Proggramering Lab 2

1dt909 Parallellprogrammering

Emil Ulvagården

Run the program

```
go run QuickSort.go

go run FloydWarshall.go
```

Problem 1

Jag börjar med att generera en random array. QuickSort kallas genom en go rutin och sedan inväntas svar i en kanal. Quicksorten börjar med att kolla längden av arrayn och om den är mindre eller lika med 1 skrivs arrayn på kanalen och returnerar sedan tillbaka. Arrans sätts pivot till det första elementet i arrayn och två separata arrays skapas. Om värdet är mindre eller lika med pivot sätts det in i den ena arrayn och om det är större sätts det in i den andra. Här skapas det två ny kanaler, en för varje regression av quickSort som ska kallas med go rutiner. En waitGroup används för att se till så att allt arbete är fördigt innan den slutgiltiga sorterade arrayn skrivs ut. Huvud quickSorten tar emot sorterad resultat från tidigare uppdelningar innan kanalerna stängs. Den sorterade vänstra kanalen sätts samman med pivot och den sorterade högra kanalen. Resultatet skickas sedan på en kanal och quickSorten är avslutad.

Problem 2

Nodes\Algoritm	Dijkstra	Floyd-Warshall
5	1.73 ms	1.10 ms
10	2.26 ms	2.27 ms
20	5.73 ms	10.33 ms
50	10.82 ms	43.45 ms
100	18.37 ms	163.76 ms

Det är en skillnad i tid mellan de olika algoritmerna vilket delvis beror på utskrift av den minsta vikten mellan flera noder. Dijksta är snabbare i alla fall som testats med fler noder än 10. När graferna blir större växer tiden för dijkstra mycket mindre än vad tiden för floyd-warshall gör. Floyd-warshall är beroende av fler lopar än dijksta vilket gör att ju fler noder som läggs till desto större blir skillnaden mellan algoritmerna.

Då all startnoder körs parrallelt för dijksta så används waitgroup och kanaler för att se till så att resultatet inte skrivs ut och så att resultat inte skrivs över.

Flojd-Warshall använder sig av kanaler för att sammanställa resultatet från de olika go kanalerna. Detta används för att inte skriva över resultatet från andra go kanaler och för att inte behöva använda lås. Go kanaler används i en for loop för att kunna jämföra summan av två distanser i dist arrayn med det direkta avståndet till noden. Om summman av de två distanserna är mindre ersätts det gamla avståndet med ett nytt och när alla avstånd för den kolumnen har genomförts skickas resultatet på en kanal.