TÖL101G - Tölvunarfræði 1
Vikublað 12
Lausn

Æfingar

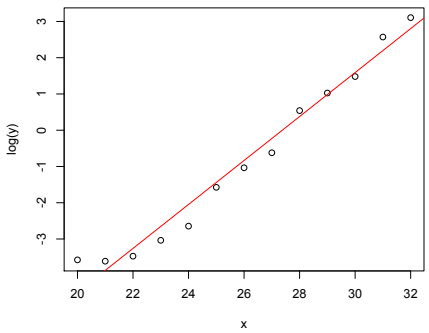
4.1.12

Ef við prófum fallið í eftirfarandi klasa með $N = 25, 26, 27, 28, 29, 30$ fæst.

```
public class ExpTime {
    public static int f(int n) {
        if (n==0) return 1;
        return f(n-1) + f(n-1);
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        Stopwatch e = new Stopwatch();
        int r = f(N);
        double t = e.elapsedTime();
        System.out.println("value: " + r + ", time: "+ t) ;
    }
}
```

Nokkuð bein lína þegar við plottum $\log(T(N))$ sem fall af N



Þetta passar við veldisvísisvöxt, þ.e. $O(a^N)$ fyrir eitthvert $a > 1$ (fyrir margliður yrði grafið flatt eftir því sem N hækkar). Við sjáum að fjöldi aðgerða er $T(N) = O(1) + 2 \cdot T(N-1)$. Ef við stingum inn $T(N) = b \cdot a^N$ inn þá fáum við

$$\begin{aligned} & T(N) = O(1) + 2 \cdot T(N-1) \\ \Leftrightarrow & b \cdot a^N = O(1) + 2 \cdot b \cdot a^{N-1} \\ \Leftrightarrow & b \cdot a^{N-1}(a-2) = O(1) \end{aligned}$$

En seinasta jafnan getur aðeins verið sönn ef $a = 2$ og því er $T(N) = O(2^N)$.

4.1.16

1. Látum $T(N)$ tákna keyrslutímann fyrir inntakið N.

```
1 public static String method1(int N) {
2   if      (N == 0) return "";
3   String temp = method1(N / 2);
4   if (N % 2 == 0) return temp + temp;
5   else            return temp + temp + "x";
6 }
```

   Fjöldi aðgerða er $T(N/2)+O(1)$ fyrir utan strengjasamskeytingu sem er háð stærðinni á `temp`. Við nánari athugun sést að `method1(N)` skilar strengnum `"xxx...xx"`, þ.e. x endurtekið N sinnum. Því kostar strengjasamskeytingin $O(N)$ og við fáum jöfnuna $T(N) = T(N/2) + O(N)$. Með endurtekinni ítrekun fæst

$$\begin{aligned} T(N) &= T(N/2) + O(N) \\ &= T(N/4) + O(N) + O(N/2) \\ &= T(N/8) + O(N) + O(N/2) + O(N/4) \\ &= \ldots \\ &= T\left(\frac{N}{2^k}\right) + O\left(\sum_{i=0}^{k} \frac{N}{2^i}\right) \\ &= T\left(\frac{N}{2^k}\right) + O\left(N \cdot \sum_{i=0}^{k} \frac{1}{2^i}\right) \\ &= \ldots \\ &= T(0) + O(N) \end{aligned}$$

   Þar sem summan af 2^{-i} er í mesta lagi 2 verður seinni hlutinn bara $O(N)$.

2. Látum $T(N)$ tákna keyrslutímann fyrir inntakið N.

```
1 public static String method2(int N) {
2   String s = "";
3   for (int i = 0; i < N; i++) {
4     s = s + "x";
5   }
6   return s;
7 }
```

   Línur 2 og 5 taka $O(1)$ tíma og línur 3-4 eru keyrðar N sinnum. Í línu 4 er x bætt við strenginn `s` og í i-tu ítrun samanstendur `s` af i x-um. Því kostar lína 4 $O(i)$

aðgerðir. Því fáum við að

$$\begin{aligned} T(N) &= O(1) + \sum_{i=1}^{N} O(i) \\ &= O(1) + O\left(\sum_{i=1}^{N} i\right) \\ &= O(1) + O\left(\frac{N(N-1)}{2}\right) \\ &= O(N^2) \end{aligned}$$

3. Látum $T(N)$ tákna keyrslutímann fyrir inntakið N.

```
1 public static String method3(int N) {
2   if      (N == 0) return "";
3   else if (N == 1) return "x";
4   else return method3(N / 2) + method3(N - (N / 2));
5 }
```

Línur 2-3 taka $O(1)$ tíma og `method3(N)` skilar streng af lengd N. Því kostar samskeytingin á strengjunum sem skilað er í línu 4 samtals $O(N)$ aðgerðir. Tímaflækjan hefur því jöfnuna $T(N) = 2 \cdot T(N/2) + O(N)$. Þetta er sama formúlan og fyrir Mergesort og því vitum við að svarið er $T(N) = O(N \log N)$.

4. Látum $T(N)$ tákna keyrslutímann fyrir inntakið N.

```
1 public static String method4(int N) {
2   char[] temp = new char[N];
3   for (int i = 0; i < N; i++) {
4     temp[i] = 'x';
5   }
6   return new String(temp);
7 }
```

Lína 2 tekur $O(N)$ skref, lína 4 tekur $O(1)$ skref og er framkvæmd N sinnum. Loks er kostar smiðurinn fyrir `String` í línu 6 $O(N)$ skref. Samtals verða þetta því $T(N) = O(N)$ skref.

