Jon Bergland

Øving 3, algoritmer og datastrukturer

Alternativ 1:

Implementasjon

I denne øvingen skal jeg sammenligne vanlig quicksort med et delingstall, med quicksort som bruker to delingstall.

Under er min implementasjon av quicksort med et delingstall. Denne er tatt fra boka Algoritmer og datastrukturer. Metodene splitt, median3sort og bytt er også tatt fra boka.

```
public static void quickSort(int[] t, int v, int h) {  4 usages
  if (h - v > 2) {
    int delepos = splitt(t, v, h);
    quickSort(t,v,  h: delepos - 1);
    quickSort(t,  v: delepos  + 1, h);
  } else median3sort(t, v, h);
}
```

Under er min implementasjon av quicksort med to delingstall:

```
public static void quickSortDualPivots(int[] t, int v, int h) {  4 usages
  if (v < h) {
    int[] delepos = dualSplitt(t, v, h);
    quickSortDualPivots(t,v, delepos[0]);
    if (delepos[0] != delepos[1]) {
      quickSortDualPivots(t,  v: delepos[0] + 1,  h: delepos[1] - 1);
    }
    quickSortDualPivots(t,delepos[1], h);
  }
}
```

Denne er basert på implementasjonen fra geeksforgeeks, med noen viktige endringer. I metoden dualsplitt blir første tall i listen byttet med tallet som ligger 1/3 av lengden inn i listen. Det siste tallet blir byttet med tallet som ligger 1/3 av lengden bakover inn i listen. Dette sikrer at det ikke blir en skjevdeling av intervallene hvis tabellen er sortert fra før av.

En annen endring som er gjort er å sjekke om tallene på deleposisjonen er like. Hvis de er det, kan vi hoppe over å sortere de midterste elementene. Dette øker sorteringen på tabeller som består av mye duplikat informasjon.

Testing

For hver kjøring av sorteringsalgoritmen, kjører programmet et par tester for å verifisere at den har lykkes i sorteringen.

Den ene testen er å verifisere at summen av tallene i listen er lik, både før og etter sorteringen. Dette er for å verifisere at ingen data har gått tapt.

Den andre testen sjekker om listen faktisk er sortert. Den returnerer false hvis den ikke er sortert og true hvis den er det.

```
beforeSum = Arrays.stream(list).sum();

beforeTime = System.currentTimeMillis();
quickSortDualPivots(list, v: 0, h: list.length - 1);
afterTime = System.currentTimeMillis();
time = afterTime - beforeTime;
aftersum = Arrays.stream(list).sum();

sumtest(beforeSum, aftersum);

System.out.println("The list is ordered: " + orderTest(list));
```

```
public static void sumtest(int sum1, int sum2) {  1 usage
  if (sum1 == sum2) {
    System.out.println("The sum is the same");
  } else {
    System.out.println("The sum is different: " + sum1 + " (before) , " + sum2 + " (after)");
  }
}
```

```
public static boolean orderTest(int[] t) {  1 usage
  for (int i = 0; i < t.length - 2; i++) {
    if (t[i] > t[i + 1]) {
      return false;
    }
  }
  return true;
}
```

Tidtaking

Programmet kjører tidtaking av 4 forskjellige lister med tall. De listene er:

- En liste med tilfeldige tall
- En liste hvor annethvert tall er likt
- En liste som allerede er sortert
- En liste som er sortert baklengs

Alle listene består av 50 millioner tall (50 000 000) og tidtakingen ble gjort bare én gang for hver algoritme.

Under er resultatene jeg fikk av den vanlige quicksort algoritmen:

```
Quicksort with one pivot
----------------------
List with random numbers
The sum is the same
The list is ordered: true
----------------------
List with only two unique numbers
The sum is the same
The list is ordered: true
----------------------
List that already is sorted
The sum is the same
The list is ordered: true
----------------------
List that already is reversed sorted
The sum is the same
The list is ordered: true
```

List	Sorting time
Tilfeldige tall	4295ms
Duplikater	1501ms
Sortert	1695ms
Revers sortert	4371ms

Under er resultatene jeg fikk av quicksort med to delingstall:

```
Quicksort with dual pivot
----------------------
List with random numbers
The sum is the same
The list is ordered: true
----------------------
List with only two unique numbers
The sum is the same
The list is ordered: true
----------------------
List that already is sorted
The sum is the same
The list is ordered: true
----------------------
List that already is reverse sorted
The sum is the same
The list is ordered: true
```

List	Sorting time
Tilfeldige tall	3641ms
Duplikater	1630ms
Sortert	1384ms
Revers sortert	3702ms

Ut ifra tidsmålingene kan vi se at quicksort med to delingstall er i ca 10-20% på alle listene utenom duplikater, hvor den er ca. 10% treigere. Dette kommer nok av at listen med duplikater består av to unike tall, så i quick sort blir listen sortert etter å ha plukket ut delingstallet. Alle tallene av den ene typen havne over eller under delingstallet, basert på hvilket delingstall som blir valgt.

Konklusjon

For mine implementeringer av quick sort med en og to delingstall, ville jeg valgt quick sort med et delingstall for data som jeg vet består av få unike tall og den med to delingstall for alle øvrige dataset.