Marius Madsen

Mariumh

In3030 v2019 Oblig5

Konveks innhylling baserer seg på å finne alle punkter i ett sett som er ytterst i grafen, og mellom hver punkt skal vinkelen ikke være mer enn 180 grader. På denne måten vil det se ut som når en lasso fanger vedkommende.

For å kjøre programmet skriv i terminal Javac *.java && java -Xmx7500m Oblig5 k hvor k er hvor mange tråder som skal operere på innhyllingen parallelt.

Koden er fordelt inni 3 klasser. Main hvor man starter på klassene som opererer på innhyllingen. Sek som utfører den sekvensielle løsningen av algoritmen, og Par som løser parallelt.

For å løse sekvensielt så sender du inn to lister x og y som parametre som inneholder posisjon til hver punkt. Også en IntList remainingInds som representerer hvilke punkter den skal operere på (vi vil komme tilbake til hvorfor).

Allerede i Sek sin konstruktør starter den å løse oppgaven med å kalle på conv() som står for konveks innhylling. Her vil den forsøke å finne minste og største verdi av x og y, samt å lagre punktindekseringen av minste og største x. når den har gjort dette vil lagre venstre punktet i listen av løsninger før den kaller på metoden getPoints() med side = 0. deretter lagrer den punktet til høyre før den kaller getPoints() med side = 1. Vi vil forklare getPoints() sin funksjon nå.

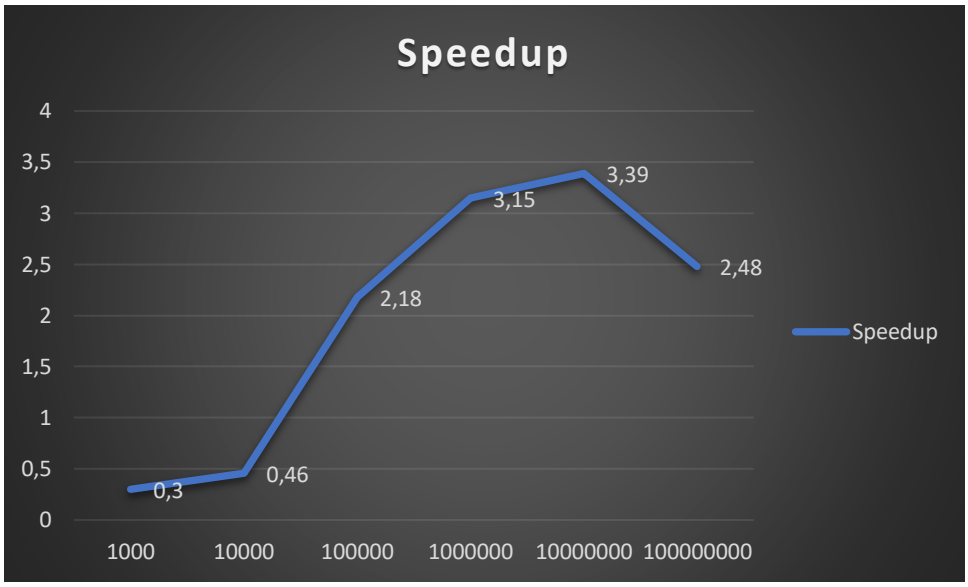
getPoints er en rekrusiv funksjon som tar inn to indekseringer som representer mest til venstre og mest til høyre, og en pointind som er til for å lagre eventuelle punkter den finner inni metoden. Den tar inn IntListen

med indekser den skal operere på, og til slutt en int variabel side som fortelle om den jobber på toppdelen eller nedre delen av algoritmen.

Den starter med å designere en boolean verdien falsk, som representerer på om den har funnet nye punkter eller ikke. Den lager to IntLister online som lagrer indekser som befinner seg midt imellom de to ytterste punktene. Og newList som lagrer alle punkter som er over eller under linjen avhengig av hvilken side den opererer på. Deretter vil den bruke den matematiske formelen oppgitt i oppgaven til å lage en linje mellom de to punktene, og lete igjennom alle punkter i remainingInds for å se om de er punkter som ligger på eller lengre ute i grafen.
Skulle algoritmen holde på toppdelen av innhyllingen vil den endre fortegn på linjen. Hvis verdien den har beregnet er mindre enn 0 vil dette være en punkt den vil måtte lagre i newList, og da endrer den verdien fant til true. Nå vil den hvis fant er true. Kalle på metoden på ny rekursivt 2 ganger. En hvor den går fra venstrepunktet til den mest ytterste punktet den fant nå, og en med det nye punktet til høyrepunktet (omvendt hvis den jobber med toppdelen). Hvis den ikke fant noe vil den kalle på checkline() for å legge til de punktene den fant langs linjen. Checkline er en innstikksorterings metode som sorterer etter minste verdi, som bli beregnet med pythagoras.

Par gjør i likhet med sek og starter allerede i konstruktøren. Der kaller den på startThreads som fordeler jevnt en mengde med punkter som hver tråd skal kjøre en sekvensiell innhylling av. Av den grunn trenger sekvensiell konstruktør å ta en liste av inter den skal gå igjennom. tter hver tråd har kjørt sin sekvensiell lagrer den sin innhylling i en IntList array lists som er global. På denne måten kan tråd 0 samle alle de samlede punktene og kjøre sekvensiell på den samlede.

n	Sek time(ms)	Par time(ms)	Speedup
1000	0,35	1,16	0,30
10000	1,36	2,99	0,46
100000	6,89	3,15	2,18
1000000	75,73	24,05	3,15
10000000	834,35	246,17	3,39
100000000	8176,08	3291,50	2,48



Ved observasjoner ser man at algoritmen ikke trenger å anvendes parallell før n >= 100000. fra der og ut har den 100% bedre eksekveringstid.

Dette var testet med Prosessor Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 2601 Mhz, 4 kjerne(r), 8 logiske prosessor(er)