4.2.5

Til að helmingunarleit virki þarf fylkið að vera raðað. Ef lyklarnir í fylkinu eru breytanlegir þá væri hægt að breyta gildinu á staki í fylkinu og því mögulegt að eyðileggja röðunina. Ef fylkið er ekki lengur raðað er líklegt að helmingunarleit skili ekki réttu svari.

Verkefni

4.1.35

Athugið að reverse fallið notar $O(1)$ auka minni og hefur hliðarverkanir. Fyrst snúum við fyrstu k stökunum við, síðan síðustu $N - k$ stökunum og loks öllu fylkinu. Þetta er

jafngilt því að taka fyrstu k stökin og setja þau aftast í fylkið.

```
public class Rotate {
  // Notkun: reverse(a,i,j)
  // Fyrir:  a[i..j] er svæði í a
  // Eftir:  a[i..j] er í öfugri röð
  public static void reverse(char[] a, int i, int j) {
    int N = j-i;
    for (int k = 0; k < N/2; k++) {
      char tmp = a[i+k];
      a[i+k] = a[i + (N-k-1)];
      a[i+(N-k-1)] = tmp;
    }
  }

  // Notkun: rotate(a,k)
  // Fyrir:  0 <= k <= a.length
  // Eftir:  a er breytt í a[k..N]+a[0,..,k-1]
  public static void rotate(char[] a, int k) {
    int N = a.length;
    reverse(a,0,k);
    reverse(a,k,N);
    reverse(a,0,N);
  }

  public static void main(String[] args) {
    String s = args[0];
    int k = Integer.parseInt(args[1]);
    System.out.println(s);
    char[] a = s.toCharArray();
    rotate(a,k);
    System.out.println(new String(a));

  }
}
```

4.2.25

Fyrri lausnin telur fjölda 3SUM lausna rétt eins og lausnin í bókinni. Fyrstu tvær for lykkjurnar eru eins en í lokin er leitað að stakinu sem vantar upp á með helmingunarleit. Til að telja nákvæman fjölda lausna verðum við að finna hve oft $-s$ kemur fyrir. Athugum að helmingunarleitin skilar minnsta vísi þar sem $-s$ kemur fyrir og til að finna stærsta vísinn leitum við að næstu tölu fyrir ofan, þ.e. $-s+1$. Athugið að þetta tekur aðeins $O(\log(N))$ skref. Ef við leitum að hvar $-s$ kemur fyrir með for lykkju þá gæti það tekið $O(N)$ í versta tilfelli (þegar talan er endurtekin mjög oft). Fjöldi aðgerða verður því $O(N^2 \log(N))$.

Seinni aðferðin finnur aðeins hvort að til sé lausn. Hún keyrir á $O(N^2)$ tíma en það er skilið eftir sem æfing að sýna fram á það.

```java
import java.util.*;

public class ThreeSum {


  // Notkun: k = helmingunarleit(a,x,i,j)
  // Fyrir:  a[i],..,a[j] er í vaxandi röð
  // Eftir:  a[i],..,a[k-1] < x <= a[k],..,a[j]
  public static int helmingunarleit(int[] a, int x, int i, int j) {
    if (i > j) return i;
    int mid = (i+j)/2;
    if (a[mid] < x) {
      return helmingunarleit(a,x,mid+1,j);
    } else {
      return helmingunarleit(a,x,i,mid-1);
    }
  }

  // Notkun: x = count(a)
  // Fyrir:  a er raðað í vaxandi röð
  // Eftir:  x er fjöldi vísa i<j<k þ.a. a[i] + a[j] + a[k] == 0
  public static int count(int[] a) {
    int N = a.length;
    int cnt = 0;
    for(int i = 0; i < N; i++) {
      for(int j = i+1; j < N; j++) {
        int s = a[i] + a[j];
        int k = helmingunarleit(a,-s,j+1,N-1);
        if (k >= 0 && k < N &&  a[k] == -s) {
          int t = helmingunarleit(a,-s+1,k+1,N-1);
          cnt += t-k;
        }
      }
    }
    return cnt;
  }

  // Notkun: b = count2(a)
  // Fyrir:  a er raðað í vaxandi röð,
  // Eftir:  b er satt ef til eru  i<j<k þ.a. a[i] + a[j] + a[k] == 0
  public static boolean count2(int[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
      int low = i+1;
      int high = N-1;
      while (low < high) {
        // Allar lausnir i,j,k eru þ.a. low <= j < k <= high
        int s = a[i] + a[low] + a[high];
```

```
        if (s == 0) {
          return true;
        } else if (s > 0) {
          high--; // lækkum efri mörk
        } else {
          low++;  // hækkum neðri mörk
        }
      }
    }
    return false;
  }

  public static void main(String[] args) {
    Vector<Integer> v = new Vector<Integer>();
    while (!StdIn.isEmpty()) {
      v.add(StdIn.readInt());
    }
    int[] a = new int[v.size()];
    for (int i = 0; i < a.length; i++) {
      a[i] = v.get(i);
    }
    Arrays.sort(a);
    System.out.println(count(a));
    System.out.println(count2(a));
  }

}
```