Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström Göteborg 5 mar 2010

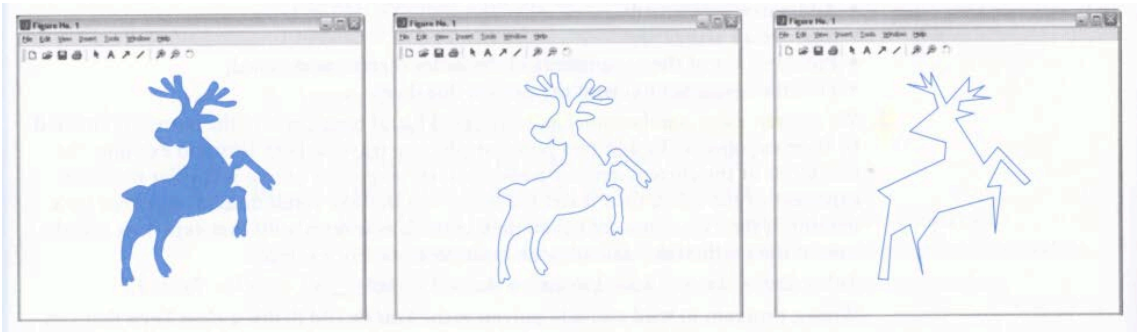
Testar: Fält, länkade listor, komplexitet.

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

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.

Inlämning: Svaren i a och metoderna i `addLast, calcInitialImportance` och `importanceRemoveList` kan läggas i samma fil. Om du vill ha med figurer så skapa en pdf med svaren från a döpt till ”lab1U3a”.

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.



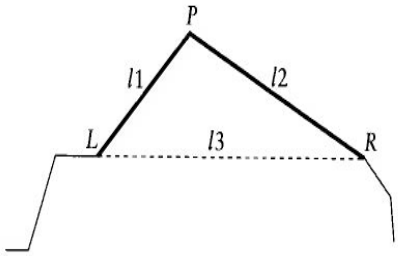
Vi kan ge varje punkt, P, ett värdemått på hur viktig den är i gränslinjen genom att titta på dess grannar L och R. Vi beräknar distanserna LP, PR och LR, kallar dem *l1*, *l2* och *l3,* se figur nedan. Definiera sedan värdemåttet som *l1+l2-l3*.

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

1. while antalet punkter är större än *k*
2. beräkna 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.



Så om L, P och R ligger på en linje så är P oviktig och ju spetsigare vinkeln vid P är ju viktigare är 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 värdet oändligheter och sedan inte beräkna om värdet (vilket ju skulle blir svårt eftersom dom saknar en granne)

Du funderar lite på den här uppgiften och 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)

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. Ganska snart inser du att fält innebär att du måste flytta mycket data under punkt 3 vilket gör att loopens

kropp tar *O(n)* istället för *O(k)* som du hoppats på. 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.

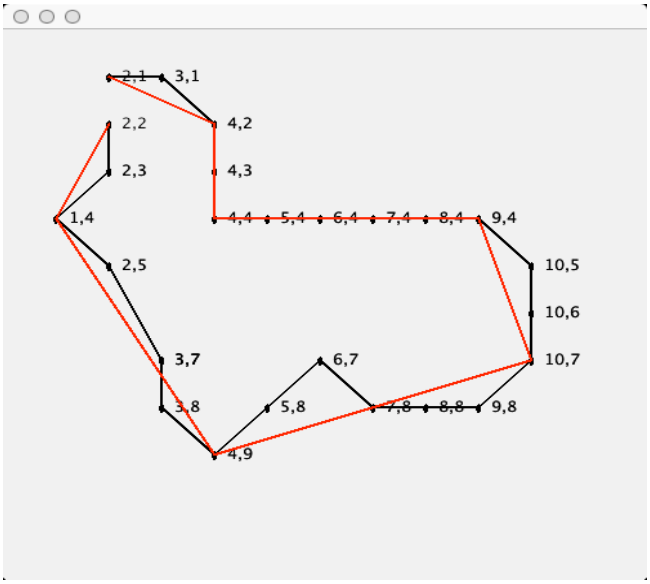
b) Du tänker nu att enda sättet att göra detta effektivt kanske är att skriva kod för en dubbellänkad lista själv så det gör du. För att snabbt kunna hitta punkten med minst värde så använder du en prioritetskö (från Javas API) för punkterna och deras värden. Ett skal finns i `DimpLinkedList`. Du behöver bara skriva metoderna `addLast, calcInitialImportance` och `importanceRemoveList`. De övriga finns färdiga. (`addLast` och `calcInitialImportance` är ganska enkla, `importanceRemoveList` är lite svårare).

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 som ritar upp ursprungspunkterna (i svart) och ditt resultat (i rött) som du kan använda om du vill. Det är inte perfekt men funkar hyfsat. Finns på hemsidan.

Man kompilerar klasserna och kör med

```
javac DrawGraph.java
javac DimpLinkedList.java
javac Main.java
java Main -k8 -w12 -h12 < fig1.txt
```

Flaggan k säger hur många punkter man skall reducera till, w och h är ritytans logiska koordinater dvs i det här fallet kan man ha punkter i intervallen 0..w, 0..h 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



I zip filen med kod 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.

c) (Den sista frågan som var med på tentan kan du kanske inte svara på ännu (så du behöver inte göra det) men återkom hit när vi pratat om prioritetsköer.)
Till din stora besvikelse inser du att även den här versionen kommer att vara ineffektiv pga hur prioritetskön fungerar. Varför?

* ”traversal” refers to the process of visiting (examining and/or updating) each node in a data structure, exactly once, in a systematic way.