Usman Ghafoorzai

Forskningsoppgave sortering

Kildekode

```
import java.util.Random;

public class Oving3 {

  public static void main(String[] args) {
    int n = 50_000_000;

    int[] randomArray = generateRandomArray(n);
    int[] duplicateArray = generateDuplicateArray(n);
    int[] sortedArray = generateSortedArray(n);
    int[] reverseArray = generateReverseSortedArray(n);

    testSortWithTiming(randomArray.clone(), "QuickSort", "Tilfeldig data");
    testSortWithTiming(duplicateArray.clone(), "QuickSort", "Mange duplikater");
    testSortWithTiming(sortedArray.clone(), "QuickSort", "Sortert fra før");
    testSortWithTiming(reverseArray.clone(), "QuickSort", "Baklengs sortert");

    testSortWithTiming(randomArray.clone(), "DualPivotQuickSort", "Tilfeldig data");
    testSortWithTiming(duplicateArray.clone(), "DualPivotQuickSort", "Mange duplikater");
    testSortWithTiming(sortedArray.clone(), "DualPivotQuickSort", "Sortert fra før");
    testSortWithTiming(reverseArray.clone(), "DualPivotQuickSort", "Baklengs sortert");
  }

  public static void testSortWithTiming(int[] arr, String sortType, String dataType) {
    System.out.println("\nTester " + sortType + " på " + dataType + "...");

    int checksumBefore = calculateChecksum(arr);
    System.out.println("Sjekksum før sortering: " + checksumBefore);

    long startTime = System.currentTimeMillis();

    if (sortType.equals("QuickSort")) {
      quicksort(arr, 0, arr.length - 1);
    } else if (sortType.equals("DualPivotQuickSort")) {
      dualPivotQuickSort(arr, 0, arr.length - 1);
    }

    long endTime = System.currentTimeMillis();
    long duration = endTime - startTime;

    int checksumAfter = calculateChecksum(arr);
    System.out.println("Sjekksum etter sortering: " + checksumAfter);

    if (checksumBefore == checksumAfter) {
      System.out.println("Sjekksummene stemmer. Ingen datatap.");
    } else {
      System.out.println("Sjekksummene stemmer ikke. Det har skjedd en feil.");
    }

    if (isSorted(arr)) {
      System.out.println("Arrayet er korrekt sortert.");
    } else {
      System.out.println("Arrayet er IKKE korrekt sortert.");
    }

    System.out.println(sortType + " på " + dataType + " tok " + duration + " millisekunder.");
  }

  public static int calculateChecksum(int[] arr) {
    int sum = 0;
    for (int i : arr) {
      sum += i;
    }
    return sum;
  }

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

Usman Ghafoorzai

```
public static int[] generateRandomArray(int n) {
  Random rand = new Random();
  int[] arr = new int[n];
  for (int i = 0; i < n; i++) {
    arr[i] = rand.nextInt(n);
  }
  return arr;
}

public static int[] generateDuplicateArray(int n) {
  int[] arr = new int[n];
  for (int i = 0; i < n; i++) {
    arr[i] = (i % 2 == 0) ? 1 : i;
  }
  return arr;
}

public static int[] generateSortedArray(int n) {
  int[] arr = new int[n];
  for (int i = 0; i < n; i++) {
    arr[i] = i;
  }
  return arr;
}

public static int[] generateReverseSortedArray(int n) {
  int[] arr = new int[n];
  for (int i = 0; i < n; i++) {
    arr[i] = n - i;
  }
  return arr;
}

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

private static int median3sort(int[] t, int v, int h) {
  int m = (v + h) / 2;
  if (t[v] > t[m]) bytt(t, v, m);
  if (t[m] > t[h]) {
    bytt(t, m, h);
    if (t[v] > t[m]) bytt(t, v, m);
  }
  return m;
}

private static int splitt(int[] t, int v, int h) {
  int iv, ih;
  int m = median3sort(t, v, h);
  int dv = t[m];
  bytt(t, m, h - 1);
  for (iv = v, ih = h - 1;;) {
    while (t[++iv] < dv);
    while (t[--ih] > dv);
    if (iv >= ih) break;
    bytt(t, iv, ih);
  }
  bytt(t, iv, h - 1);
  return iv;
}

private static void bytt(int[] t, int i, int j) {
  int k = t[j];
  t[j] = t[i];
  t[i] = k;
}
```

Usman Ghafoorzai

