# Tentamen Geometrische Algoritmen, 29 januari 2004, 9–12 uur

Voor dit tentamen zijn 10 punten te halen. Lees elke opgave zorgvuldig door voordat je 'm beantwoordt; zorg dat je begrijpt wat er gevraagd wordt. Controleer achteraf of je inderdaad de vraag hebt beantwoord. Maak eerst de opgaven die je makkelijk vindt, en daarna pas de moeilijkere. Het tentamen is gesloten boek.

Je krijgt het eerste punt kado als je je naam op alle ingeleverde vellen zet, op de eerste ook je collegekaartnummer, en als je voldoende netjes schrijft.

1. (1 punt) Leg duidelijk uit hoe een segmentboom voor een verzameling $I$ van intervallen deze intervallen opslaat.

2. (1 punt) Gegeven een driehoek met vertices $(0,0)$, $(3,0)$, $(0,1)$ en een vierhoek met vertices $(0,0)$, $(4,0)$, $(2,1)$, en $(1,3)$ (tegen de klok in gegeven). Waar liggen de vertices van de Minkowski sum van de driehoek en het vierkant? Geef de coördinaten tegen de klok in.

3. (1 punt; -0.5 punt per fout of niet ingevuld antwoord) Geef de bouwtijd, het geheugengebruik en de zoektijd van de volgende datastructuren. Druk de tijd uit in $n$, het aantal opgeslagen objecten, en $k$, het aantal gerapporteerde antwoorden.

   (a.) intervalboom

   (b.) 2-dimensionale kd-boom

   (c.) $d$-dimensionale rangeboom

4. (1 punt) Gedegenereerde gevallen kunnen vervelend zijn voor een algoritme. In de hoofdstukken van het boek worden twee technieken behandeld om hiermee om te gaan: de *shear transformation* (Ch 6, point location) en *composite numbers* (Ch 5, range searching). Leg van één van deze twee technieken uit hoe die werkt. Een antwoord van 5 à 10 regels volstaat.

5. (1 punt) Gegeven een verzameling $G$ met groene punten, een verzameling $B$ met blauwe punten en een verzameling $R$ met rode punten. We willen weten of er een punt $p$ is vanwaaruit 3 half-lijnen getrokken kunnen worden, zodanig dat $G$, $B$ en $R$ volledig in de drie verschillende faces liggen. Je mag hierbij aannemen dat de drie faces waarin het vlak wordt verdeeld door de half-lijnen convex zijn.

   Specificeer dit probleem exact in de duale setting.

**Z.O.Z.**

6. (2 punten) We willen het volgende probleem oplossen met een sweep algoritme: Gegeven een verzameling $S$ met $n$ niet-snijdende lijnstukken in het platte vlak. De lijnstukken hebben ook niet dezelfde eindpunten. Gegeven verder een punt $p$ dat niet op enig lijnstuk van $S$ ligt. We willen alle lijnstukken vinden waarvoor $p$ *beide* eindpunten kan zien. Punt $p$ ziet een eindpunt $q$ van een lijnstuk als het lijnstuk $\overline{pq}$ dat $p$ en $q$ verbindt geen enkel punt bevat van de lijnstukken uit $S$, met uitzondering van de eindpunten uit $S$. (Intuitief: $p$ kan "door" een eindpunt van een lijnstuk een verder gelegen eindpunt zien.)

   Geef een sweep algoritme voor dit probleem, waarbij een half-lijn beginnend bij $p$ 360 graden draait om $p$. Definieer de status, beschrijf de statusstructuur, beschrijf de eventafhandeling, en vergeet niet de gewenste lijnstukken te rapporteren in je algoritme.

   (Indien je alle lijnstukken rapporteert waarvan minstens één eindpunt zichtbaar is dan krijg je 1 punt voor deze opgave.)

7. (2 punten) In hoofdstuk 6 werd een point location datastructuur gebouwd voor een subdivisie $S$ bestaande uit lijnstukken. In deze opgave willen we weten of de oplossing ook werkt voor een verzameling $C$ met $n$ niet-snijdende cirkelbogen. Een cirkelboog is gedeelte van een cirkel en heeft precies twee eindpunten.

   (a.) In hoofdstuk 6 werd de trapezoidale decompositie (vertikale decompositie) gebruikt in de oplossing. Hoe zou je die willen definiëren voor de verzameling $C$ van niet-snijdende cirkelbogen? Teken een voorbeeld met 3 cirkelbogen waarvan minstens eentje groter is dan een halve cirkel.

   (b.) Ga na of de zoekstructuur zoals gebruikt in hoofdstuk 6 nog werkt. Geef eventueel de aanpassingen die nodig zijn om de zoekstructuur goed te laten werken.

   (c.) Leg vervolgens uit of een randomized incremental construction algoritme voor het bouwen van een point location structuur op $C$ en de decompositie daarvoor uit (a.) nog werkt, en nog steeds een verwacht geheugengebruik van $O(n)$ en een verwachte zoektijd van $O(\log n)$ geeft voor point location.