|  |  |  |
| --- | --- | --- |
| **Toets informatica** | **VWO 6** | **PTA 13** |
| **Algoritmiek** |  | **16 april 2018** |

Deze toetst bestaat uit 10 vragen.

Veel succes!!!

**Evaluatie:**

**Vraagstelling bij 3 was niet duidelijk, lln dachten dat ze moesten kiezen uit a, be en c.**

**Opgave 4 is te moeilijk, bovendien nooit mee geoefend in de les.**

**Voor 9d wel heel veel punten (5).**

6p **Opgave 1**

Gegeven de volgende rij getallen: 13 – 8 – 5 – 4 – 12

Laat zien hoe deze gegevens gesorteerd worden met ‘Selection sort’

Geef aan welke getallen met elkaar worden vergeleken en welke getallen met elkaar worden verwisseld.

**Opgave 2**

Gegeven het volgende algoritme.

**Algoritme** Verrassing (*A*)

Invoer: array *A*[0..*n-1*] van *n* getallen, ongesorteerd (het eerste getal staat dus op positie 0)

Uitvoer: een index *i* en twee sommen te weten *som1* en *som2*

1. *i* = 0; *som1* = 0; *som2* = 0
2. zolang *i* ≤ *n-1* en *A*[*i*] ≠ 0

doe als *A*[*i*] < 0

dan *som1* = *som1* + *A*[*i*]

anders *som2* = *som2* + *A*[*i*]

*i* = *i*+1

3. Rapporteer *i*, *som1*, *som2*

Welke waarden worden bij de uitvoer gerapporteerd als

3p a Array *A* is [-4, 8, -3, -1, 7, 6, 5]

3p b Array *A* is [7, -3, 2, 6, 0, -5, 1]

4p **Opgave 3**

Stel dat je binair gaat zoeken op een ongesorteerd array. Wat gebeurt er:

1. Het algoritme wordt traag maar levert wel het correcte antwoord;
2. Het algoritme geeft een antwoord maar dat hoeft niet juist te zijn;
3. Het algoritme stopt niet.

**Kies de juiste optie en licht je antwoord toe.**

8p **Opgave 4**

Schrijf een op binair zoeken gebaseerd algoritme (kies zelf een naam) dat er voor zorgt dat uit een array *A*[0..*n-1*] van *n* gesorteerde getallen het kleinste positieve getal gerapporteerd wordt. Ga er vanuit dat het array gesorteerd is van klein naar groot, dat er zowel positieve als negatieve getallen in staan en dat er geen dubbele waarden in voorkomen.

6p **Opgave 5**

Bestudeer het volgende algoritme:

Algoritme SorteerSuperSimpel(A)

Invoer: array *A*[0..*n-1*] van *n* getallen die enkel de waarde 1 of 2 hebben, ongesorteerd

Voorbeeld: A=[1, 2, 2, 2, 1, 1, 2]

Uitvoer: array A gesorteerd in oplopende volgorde

1. k = 0
2. for i = 0 to n-1:

if A[i] = 1, then k++

1. for i = 0 to k-1:

A[i] = 1

1. for i = k1 to n-1:

A[i] = 2

3p a Leg uit hoe het SorteerSuperSimpel-algoritme werkt.

2p b Wat weet je van de looptijd van dit algoritme?

5p **Opgave 6**

Bekijk de volgende instructies, die bedoeld zijn om een lijst met getallen te sorteren (van klein naar groot).

1. Vergelijk het eerste en tweede getal in de lijst.
2. Als het eerste getal groter is dan het tweede, verwissel dan het eerste en tweede getal in de lijst. Als het eerste getal niet groter is dan het tweede getal, laat beide getallen dan gewoon op hun plaats staan.
3. Herhaal stap 1 en 2 voor het tweede en derde getal, voor het derde en vierde getal, enz. totdat je het eind van de lijst hebt bereikt.

Laat zien wat er gebeurt als je deze instructies uitvoert op een lijst met de

getallen 31, 4, 15, 9, 27 en 18 (in die volgorde). Is de lijst compleet gesorteerd nadat de instructies zijn uitgevoerd? Zo niet, hoe zou je de instructies aanpassen zodat elke willekeurige lijst na het uitvoeren van de instructies gesorteerd is?

