Skriftlig Eksamen
Geometriske Beregninger (DM45)

Institut for Matematik og Datalogi
Syddansk Universitet, Odense

Torsdag den 21. januar 1999, kl. 9–13

Alle sædvanlige hjælpemidler (lærebøger, notater, etc.) samt brug af lommeregner er tilladt.

Eksamenssættet består af 4 opgaver på 6 nummererede sider (1–6). Fuld besvarelse er besvarelse af alle 4 opgaver.

De enkelte opgavers vægt ved bedømmelsen er angivet i procent. Der må gerne refereres til algoritmer og resultater fra lærebogen inklusive øvelsesopgaverne. Henvisninger til andre bøger (udover lærebogen) accepteres ikke som besvarelse af et spørgsmål.

Bemærk, at hvis der er et spørgsmål i en opgave, man ikke kan besvare, må man gerne (så vidt det er muligt) besvare de efterfølgende spørgsmål og blot antage, at man har en løsning til de foregående spørgsmål.

Man behøver ikke specificere, hvordan simple (gymnasiepensum) geometriske konstant-tids-beregninger foretages. Man må altså f.eks. gerne antage, at man har følgende funktioner til rådighed:

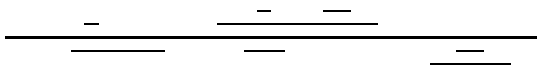
- Skærer 2 givne liniestykker?
- Beregn et evt. skæringspunkt for 2 givne liniestykker.
- Beregn en given linies vinkel med x-aksen.
- Ligger et givet punkt til højre for et givet orienteret liniestykke?

Opgave 1 (25%)

I denne opgave skal der laves to $O(n \log n)$-algoritmer vha. line-sweep teknikken. Det er ikke nødvendigt at skrive kode eller algoritmestumper, hvis man på tekstform kan forklare tilstrækkeligt præcist, hvad der skal foretages.

Spørgsmål a: Vi har givet n intervaller $I_1, I_2, \ldots, I_n$. Hvert interval er givet som $[x_1 : x_2]$, hvor $x_1 < x_2$. Man kan antage, at der iblandt de $2n$ endepunkter ikke optræder to, der er ens.

Man må udnytte følgende restriktion på input: Hvis to intervaller I_i og I_j overlapper, så er enten I_i indeholdt i I_j eller I_j er indeholdt i I_i. Et typisk input kunne se således ud (som i lærebogen tegner vi intervallerne lidt forskudt, så man nemmere kan se, hvor de starter og ender):



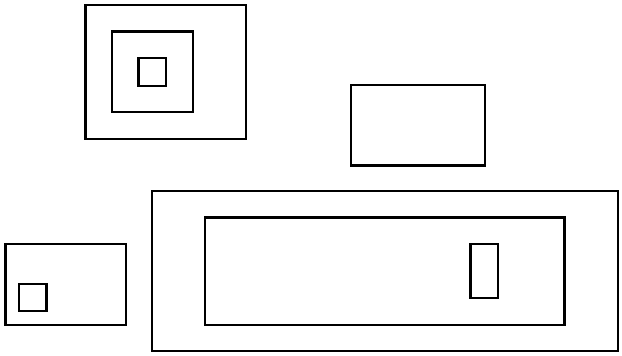
En sekvens af intervaller $I_{i_1}, \ldots, I_{i_k}$ kaldes en *én-dimensionel Matryoshka*, hvis $I_{i_1} \subset I_{i_2} \subset \cdots \subset I_{i_{k-1}} \subset I_{i_k}$. Størrelsen af denne er k. Rækkefølgen af intervallerne i sekvensen behøver ikke være den samme som i de n input-intervaller.

I eksemplet er den største Matryoshka af størrelse 4.

Beskriv en algoritme, der kan finde størrelsen af den største én-dimensionelle Matryoshka. Argumentér for, at tidskompleksiteten bliver $O(n \log n)$. □

Spørgsmål b: Vi har givet n akse-parallelle rektangler $R_1, R_2, \ldots, R_n$ specificeret ved koordinaterne for deres syd-vestlige hjørne samt højde og bredde.

