\ \ 1$, dan $p = 0$; als $A = 5$ (dus bestaat uit 1 element), dan $p = 0$; als $A = 4\ \ 9$, dan $p = 1$.
Geef een decrease-by-half algoritme (in pseudocode of C++) voor het bepalen van deze index p en leg uit waarom het werkt.

Opgave 15. (uit: tentamen 2013)
Gegeven een array $A = A[0], A[1], \ldots, A[n-1]$ dat n (≥ 2) verschillende gehele getallen bevat. Neem aan dat n een 2-macht is. We zoeken de *grootste* index j waarvoor geldt dat $A[j] < j$. Voor $9, 4, 1, 7, 11, 13, 3, 5, 8$ is deze index gelijk aan 7. Als zo'n index niet bestaat moeten de te schrijven algoritmen -1 opleveren.
We zullen het probleem eerst oplossen met brute force en daarna met verdeel en heers.

a. Schrijf een eenvoudig brute force algoritme in C++ dat deze index oplevert.
b. Geef een divide-and-conquer algoritme in C++ voor het probleem. Het array dient hiervoor in twee gelijke delen te worden verdeeld. Schrijf hiertoe een *recursieve* C++-functie `int grootste2(int A[ ], int links, int rechts)` die het probleem oplost voor het deelarray A[`links`], ..., `A`[`rechts`] ter lengte een 2-macht.
c. We veronderstellen nu dat het array oplopend gesorteerd is. Geef een decrease-by-half algoritme in C++ voor het bepalen van de gevraagde index. Schrijf hiertoe een *recursieve* C++-functie `int grootste3(int A[ ], int links, int rechts)`. Leg uit waarom je algoritme werkt, dus waarom je telkens maar één van beide helften hoeft te bekijken. Gebruik daarbij wat gegeven is over het array (verschillende gehele getallen, oplopend gesorteerd).

Opgave 16. **a.** (Levitin, 4.2.1) Apply the DFS-based algorithm to solve the topological sorting problem for the following digraphs.





b. Apply the source-removal algorithm to solve the topological sorting problem for the same two digraphs.

Opgave 17. **a.** (Levitin, 4.2.6.a) Prove that a nonempty DAG must have at least one source.
b. (Levitin, 4.2.2.a) Prove that the topological sorting problem has a solution, if and only if the digraph is a DAG.
c. (Levitin, 4.2.2.b) For a digraph with n vertices, what is the largest number of distinct solutions the topological sorting problem can have?