TDDD86 – Laboration #7

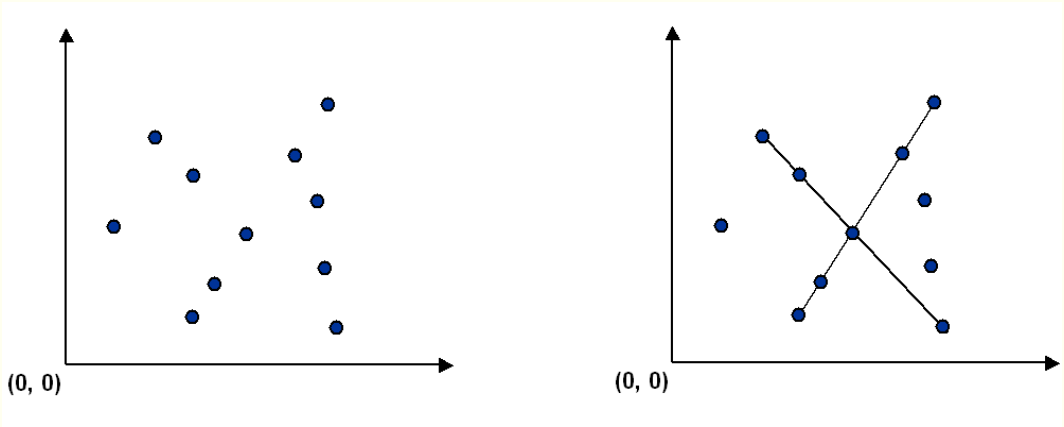
23 november 2022

I den här uppgiften får du använda sortering för att konstruera ett effektivt program som känner igen linjemönster i en given mängd av punkter. Filerna du behöver för att komma igång finns som `labb7.tar.gz` på kurshemsidan.

Redovisning: Efter att du redovisat muntligt, gör en `git commit -m ''TDDD86 Lab 7 redovisning''` och en `git push`. Se till att filerna **fast.cpp** och **readme.txt** är med. Informera sedan din assistant genom att maila honom/henne.

Mönsterigenkänning

Givet en mängd av `N` distinkta punkter i planet, hitta alla (maximala) linjesegment som innehåller en delmängd av 4 eller fler av punkterna.



Tillämpningar

Viktiga komponenter i datorseende är att använda mönsteranalys av bilder för att rekonstruera de verkliga objekt som genererat bilderna. Denna process delas ofta upp i två faser: *feature detection* och *pattern recognition*. I *feature detection* väljs viktiga områden hos bilden ut; i *pattern recognition* försöker man känna igen mönster i områdena. Här får du chansen att undersöka ett särskilt rent mönsterigenkänningsproblem rörande punkter och linjesegment. Den här typen av mönsterigenkänning dyker upp i många andra tillämpningar som t.ex. statistisk dataanalys.

Punktdatatyp

Bland de tillhandahållna filerna finns datatypen `Point` som representerar en punkt i planet genom att implementera följande API (utvalda delar):

```
// konstruera punkten (x, y)
Point(unsigned int _x, unsigned int _y);
// lutningen mellan this och punkten that
double slopeTo(const Point& that) const;
// rita ut punkten på ytan s
void draw(QGraphicsScene* s) const;
// rita linjen mellan this och punkten that på s
void lineTo(QGraphicsScene* s, const Point& that) const;
// jämför punkterna p1 och p2 lexikografiskt
friend bool operator<(const Point& p1, const Point& p2);
```

Totalsökning

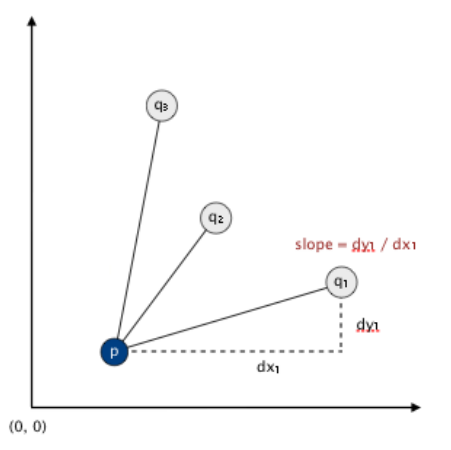
Programmet `brute.cpp` finns också bland de tillhandahållna filerna. Dessutom finns en mängd testdata i resurskatalogen. `brute.cpp` läser in punkter från en fil och ritar upp dessa. Sedan söker programmet igenom alla kombinationer av 4 punkter och kontrollerar om de ligger på samma linjesegment. Om en sådan kombination hittas ritas en linje mellan ändpunkterna av linjesegmentet ut. En liten optimering är att låta bli att undersöka om 4 punkter är linjärt beroende om de 3 första inte är det. Programmet skriver också ut information om hur lång tid beräkningen tagit.

Det är nu dags att du undersöker källkoden, kompilerar programmet och testkör med några av indatafilerna. Experimentera med programmet och gör en analys av dess tidskomplexitet. Redogör för experimenten och den teoretiska analysen genom att fylla i filen `readme.txt` för totalsökningslösningen.

En snabbare, sorteringsbaserad, lösning

Anmärkningsvärt nog går det att lösa problemet på ett mycket effektivare sätt än med totalsökningslösningen. Givet en punkt p, bestämmer följande metod om p finns med i en mängd av 4 eller fler linjärt beroende punkter.

- Tänk på p som origo.
- För varje annan punkt q, bestäm lutningen den har gentemot p.
- Sortera punkterna efter vilken lutning de har gentemot p.
- Kontrollera om 3 (eller fler) på varandra följande punkter i den sorterade ordningen har samma lutning gentemot p. Om det är så är dessa punkter, tilsammans med p, linjärt beroende.



Skriv ett program `fast.cpp` som implementerar ovanstående algoritm. Utför samma typ av analys som för totalsökningsalgoritmen och fyll i resultaten i `readme.txt`.

Tips

En komplicerande faktor är att jämförelsefunktionen i den sorteringsbaserade lösningen behöver ändras från sortering till sortering. En variant är att skapa ett funktionsobjekt som överlagrar `operator()` och som kan användas för att göra jämförelser. Fundera på om sorteringen behöver någon ytterligare egenskap.

Möjliga utökningar

E8 — inkludera inte delsegment (2 poäng):
Det är snyggare (samt effektivare och med samma eller lägre komplexitet som den tidigare lösningen) om `fast.cpp` inte ritar ut delsegment av linjesegment innehållande 5 eller fler punkter. Om du, till exempel, ritar $p \to q \to r \to s \to t$, rita inte också $p \to q \to s \to t$ eller $q \to r \to s \to t$.

Skapa en ny branch som heter E8. Efter att du redovisat muntligt, gör en `git commit -m ''TDDD86 E8 redovisning''` och en `git push`. Skicka sedan ett mail till din assistent med ämnet: `[TDDD86] E8 redovisning`.