Tentamen Geometrische Algoritmen, 25 oktober 2001, 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 antwoord. Maak eerst de opgaven die je makkelijk vindt, en daarna pas de moeilijkere. Het tentamen is gesloten boek.

1. **(1 punt)**
   Geef een algoritme dat voor een meegegeven face f in een planaire subdivisie de coördinaten van de hoogste vertex in de rand van f bepaalt (als er meerdere hoogste vertices zijn mag er willekeurig een van deze vertices gerapporteerd worden). De subdivisie is opgeslagen als doubly-connected edge list. Gebruik de standaardnamen van de velden van de doubly-connected edge list in je algoritme. Je algoritme moet correct zijn voor elke face van de subdivisie. Het algoritme heeft dus HoogsteVertex(face-object f) als heading.

2. Gegeven een verzameling D van n driehoeken in het platte vlak, en neem aan dat de driehoeken niet snijden of binnen een andere driehoek liggen. We beschouwen een sweep line algoritme dat in $O(n \log n)$ tijd een verzameling met $n-1$ lijnstukken bepaalt zodanig dat:

   - elk lijnstuk tussen twee hoekpunten van twee driehoeken loopt;
   - de lijnstukken niet snijden, behalve mogelijk bij een gezamenlijk eindpunt;
   - de lijnstukken gezamenlijk alle driehoeken met elkaar verbinden.

   (a.) **(1 punt)**
   Beschrijf de statusstructuur die je gebruikt in het algoritme, met alle opgeslagen informatie.

   (b.) **(1 punt)**
   Geef aan wanneer de events optreden, en beschrijf precies hoe de eventafhandeling plaatsvindt.

   (c.) **(0.5 punt)**
   Bewijs de tijdgrens van het algoritme.

3. **(1 punt)**
   Bij lineair programmeren met een verzameling half-vlakken H beschouwen we het randomized incremental construction algoritme. Neem aan dat de doelfunctie vertikaal omlaag is (kleinst mogelijke y-coördinaat), en bij gelijke y-coördinaten de kleinste x-coördinaat. Neem aan dat twee half-vlakken h_1 en h_2 de oplossing begrenzen, en $h_3, \ldots, h_n$ in random volgorde staan.

   Als we backwards analysis gebruiken om de verwachte tijd voor de toevoeging van h_i te bepalen, gaan we uit van de situatie na het toevoegen van dat half-vlak h_i, begrensd door een lijn ℓ_i. We hebben dan een optimum punt p_i. Omdat de toevoegvolgorde random was, heeft elk half-vlak uit $\{h_3, \ldots, h_i\}$ dezelfde kans om als laatste te zijn toegevoegd.

   Ga zorgvuldig na hoe groot de kans is dat de laatste toevoeging het optimum op p_i heeft gezet. Behandel daarbij zowel het algemene geval als de gedegenereerde gevallen.

1

4. Gegeven is een verzameling P met n punten en een verzameling L met m lijnen (met willekeurige oriëntaties). We willen graag weten of er (minstens) een punt uit P op een lijn uit L ligt. Een triviale oplossing voor het probleem is voor elk punt $p \in P$ en elke lijn $l \in L$ te controleren of $p \in l$. Zo ja, dan zijn we klaar; zo nee, dan testen we het volgende paar. Deze oplossing kost natuurlijk $O(nm)$ tijd.

   (a.) **(1 punt)**
   Stel je voor we kiezen de volgende aanpak: (i) Bouw een kd-boom om de punten uit P op te slaan. (ii) Zoek met elke lijn uit L om te bepalen of er een punt uit P op die lijn ligt.

   Het zoekalgoritme voor een willekeurige lijn in een kd-boom is geheel analoog aan de het algoritme voor zoeken met een as-parallelle rechthoek: bij elke knoop die we tegenkomen en waarvoor de zoeklijn ℓ het bijbehorende gebied snijdt, gaan we in beide kinderen recursief verder zoeken. Snijdt de lijn het bijbehorende gebied niet, dan stopt het algoritme in die knoop. Bij elk blad waarin we uitkomen testen we of het opgeslagen punt op de zoeklijn ℓ ligt.

   Hoeveel tijd kost dit algoritme in het slechtste geval? (N.B. Zorg dat je antwoord ook correct is als n heel klein is ten opzichte van m, of andersom.)

   (b.) **(1 punt)**
   Stel je voor we kiezen als oplossing de volgende aanpak: Bouw een arrangement van de lijnen uit L. Hiermee krijgen we een planaire subdivisie bestaande uit vertices, edges, en faces. Bouw een point location structuur op alle edges van dit arrangement. Ga vervolgens met elk punt uit P zoeken in de point location structuur. Als dat punt op een edge en dus op een lijn uit L ligt, dan ontdekken we dat tijdens zo'n point location zoekactie.

   Wat is de verwachte tijd in het slechtste geval voor deze aanpak? (N.B. Zorg dat je antwoord ook correct is als n heel klein is ten opzichte van m, of andersom.)

   (c.) **(1 punt)**
   Stel we weten dat $m = \Theta(n^2)$, dus we hebben aanzienlijk meer lijnen dan punten. Geef nu een zo efficiënt mogelijke oplossing. Schets de oplossing en geef de tijdgrens, uitgedrukt in n.

5. Laat R een verzameling van n as-parallelle rechthoeken in het platte vlak zijn. We willen R opslaan voor het volgende type zoekacties: Gegeven een query punt q, rapporteer alle rechthoeken van R die q bevatten. We willen een datastructuur die efficiënt in geheugenruimte en zoektijd is.

   (a.) **(1 punt)**
   Beschrijf een efficiënte datastructuur die R opslaat voor het genoemde type zoekactie. Neem als hoofdboom een segmentboom, en gebruik het begrip "kanonieke deelverzameling" in je beschrijving.

   (b.) **(0.5 punt)**
   Geef van de datastructuur uit (a.) aan hoeveel geheugen deze kost.

   (c.) **(1 punt)**
   Geef het zoekalgoritme en de tijd nodig voor de genoemde zoekactie.

2