ALGORITMIEK: opgaven werkcollege 8

Divide and conquer (1 t/m 2); Decrease and conquer (3 t/m 5); Decrease by half (6); Dynamisch programmeren (7)

Opgave 1. (Levitin, 5.4.2 **(vgl. tweede editie, opgave 4.5.2)**)
Bereken $1201 * 2430$ door het op college en Levitin 5.4 **(zie tweede editie, paragraaf 4.5)** beschreven verdeel & heers (divide & conquer) algoritme toe te passen.

Opgave 2. (naar Levitin, 5.5.1 **(zie tweede editie, opgave 4.6.1)**)
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. (zie tweede editie, opgave 5.4.7)
Schrijf een decrease by one algoritme (in pseudocode of C++) dat alle 2^n bitstrings van lengte n genereert.

Opgave 4.
Lees op bladzijde 173-174 van Levitin **(vgl. tweede editie, bladzijde 181)** wat Gray-codes zijn en maak vervolgens de volgende opgave.
a. (Levitin, 4.3.9.a **(zie tweede editie, opgave 5.4.9.a)**) Gebruik de decrease-by-one techniek om de Gray-code voor $n = 4$ te genereren.
b. (Levitin, 4.3.9.b **(niet in tweede editie)**) 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 5. (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 6. (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 7. (uit een oud tentamen)
Langs een lang stuk snelweg zijn $n \geq 1$ posities, die steeds 500 meter van elkaar liggen, aangewezen als plekken waar reclameborden mogen worden neergezet. De (verwachte) opbrengst als gevolg van zo'n reclamebord hangt af van de positie waar het staat en is gegeven middels een 1-dimensionaal array: opbrengst$[i]$ (> 0; $i = 1, \ldots, n$) = opbrengst van een reclamebord op plek i. Reclameborden moeten altijd meer dan ($>$) 1 km van elkaar staan. De bedoeling is om reclameborden zo te positioneren dat de totaalopbrengst maximaal is.

a. Wat is de maximale totaalopbrengst als $n = 1$? En als $n = 2$? En als $n = 3$?
b. Geef een recursieve formulering voor `maxtotaal`(n), de maximale totaalopbrengst. De basisgevallen ($n = 1, 2, 3$) zijn in **a** berekend.
c. Het probleem kan recursief worden opgelost, maar dat is in dit geval niet efficiënt. Leg uit waarom niet, en geef aan hoe dynamisch programmeren de efficiëntie aanzienlijk kan vergroten. Bespreek vervolgens (kort) zowel de top down methode als de bottom up methode voor dynamisch programmeren en leg het verschil uit.
d. Geef in pseudocode of in C++ een bottom up dynamisch programmeren algoritme dat de maximale totaalopbrengst berekent. Formuleer daarbij duidelijk wat voor array je gebruikt en geef de recurrente betrekking volgens welke het array gevuld wordt.