Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström Göteborg 6 nov 2018

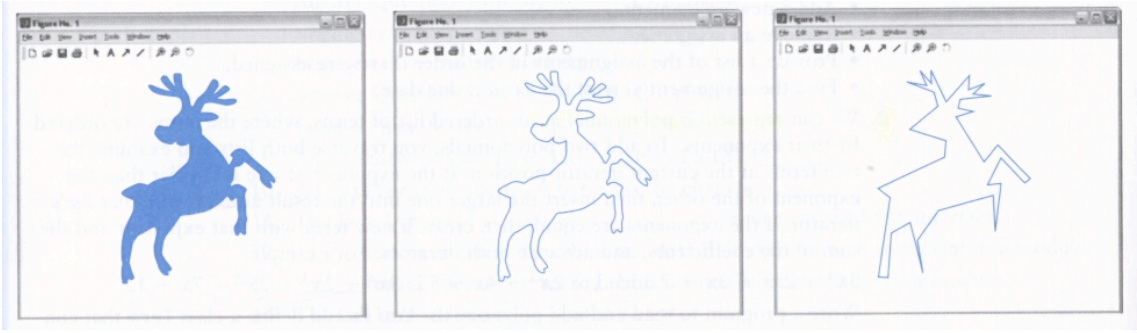
Lab 1 Uppg 2 Testar: länkade listor, komplexitet.

Vi går igenom länkade listor i LV2 så har du inte redan koll på dom så vänta till efter den föreläsningen med att lösa denna uppgiften.

Inlämning: Svaren på frågorna i a och c (märkta med ”*”) samt metoderna `addLast` och `reduceListToKElements`, allt kan läggas i samma pdf fil som döps till ”lab1.2.pdf”.

Detta är en gammal tentauppgift som jag lånade från boken. Så först en fri översättning av uppgift 2.7 från Koffman&Wolfgang, uppgifterna kommer efter den.

En 2 dimensionell form (shape) kan definieras av sin gränslinje-polygon, som helt enkelt är en ordnad lista med alla koordinater (dvs punkter), ordnad genom att traversera* dess kontur. Den vänstra figuren nedan visar originalet, figuren i mitten visar konturen av formen och den högra bilden visar en abstrakt gränslinje som bara tar med de viktigaste linjerna.

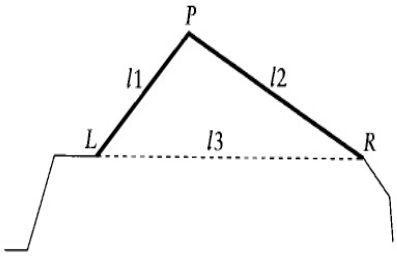


Vi kan ge varje punkt, P, ett värdemått på hur viktig den är i gränslinjen genom att titta på dess grannar L och R. Vi beräknar distanserna LP, PR och LR, kallar dem *l1*, *l2* och *l3,* se figur nedan. Definiera sedan värdemåttet som *l1+l2-l3*.

Använd sedan följande algoritm för att beräkna de *k* viktigaste punkterna:

1. while antalet punkter är större än *k*
2. beräkna om värdet av varje punkt
3. tag bort den minst betydelsefulla

Skriv ett program som läser en mängd koordinater som formar en kontur och reducerar listan till *k* punkter.



Så långt den fria översättningen från boken.

Uppgiften går alltså ut på att förenkla 2-dimensionella konturer. En kontur representeras av en punklista som utgörs av en lista av talpar, där varje talpar motsvarar x- och y-koordinaterna för en punkt på konturen. Punkterna är ordnade i listan så att de följer konturen från en start- till en slutpunkt. Ert program ska förenkla en kontur genom att ta bort de minst viktiga punkterna, en i taget, på lite olika sätt.

Man inser lätt att om L, P och R ligger på en linje så är P oviktig (och värdemåttet=0) och ju spetsigare vinkeln vid P är ju viktigare är P dvs värdemåttet är alltid icke-negativt och är det 0 så märks det inte alls om man tar bort P .

Vi gör en förenkling; listan med punkter är inte sluten dvs det finns en första och en sista punkt och dessa kan inte tas bort . Lämpligen löser man det genom att låta första och sista

punkten ha värdemåttet oändligheten och sedan inte beräkna om värdet (vilket ju skulle blir svårt eftersom dom saknar en granne). I indatafilen ni får så är oändligheten = 1.0E300.