Man må udnytte følgende restriktion på input: Ingen rektangler skærer eller rører hinandens omkreds, men rektangler kan godt ligge helt indeholdt i et eller flere andre rektangler. Dog er det sådan, at hvis to rektangler R_i og R_j begge er indeholdt i et tredje rektangel, så er enten R_i indeholdt i R_j eller R_j er indeholdt i R_i. Nedenfor ses et typisk input:



En sekvens af rektangler $R_{i_1}, \ldots, R_{i_k}$ kaldes en *to-dimensionel Matryoshka*, hvis $R_{i_1} \subset R_{i_2} \subset \cdots \subset R_{i_{k-1}} \subset R_{i_k}$. Størrelsen af denne er k. Rækkefølgen af rektanglerne i sekvensen behøver ikke være den samme som i de n input-rektangler.

I eksemplet er den største Matryoshka af størrelse 3.

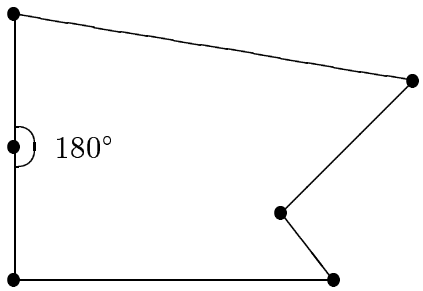
Beskriv en algoritme, der finder størrelsen af den største to-dimensionelle Matryoshka. Argumentér for, at tidskompleksiteten bliver $O(n \log n)$. □

Opgave 2 (25%)

Opgaven handler om at forbinde n punkter på forskellig måde.

Husk, at en polygonvej er en sekvens af liniestykker, hvor slutpunktet for et liniestykke er startpunktet for det næste. F.eks. er en polygon, hvor man fjerner en kant, en polygonvej.

Husk, at der gælder for både polygoner og polygonveje, at de ikke må krydse sig selv. Til gengæld accepterer vi, at polygoner og polygonveje har vinkler på 180°. Dvs. at der godt må ligge et hjørne på en lige linie mellem to andre hjørner. Følgende er altså en polygon:



Der er ingen restriktioner på de n punkter. F.eks. kan adskillige punkter have samme x-koordinat. De n punkter er dog alle forskellige.

Spørgsmål a: Givet n punkter i planen, beskriv en algoritme, der i tid $O(n \log n)$ forbinder punkterne til en polygonvej.

Algoritmen kan og skal laves, uden der anvendes egentlige beregninger på input. Sammenligninger er tilladte. □

Spørgsmål b: Givet n punkter i planen, beskriv en algoritme, der i tid $O(n \log n)$ forbinder punkterne til et polygon.

Der er ingen yderligere krav til algoritmen i dette spørgsmål. □

Spørgsmål c: Kan enhver polygonvej laves ud fra en polygon ved at fjerne en kant? □

Opgave 3 (30%)

Det er kendt fra kurset, at hvis man tager fællesmængden af n halvplaner, så får man en konveks mængde. Denne mængde kan være ubegrænset i nogle retninger. For at undgå dette anvender vi et trick, der minder om den velkendte "bounding box". Vi definerer en start-trekant, der er stor nok til, at alt, hvad vi er interesseret i, vil komme til at ligge i dens indre. Derved kommer der i den inkrementelle algoritme beskrevet nedenfor til at gælde, at det delsvar (P i algoritmen), vi har på et givet tidspunkt, altid er en polygon. Trekanten skal ikke konstrueres, men er givet som en del af input.

Vi definerer nu et punkt p i planen til at være i *konflikt* med et halvplan h, hvis p ligger i det indre af $\overline{h}$, hvor $\overline{h}$ betegner komplementet til h.

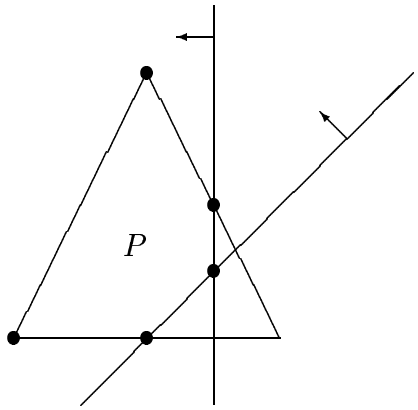
I den repræsentation, vi laver for halvplaner h, afsætter vi et felt $h.k$ til et konfliktpunkt. I den repræsentation, vi laver for punkterne p i polygonen, afsætter vi et felt $p.hl$ til en liste af halvplaner.