**Opgave 7**

3p a. Wat wordt bedoeld met de term ‘recursie’?

3p b. ‘Torens van Hanoi’ is een voorbeeld van een recursief algoritme. Leg dit uit.

2p c. Stel je hebt een spel ‘Torens van Hanoi’ met n schijven. Hoeveel schijf-

verplaatsingen (zetten) zijn nodig om dit spel op te lossen?

6p **Opgave 8**

Fibonacci-reeks: dit is een rij getallen waarbij de eerste 2 elementen 1 zijn, en de vogende getallen zijn steeds de som van de twee eerdere:

1, 1, 2, 3, 5, 8, 13, 21, 34 enz.

Geef een recursief algoritme om het n-de Fibonacci-getal te berekenen.

15p **Opgave 9**

Een vriend van je heeft een baan als krantenbezorger. Hij heeft veel vaste klanten vaak met verschillende bezorgwensen: Sommigen laten hun krant bij een ander adres bezorgen als ze op vakantie zijn, sommigen krijgen alleen op zaterdag een krant en er zijn adressen met een proefabonnement. Omdat hij per krant wordt betaald en niet per uur vraagt hij jouw hulp voor het bepalen van een optimale route.

2p a. Welk vergelijkbaar probleem hebben we in de les behandeld?

Licht je antwoord toe.

2p b. Voor het probleem bestaat een brute force algoritme. Wat betekent brute force?

2p c. Noem een voordeel van brute force. Leg uit waarom het gebruik van een brute

force algoritme niet handig is om de optimale route te bepalen.

Gegeven zijn alle huizen waar een krant bezorgt moet worden, en alle verbindingen hier tussen. De krantenbezorger is op zoek naar de kortste afstand dat hij moet afleggen.

5p d. Specificeer een (benaderend) algoritme om optimale route te bepalen.

2p e. Leg uit waarom jouw oplossing handig is voor de krantenbezorger. Hoe goed is

je oplossing? Wat zijn de voor- en nadelen ervan?

**Z.O.Z. VOOR DE LAATSTE VRAAG**

4p **Opgave 10**

Bekijk onderstaande graaf, met verschillende knooppunten en verbindingen.

****

Teken een kortste boom voor deze graaf (je mag het algoritme van Prim of dat

van Kruskal gebruiken).

**EINDE VAN DEZE TOETS**

|  |  |  |
| --- | --- | --- |
| **Toets informatica** | **VWO 6** | **PTA 13** |
| **Algoritmiek** | **Antwoorden** | **10 april 2018** |

6p **Opgave 1**

Selection sort:

1. Zoek de kleinste kaart. Verwissel hem met de eerste kaart.
2. Zoek de één-na-kleinste kaart. Verwissel deze met de tweede kaart.
3. Zoek de ‘dan-kleinste’ kaart. Verwissel deze met de derde kaart.
4. Enz. tot hele array gesorteerd is.

13 – 8 – 5 – 4 – 12

kleinste = 13

Vergelijk 13 met 8, kleinste wordt 8

Vergelijk 8 met 5, kleinste wordt 5

Vergelijk 5 met 4, kleinste wordt 4

Vergelijk 4 met 12

Verwissel 4 met 13: rij is nu: 4 – 8 – 5 – 13 – 12

Kleinste = 8

Vergelijk 8 met 5, kleinste = 5

Vergelijk 5 met 13

Vergelijk 5 met 12

Verwissel 5 met 8, rij is nu: 4 – 5 – 8 – 13 – 12

Kleinste = 8

Vergelijk 8 met 13

Vergelijk 8 met 12

Kleinste = 13

Vergelijk 13 met 12, kleinste = 12

Verwissel 12 met 13, rij is nu: 4 – 5 – 8 – 12 - 13

6p **Opgave 2**

a i = 7; som1= - 8; som2 = 26

b i = 4; som1= - 3; som2 = 15

**Opgave 3**

4p **b)** Het algoritme geeft een antwoord maar dat hoeft niet juist te zijn;

Het array is ongesorteerd, stel dat je in de eerste stap een getal in het midden aantreft dat kleiner is dan q. Dan ga je rechts van het midden zoeken en misschien vind je daar q niet. Dan zal het antwoord zijn ‘niet aanwezig’. Dit kan onjuist zijn, q zou ook links van het midden kunnen staan (array is immers ongesorteerd).