När du funderat lite på den här uppgiften så inser snart att algoritmen ovan är onödigt kostsam pga punkt 2. En punkts värde beror ju bara på dess grannars läge, inte alla punkters, så när vi tar bort en punkt så behöver vi inte räkna om alla punkternas värden. Vi borde kunna använda följande algoritm:

```
1 beräkna värdet av varje punkt
2 while antalet punkter är större än k
3   tag bort den minst betydelsefulla
4   beräkna om dess tidigare närmsta grannars värde (2 st)
  end while
```

Uppgifter:

a) Algoritmidén verkar kunna fungera, nu måste du bestämma dig för en representation.
Du bestämmer dig för att använda ett fält (dvs vanlig array) för att lagra punkterna.
* Gör en grov komplexitetsanalys dvs en väl förklarad handviftning.
Du inser snart att använder du fält så innebär det att punkten ”tag bort...” ovan gör att loopens kropp tar väldigt lång tid.
* Förklara varför. Rita gärna figurer. Du kan fota dom för inlämningen.
Du bestämmer dig istället för att använda Javas ”LinkedList” men inser snart att det blir i princip lika illa.
* Förklara kort varför. (Läs API-dokumentationen och fundera över hur man använder listan)

b) Du tänker nu att enda sättet att göra detta effektivt kanske är att skriva kod för en specialiserad dubbellänkad lista själv så det gör du. För att snabbt kunna hitta punkten med minst värde så använder du en prioritetskö (från Javas API) för punkterna och deras värden. Det mesta av koden finns i filen `DLList.java` på hemsidan.
* Du behöver bara skriva metoderna `addLast` och `reduceListToKElements` som finns sist i filen.
(Prioritetskön måste innehålla pekare till noder i den länkade listan för att ni snabbt ska hitta rätta element att ta bort och modiera grannarna för. Tänk också på att javas PriorityQueue inte har något sätt att uppdatera värdet av ett element i listan. Det skulle behövas för när en punkt tagits bort ändras värdemåttet för dess grannar. Man får lösa detta på något bra sätt, beskriv hur du gör.)

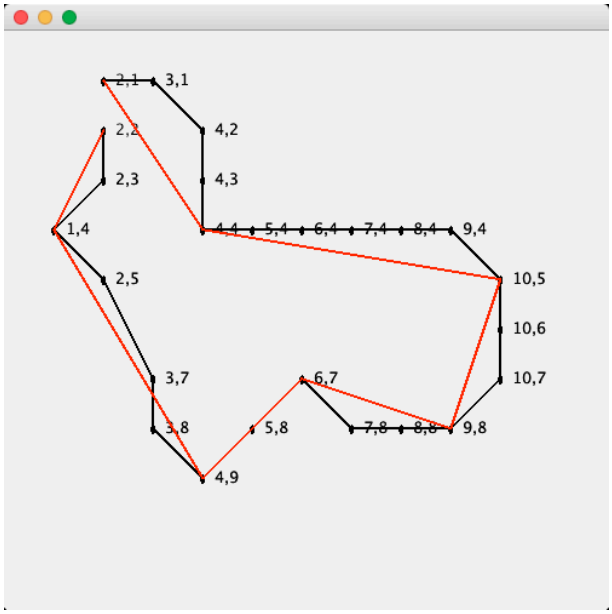
c) * Beräkna nu komplexiteten för din nya algoritm med speciallistan.
* Till din stora besvikelse inser du att även den nya versionen kommer att vara ineffektiv pga hur prioritetskön fungerar. Varför?
I API-dokumentation om klassen `PriorityQueue` (i översta avsnittet, konstruktorerna) kan ni läsa om olika metoders komplexitet.

Det är alltid roligt att kunna provköra och se om algoritmerna man skriver fungerar och hur dom gör det så jag har skrivit en klass (`DrawGraph.java`) som ritar upp ursprungspunkterna (i svart) och ditt resultat (i rött) och som tillsammans med ett huvudprogram i filen `Main.java` testar dina metoder. Det är inte perfekt men funkar hyfsat. Finns på hemsidan.

Man kompilerar klasserna och kör med

```
javac DrawGraph.java
javac DLList.java
javac Main.java
java Main -k8 < fig1.txt
```

Flaggan k säger hur många punkter man skall reducera till och sedan läser jag in en figur från filen fig1.txt. Du kan naturligtvis göra en egen indatafil och gör du en som är bättre än min så skicka den till mej så lägger jag den på hemsidan också. Så här ser det ut när jag kör enligt ovan. Det finns 2 ytterligare indatafiler fig2.txt och fig3.txt på hemsidan.



Det finns också en flagga -h som ger en hjälptext för Main och en -d som ger en del utskrifter i terminalfönstret – bra när det inte fungerar som det skall.

Alla filer finns i en zip-fil på hemsidan. I zip filen finns också artikeln som uppgiften baseras på om någon har lust att lära sig mer men det är inte nödvändigt för uppgiften här.

Maila mig om det är några konstigheter.