

ALGORITMIEK: opgaven werkcollege 7

Backtracking (1, 2);
Divide and conquer (3, 4, 5, 8);
Decrease and conquer (6,7);
Partitie (9, 10)

Opgave 1. (zie college 6; sheet 14 en 15)

Het Knapzakprobleem luidt als volgt. Gegeven n objecten met gewicht w_1, \dots, w_n en waarde v_1, \dots, 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

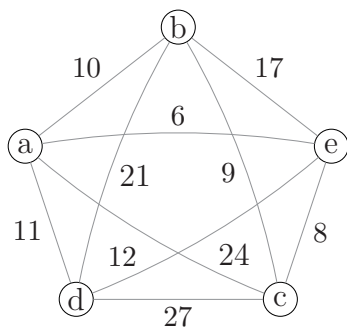
We bekijken het *backtracking* algoritme zoals dat in de collegesheets staat gegeven. Teken de state space tree die de werking van dit algoritme weergeeft op bovenstaand voorbeeld, en geef daarin de volgorde aan waarin de knopen (=deeloplossingen) bekeken worden. Doe dit voor het geval waarin deeloplossingen 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 2. (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 3. (naar Levitin, 5.1.1 (vgl. tweede editie, opgave 4.1.1))

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 (zie tweede editie, opgave 4.2.11))

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. (naar Levitin, opgaven 4.4.4 en 4.4.5 (vgl. tweede editie, opgave 4.3.5))

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 6. (Levitin, 4.1.1 (zie tweede editie, opgave 5.1.1))

Een groep van n soldaten moet een rivier oversteken. Er is geen brug, maar wel twee jongens met een roeiboort. 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 7. (Levitin, 4.1.2.a (zie tweede editie, opgave 5.1.2))

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 8. (oude tentamenopgave)

Gegeven een array A dat n (≥ 2) gehele getallen $A[0], A[1], \dots, A[n - 1]$ 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 $A[\text{links}], \dots, A[\text{rechts}]$ ter lengte een 2-macht.

Opgave 9. (Levitin, 5.2.8 (zie tweede editie, opgave 4.2.8))

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 10. (Levitin, 5.2.9.a (zie tweede editie, opgave 4.2.9): 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.