ALGORITMIEK: opgaven werkcollege 8

Divide and conquer (1 t/m 2, 10); Decrease and conquer (3 t/m 7);
Decrease by half (8 t/m 10); Kortste pad met BFS (11);

Opgave 1. (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 2. (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 3. (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 4. (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 5.
Schrijf een decrease by one algoritme (in pseudocode of C++) dat alle 2^n bitstrings van lengte n genereert.

Opgave 6.
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 7. (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.

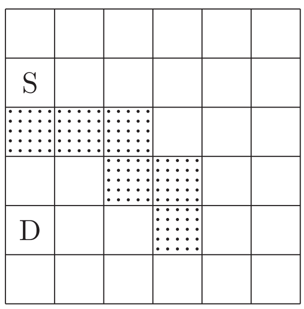
Opgave 8. (naar Levitin, opgaven 4.4.4 en 4.4.5)
Lineair zoeken heeft dezelfde complexiteit wanneer je de te doorzoeken lijst implementeert via een enkelverbonden pointerlijst als wanneer je die implementeert via een array. Hoe zit dat met binair zoeken? (Uiteraard nemen we daarbij aan dat de lijst al gesorteerd is.)

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

Opgave 10. (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`]$, \ldots,$ `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 11. Beschouw het volgende 6x6 bord:



Op dit bord kun je in één stap van een vakje naar een aangrenzend vakje (horizontaal, verticaal, diagonaal) lopen. De gearceerde vakjes zijn verboden terrein.
a. Voer een Breadth First Search uit op dit bord, uitgaande van vakje S. Schrijf in elk (niet-gearceerd) vakje de resulterende afstand vanaf S.
b. Bepaal, uitgaande van je antwoord bij **a.**, alle kortste paden van S naar D. Hoeveel verschillende kortste paden van S naar D zijn er?