# Tölvunarfræði 2 - Heimadæmi 5

## Dæmi 1)

Þegar CUTOFF er í kringum 10 þá er tími 0.055 sem kemur best út hjá mér ☺

**Hér kemur main fall:**

public static void main(String[] args) {  
 int N = 100000; // Array size  
 Double[] a = new Double[N];  
 for (int i = 0; i < N; i++) {  
 a[i] = StdRandom.uniform();  
 }  
  
 Stopwatch timer = new Stopwatch();  
 Merge.sort(a);  
 double time = timer.elapsedTime();  
  
 System.out.println("Sorting time: " + time + " seconds");  
}

## Dæmi 2)

1. Gætum notað fylkið [1, 3, 5, 7, 9] því mun minnka í lo án þess að þurfa að skoða hvort 1 er minnsta stakið.
2. Getum notað fylkið [5, 4, 3, 2, 1] hér er lo 5 og hæsta stakið í fylkinu i mun alltaf vera minna en lo en þegar i == j þá stoppar lykkjan. Ef það væri ekki samanburður inn í lykkju þá þa myndi i færast upp og vera alltaf minni en lo og þá myndi i enda út fyrir fylkið.
3. Leiðin í bókinni setur hæsta stakið aftast í fylkið og skiptir þá á hæsta staki og aftasta staki. i mun hækka og stakið sem i bendir á er borið saman við stakið sem lo bendir á. I mun þá benda á lægri eða hærri stök en lo en þegar i bendir á síðasta stakið mun less(a[++i], a[lo]) ekki vera satt og förum þá út úr while lykkju og i fer aldrei í fylkið.

[5, 4, 3, 2, 1], hæsta stakið fært aftast [1, 4, 3, 2, 5].

## Dæmi 3)

Upphaflega fylkið

Gerum fyrsta víxl á H og D (index 0 og 7)

Gerum annað víxl á P og F (index 3 og 6)

Lokastaða fylkis

## Dæmi 4)

1. Þegar við erum búinn að bæta tölunum 4, 7, 11, 2, 6, 8, 12 í þessari röð í max hrúgu sem er upphaflega tóm þá fáum við lokastöu 12, 6, 11, 2, 4, 7, 8.

11 ->6 -> 2

-> 4

->8 -> 7

12 -> 6 -> 2

-> 4

->11-> 7

-> 8

1. Möguleg sæti fyrir 4-stærsta stakið er 3, 4, 10, 11 því fjórða stærsta stakið getur ekki komið fyrir ofan þriðja stærsta í beinum legg, þá útilokum við sæti 1 og 2. Fjórða stærsta gæti komið við hlið þriðja stærsta í sæti 4 eða fyrir ofan í hinn legginn í sæti 3. Þá getur fjórða stærsta ekki farið fyrir neðan þessi sæti og því útilokum við sæti 6,7,8,9,12. En getur komið fyrir neðan þriðja stærsta og því möguleiki að fara í sæti 10 og 11.

## Dæmi 5

import edu.princeton.cs.algs4.\*;  
  
  
public class Kmax1 {  
  
 // Notum MaxPQ  
 public static Double[] findKLargestUsingMaxPQ(Double[] a, int K) {  
 MaxPQ<Double> maxPQ = new MaxPQ<>(a);  
 Double[] result = new Double[K];  
 for (int i = 0; i < K; i++) {  
 result[i] = maxPQ.delMax();  
 }  
 return result;  
 }  
  
 // Notum MinPQ  
 public static Double[] findKLargestUsingMinPQ(Double[] a, int K) {  
 MinPQ<Double> minPQ = new MinPQ<>(K);  
 for (Double item : a) {  
 if (minPQ.size() < K) {  
 minPQ.insert(item);  
 } else if (item.compareTo(minPQ.min()) > 0) {  
 minPQ.delMin();  
 minPQ.insert(item);  
 }  
 }  
 Double[] result = new Double[K];  
 for (int i = 0; i < K; i++) {  
 result[i] = minPQ.delMin();  
 }  
 return result;  
 }  
  
 public static void main(String[] args) {  
 int N = 10000000;  
 int K = 100;  
 Double[] a = new Double[N];  
  
 // 10 milljón random tölur  
 for (int i = 0; i < N; i++)  
 a[i] = StdRandom.uniformDouble(0.0, 10.0);  
  
 // tími fyrir MaxPQ  
 Stopwatch timerMaxPQ = new Stopwatch();  
 findKLargestUsingMaxPQ(a, K);  
 double timeMaxPQ = timerMaxPQ.elapsedTime();  
 StdOut.println("Tími með að nota MaxPQ: " + timeMaxPQ + " sekúndur");  
  
 // tími fyrir MinPQ  
 Stopwatch timerMinPQ = new Stopwatch();  
 findKLargestUsingMinPQ(a, K);  
 double timeMinPQ = timerMinPQ.elapsedTime();  
 StdOut.println("Tími með að nota MinPQ: " + timeMinPQ + " sekúndur");  
 }  
}

A screenshot of a computer program

Description automatically generated