Følgende skal være en invariant for **while**-løkken nedenfor:

- P er fællesmængden af den givne trekant og alle behandlede halvplaner.
- For alle ubehandlede halvplaner h skal $h.k$ være et hjørne i den aktuelle polygon P, der er i konflikt med h.
- For alle hjørner p i P skal $p.hl$ indeholde alle de ubehandlede halvplaner h, der har $h.k = p$.

algoritme: HALVPLANSFÆLLESMÆNGDE
P = den givne trekant
behandl alle halvplaner, der indeholder P
initialisér halvplanernes konfliktpunkter og polygonhjørnernes halvplanslister
while der er ubehandlede halvplaner **do**
    tag næste ubehandlede halvplan h
    opdatér P, konfliktpunkter og halvplanslister
    behandl alle halvplaner, der indeholder P
output P

Nedenfor ses som et eksempel en initiel trekant, samt P – en konveks polygon med fem hjørner – efter de to angivne halvplaner er blevet behandlet:



Når man behandler et halvplan h, og et hjørne p i polygonen ligger i $\overline{h}$, så forsvinder det polygonhjørne naturligvis, når P opdateres. Vi siger, at et ubehandlet halvplan h', der har $h'.k = p$, får sit konfliktfelt *invalideret* (da det skal opdateres, for at invarianten kan være opfyldt). Vi lader t betegne det totale antal invalideringer, der foretages gennem afviklingen af algoritmen.

Man kan antage, at input er i generel position (general position). Dvs. at man kan ignorere specialtilfælde.

Spørgsmål a: Forklar, hvordan de uspecificerede dele af algoritmen kan realiseres, så algoritmen kan afvikles i tid $O(n + t)$. □

Spørgsmål b: Vis, ved at konstruere et eksempel, at man kan komme til at opdatere konfliktpunkterne på en sådan måde, at $t \in \Omega(n^2)$. □

Vi risikerer altså en køretid på $\Theta(n^2)$.

Spørgsmål c: Diskutér (gerne ud fra dit eksempel i foregående spørgsmål), hvordan en indledende randomisering af halvplanerne kunne give en bedre forventet kompleksitet. Hvilken kompleksitet, ville du tro, man kunne få? □

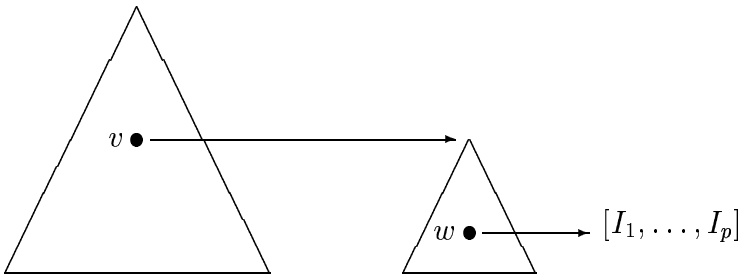
Opgave 4 (20%)

Vi har givet n akse-parallelle rektangler i planen. De skal arrangeres i en datastruktur, sådan at vi givet et forespørgselspunkt q kan rapportere alle de rektangler, der indeholder q.

Vi bestemmer os for at bruge en sammensat struktur baseret på segmenttræer, hvor vi separat bruger rektanglernes x- og y-intervaller. Sammen med ethvert interval, vi gemmer i strukturen, gemmer vi også information om, hvilket rektangel det kommer fra.

Vi beskriver nu den sammensatte struktur: På yderste niveau har vi et segmenttræ på rektanglernes x-intervaller. Hvis v er en knude i segmenttræet på yderste niveau, betegner $R(v)$ den kanoniske delmængde (canonical subset). Til enhver sådan knude v associeres et segmenttræ på y-intervallerne for rektanglerne i $R(v)$. Den kanoniske delmængde for en knude w i et segmenttræ på inderste niveau er blot lagret i en liste.

Opbygningen er illustreret nedenfor:



Spørgsmål a: Forklar (evt. vha. en algoritme og/eller tegninger), hvordan en forespørgsel besvares vha. denne struktur. □

Spørgsmål b: Hvad er tidskompleksiteten for en forespørgsel? Hvad er strukturens pladsforbrug? Argumentér for dine svar. □