Universiteit Utrecht

Opgave 1 (a)

Een ongerichte graaf heet 3-regulier als elke knoop graad 3 heeft.

- Hoeveel tijd kost het om te testen of een graaf 3-regulier is als gegeven met adjacency list datastructuur?

- Graad: aantal grenzende kanten.
- O(n).
- Loop bij elke knoop de adjacency list af. Stop als je bij een knoop 4 kanten gezien hebt, of minder dan 2 kanten: O(1) tijd per knoop.
- O(n+a): -0.5 punt.

Opgave 1 (b)

- Hoeveel tijd kost het om te testen of een graaf 3-regulier is als gegeven met adjacency matrix datastructuur.

- $O(n^2)$.
- Ga alle rijen langs en test bij elke rij of er precies drie kanten (1) zijn.

Opgave 2

- Gegeven: rij posiotief gehele getallen A[1] … A[*n*], positief geheel getal B.
- Hoeveel deelverzamelingen van {A[1] … A [*n*]} hebben som precies B.
- Schrijf: M(*i*,C) als aantal deelverzamelingen van {A[1], … , A[*i*]} met som precies C.

Opgave 2(a)

Leg uit:

2. M(0,C) = 0 als C>0.
3. M(0,0) = 1.
4. Als a[i]>C, dan is M (i,C) = M(i-1, C).

1. Er bestaat geen deelverzameling van de lege verzameling met som groter dan 0.

1. De lege verzameling is de unieke deelverzameling van de lege verzameling met som 0.
2. a[i] kan niet gebruikt worden in een som met waarde C, want er zijn geen negatieve getallen. Dus elke deelverzameling van {A[1] …A[i]} met som C is een deelverzameling van {A[1] … A[i-1]}.

Totaal 1 punt.

Opgave 2(b): recurrente betrekking voor M(i,C)

Geval (3):

M(i,C) = M(i-1,C) als a[i]>C

= M(i-1,C) + M(i-1, C-a[i]) als a[i] <= C

Het aantal mogelijkheden is het aantal mogelijkheden met a[i] niet in de som (dus aantal deelverzamelingen van {A[1], … A[i-1]}, d.w.z., M(i-1, C)) plus het aantal mogelijkheden met A[i] wel in de som: de rest van de som is een deelverzameling van {A[1]…A[i-1]} met som C-A[i]; daar zijn M(i-1,C-A[i]) mogelijkheden voor.

1 punt. Toelichting onduidelijk: -0.5 punt.

Opgave 2(c): DP algoritme

```
Function NBSubsets (A[1…n], B)
Maak array M[ 0 … n, 0 … B)
For C = 1 to B do M[0,C] = 0;
M[0,0] = 1;
For i = 1 to n
Do For C = 0 to B
      Do If (a[i]>C) then M[i,C] = M[i-1,C]
                  else M[i,C] = M[i-1,C] + M[i-1, C-A[i]]
Return M[n,B]
```

Meest gemaakte fout:
Vergeten van return-statement
-0.5 pnt

1 punt

Opgave 2(d): DP met O(B) geheugen

Maak arrays M[0 … B], N [0 … B]

For C=1 to B do M[C] = 0;

M[0] = 1;

For i = 1 to n

Do For C = 0 to B

    do If (a[i]>C) then N[C] = M[C]

        Else N[C] = M[C] + M[C-A[i]]

    for C = 0 to B do M[C] = N[C]

Return M[B]

Opgave 3(a)

- Single pair shortest path probleem met Dijkstra's algoritme: zoek pad van *s* naar *v*
- Kunnen we stoppen als *v* uit C gehaald is? Waarom?

1 punt
Dit was een werkcollegeopgave.

- *Ja*.
- Als *v* uit C gehaald wordt, dan is op dat moment *d*[*v*] gelijk aan de afstand van *s* naar *v*.
- *Of*: Nadat *v* uit C gehaald wordt verandert zijn waarde *d*[*v*] niet meer.

Opgave 3(b)

- Hoeveel tijd kost Dijkstra + stoppen als *v* uit C gehaald wordt worst case als we de priority queue implementeren met een heap?

1 punt

- $O((n+a) \log n)$

(De worst case tijd verandert niet door het eerder stoppen, want *v* kan bijvoorbeeld de laatste knoop zijn die uit C gehaald wordt. Dus, net als bij `gewoon' Dijkstra $O(n+a)$ heap operaties die elk in $O(\log n)$ tijd kunnen.)

Opgave 4

- Gegeven: rij getallen A[1] … A[n], x
- A is niet noodzakelijk gesorteerd
- O(n log n) algoritme voor vraag:
  - Zijn er i, j, met A[i] + A[j] = x ?

Opgave 4; 1e oplossing

- Sorteer A, bijv. met mergesort: O(n log n)
- Voor alle i do
  - Zoek of x – A[i] in A voorkomt met binary search.
    - Dit kost n * O(log n) tijd.

Opgave 4, 2e oplossing

- Sorteer A, bijv. met mergesort: O(n log n).
- Links = 1; rechts = n.
- While links ≤ rechts
- Do
  - If A[links]+ A[rechts] = x then return YES!
  - Else If A[links] + A[rechts] < x then links ++
  - Else (A[links]+A[rechts]> x) then rechts – –;
- Return NO!

O(n)

simplification