Uitleg: 2 pt

Als bij uitleg staat dat een verkeerd getal wordt teruggegeven: -1 punt

4p **Opgave 4**

Schrijf een op binair zoeken gebaseerd algoritme (kies zelf een naam) dat er voor zorgt dat uit een array *A*[1..*n*] van *n* gesorteerde getallen het kleinste positieve getal gerapporteerd wordt. Ga er vanuit dat het array gesorteerd is van klein naar groot, dat er zowel positieve als negatieve getallen in staan en dat er geen dubbele waarden in voorkomen.

Algoritme BinairZoeken(A)

Invoer: Een array A[1..n] met positieve en negatieve getallen, gesorteerd van klein naar groot..

Uitvoer: Een getal A[i] = q, waarbij q

het kleinste positieve getal is.. 1p

1. links = 1; rechts = n 2p

2. zolang links <= rechts 1p

doe: mid = └(links + rechts)/2┘ 1p

als A[mid] <= 0 1p

dan links = mid; 1p

anders rechts = mid; 1p

1. rapporteer A[mid];

Andere optie bij 2:

zolang links <= rechts

doe: mid = └(links + rechts)/2┘

als A[mid] > 0

dan als A[mid-1] <= 0

dan Rapporteer A[mid] en stop

anders rechts = mid – 1

anders links = mid +1;

1. *als een antwoord niet in pseudo-code wordt gegeven, maar als enkel het idee in gewoon Nederlands wordt beschreven: max. 4 punten*
2. *pseudo-code waaruit blijkt dat idee van ‘binair zoeken’ goed wordt toegepast: 4 pt*
3. *pseudo-code: 8 pt (met -1 pt voor fouten) maar genoemde bij 2. blijft gelden*

5p **Opgave 5**

SorteerSuperSimpel: eerst tel je hoe vaak een element de waarde 1 heeft (hoeveel enen er in het array voorkomen). Dat houdt je bij in de variabele k.

Daarna geef je de eerste k elementen de waarde 1 en de rest (k+1 tot n) geef je de waarde 2.

Looptijd: eerste for-loop gaat 1 keer door array: n stappen; tweede en derde loop doen samen ook n herhalingen; dus 2n stappen

5p **Opgave 6**

Deze instructies vormen het bubble sort algoritme.

Na het uitvoeren van deze instructies is de volgorde in de lijst:

4, 15, 9, 27, 18, 31

Nu staat het grootste element helemaal rechts, maar de rest van de lijst is nog niet gesorteerd. Dus bij n elementen moet je dit n keer herhalen, waarbij je rechts steeds 1 element minder ‘mee hoeft te laten doen’.

8p **Opgave 7**

Recursie: probleem oplossen door een ‘kleiner’ gelijk probleem op te lossen, totdat probleem zo klein is dat je het direct kunt oplossen.

Recursie: het optreden van een constructie als onderdeel van zichzelf.

Recursieve functie: functie die zichzelf aanroept.

Om ‘Torens van Hanoi’ op te lossen voor n schijven waarbij de stapel van pin A naar pin C moet doe je achtereenvolgens:

* Torens van Hanoi voor n-1 schijven, van A naar pin B
* Je verplaatst de onderste schijf van A naar C
* Torens van Hanoi voor n-1 schijven, van B naar C

Aantal verplaatsingen: 2ˆn – 1

6p **Opgave 8**

Fibonacci-reeks:

Als n = 1 of n = 2, dan F(n) = 1 (2p)

Anders: F(n) = F(n-1) + F(n-2) (4p)

**Opgave 9**

1. Travelling salesman problem. Zoeken naar optimale rondrit.
2. Brute force: alles uitrekenen om antwoord te bepalen.
3. Brute force: je weet zeker dat je het juiste antwoord vindt. Maar bij dit probleem duurt het te lang om deze methode te gebruiken.
4. Zoek dichtstbijzijnde buurman die nog niet bezocht is
5. Werkt redelijk goed, maar in sommige gevallen vind je geen optimale route

4p **Opgave 10**



Kortste boom