```
    public static void dualPivotQuickSort(int[] t, int v, int h) {
      if (v < h) {
        int pivot1 = v + (h - v) / 3;
        int pivot2 = h - (h - v) / 3;

        swap(t, v, pivot1);
        swap(t, h, pivot2);

        int[] piv = partition(t, v, h);

        dualPivotQuickSort(t, v, piv[0] - 1);
        if (t[piv[0]] != t[piv[1]]) {
          dualPivotQuickSort(t, piv[0] + 1, piv[1] - 1);
        }
        dualPivotQuickSort(t, piv[1] + 1, h);
      }
    }

    private static int[] partition(int[] t, int v, int h) {
      if (t[v] > t[h]) swap(t, v, h);

      int j = v + 1;
      int g = h - 1, k = v + 1;
      int p = t[v], q = t[h];

      while (k <= g) {
        if (t[k] < p) {
          swap(t, k, j);
          j++;
        } else if (t[k] >= q) {
          while (t[g] > q && k < g) {
            g--;
          }
          swap(t, k, g);
          g--;

          if (t[k] < p) {
            swap(t, k, j);
            j++;
          }
        }
        k++;
      }
      j--;
      g++;

      swap(t, v, j);
      swap(t, h, g);

      return new int[]{j, g};
    }

    private static void swap(int[] t, int i, int j) {
      int temp = t[i];
      t[i] = t[j];
      t[j] = temp;
    }
  }
```

Usman Ghafoorzai

Tidsmålinger samt sjekksum- og rekkefølgetest

Enkel-pivot-Quicksort

```
C:\Users\47968\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent

Tester QuickSort på Tilfeldig data...
Sjekksum før sortering: -755718819
Sjekksum etter sortering: -755718819
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
QuickSort på Tilfeldig data tok 14767 millisekunder.

Tester QuickSort på Mange duplikater...
Sjekksum før sortering: 679053376
Sjekksum etter sortering: 679053376
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
QuickSort på Mange duplikater tok 4488 millisekunder.

Tester QuickSort på Sortert fra før...
Sjekksum før sortering: 1283106752
Sjekksum etter sortering: 1283106752
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
QuickSort på Sortert fra før tok 2883 millisekunder.

Tester QuickSort på Baklengs sortert...
Sjekksum før sortering: 1333106752
Sjekksum etter sortering: 1333106752
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
QuickSort på Baklengs sortert tok 4471 millisekunder.
```

Usman Ghafoorzai

Dual-pivot-Quicksort

```
Tester DualPivotQuickSort på Tilfeldig data...
Sjekksum før sortering: -755718819
Sjekksum etter sortering: -755718819
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
DualPivotQuickSort på Tilfeldig data tok 12975 millisekunder.

Tester DualPivotQuickSort på Mange duplikater...
Sjekksum før sortering: 679053376
Sjekksum etter sortering: 679053376
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
DualPivotQuickSort på Mange duplikater tok 2677 millisekunder.

Tester DualPivotQuickSort på Sortert fra før...
Sjekksum før sortering: 1283106752
Sjekksum etter sortering: 1283106752
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
DualPivotQuickSort på Sortert fra før tok 2779 millisekunder.

Tester DualPivotQuickSort på Baklengs sortert...
Sjekksum før sortering: 1333106752
Sjekksum etter sortering: 1333106752
Sjekksummene stemmer. Ingen datatap.
Arrayet er korrekt sortert.
DualPivotQuickSort på Baklengs sortert tok 2910 millisekunder.

Process finished with exit code 0
```

Usman Ghafoorzai

Analyse av tidsmålinger

Ifølge s.65 i boka *Algoritmer og datastrukturer* (2014) av Helge Hafting og Mildrid Ljosland er den gjennomsnittlige kjøretiden for enkel-pivot-Quicksort $O(n\log n)$. Utfra s. 19 i forelesningsnotatene fra forelesningen om sortering i faget *IDATT2101 Algoritmer og datastrukturer* mandag 02/09/2024, står det at til tross for samme tidskompleksitet som enkel-pivot-Quicksort, er dual-pivot-Quicksort noe raskere enn det forrigenevnte.

For å sammenligne ytelsen mellom enkel-pivot-Quicksort og dual-pivot-Quicksort ble begge algoritmene testet på fire forskjellige datasett med 50 millioner elementer. Dette er vist i *Tidsmålinger samt sjekksum- og rekkefølgetest*. Nedenfor fremstilles resultatene i en tabell:

Datatype	**Enkel-pivot-Quicksort (ms)**	**Dual-Pivot-Quicksort**
Tilfeldig data	14767	12975
Mange duplikater	4488	2677
Sortert data	2883	2779
Baklengs sortert data	4471	2910

Vi ser at resultatene er i samsvar med at dual-pivot er raskere enn enkel-pivot. På tilfeldig data ser vi at dual er raskere, ettersom antallet sammenligninger er redusert vha. to pivoter, siden det fører til mer balansert oppdeling. I mange duplikater var dual raskere siden algoritmen ble modifisert for å unngå rekursive kall på midt-intervallet når pivoter er like. På sortert data er også dual raskere, siden der òg ble algoritmen modifisert til å velge pivoter fra en tredjedel av arrayet for å unngå skjevfordeling. På baklengs sortert data drar dual igjen fordelen av to pivoter som er raskere enn enkel.

Konklusjonen er at dual-pivot gjennomgående viser seg å være raskere enn enkel-pivot. Spesielt i tilfeller med mange duplikater og sortert/baklengs sortert data. Dette er på grunn av mer effektiv partisjonering med to pivoter, samt optimalisering som unngår unødvendige rekursive kall når pivoter er like.