Vježbe 9 - mjerenje vremena izvršavanja i memorijske potrošnje *Java* programa

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

21. prosinca, 2022.

Mjerenje vremena izvršavanja

Vrijeme izvršavanja programa ili naredbi u *Javi* možemo realizirati koristeći:
- Naredbe `java.lang.System.nanoTime()` i `currentTimeMillis()` koje omogućavaju **bilježenje vremena izvršavanja** *Java virtualnog stroja* u **nanosekundama odnosno milisekundama**. **Bilježenjem vremena izvršavanja prije poziva metode ili programa te nakon završetka poziva metode ili neposredno prije kraja izvršavanja programa te računanjem razlike u vremenima dobijemo vrijeme izvršavanja metode ili programa u nanosekundama ili milisekundama**.
- `java.time.Instant.Now()`, dohvaća **trenutno sistemsko vrijeme**. Ova naredba se može koristiti u kombinaciji s `java.time.Duration` (koja **računa razliku između dva vremenska trenutka**).
- **Klasu** `StopWatch` iz paketa `org.apache.commons.lang3.time .StopWatch;`. Ovaj paket treba skinuti i dodati kao biblioteku programu prije korištenja. Paket se može pronaći na poveznici: `https://commons.apache.org/proper/commons-lang/download_lang.cgi`.

Računanje izraza nad dva vektora

Zadatak 1

Za dva vektora $\vec{v_1},\ \vec{v_2} \in \mathbb{R}^n$, izračunajte vrijeme izvođenja računanja izraza $\sqrt{\sum_{i=1}^{n} |v_{1_i} \cdot v_{2_i}|}$. Ispitajte vremena za $n = 100000 * (k + 1)$, $k = 0, \ldots, 100$. Elemente oba vektora generirajte na slučajan način korištenjem naredbe `r.nextDouble()*(r.nextInt()%100)`, za varijablu r tipa `Random`. Nacrtajte linijski graf koji vizualizira dobivena vremena izvođenja u milisekundama tako da se na x osi nalazi veličina korištenih vektora a na y osi vrijeme računanja izraza u milisekundama. Graf spremite kao sliku u `.png` formatu.

Računanje izraza nad dva vektora

```
1  import java.io.File;
2  import java.io.IOException;
3  import java.time.Duration;
4  import java.time.Instant;
5  import java.util.ArrayList;
6  import java.util.Random;
7  import java.util.concurrent.TimeUnit;
8  import org.apache.commons.lang3.time.StopWatch;
9  import org.jfree.chart.ChartFactory;
10 import org.jfree.chart.ChartUtils;
11 import org.jfree.chart.JFreeChart;
12 import org.jfree.chart.plot.PlotOrientation;
13 import org.jfree.data.xy.XYSeries;
14 import org.jfree.data.xy.XYSeriesCollection;
15
16 public class MjerenjeVremena {
17     ArrayList<Double> vec1, vec2;
```

Mjerenje vremena izvršavanja izraza nad vektorima.

Računanje izraza nad dva vektora

```
1   MjerenjeVremena(ArrayList<Double> v, ArrayList<Double> v1)
     {
2        vec1 = v; vec2 = v1;
3        if(vec1.size()!=vec2.size())
4            throw new java.lang.IllegalArgumentException();
5    }
6
7     public double racunaj(){
8         double res = 0.0;
9         for(int i=0;i<vec1.size();i++)
10            res+=Math.abs(vec1.get(i)*vec2.get(i));
11        return Math.sqrt(res);
12     }
13
14   public static void main(String args[])throws IOException
     {
15
16    ArrayList<Long> vremena = new ArrayList<>();
```

Mjerenje vremena izvršavanja izraza nad vektorima.

Računanje izraza nad dva vektora

```
1  for(int vel=0;vel<100;vel++){
2    System.out.println("Velicina vektora: "+100000*(vel+1));
3    ArrayList<Double> prvi = new ArrayList<>(), drugi = new
       ArrayList<>();
4    Random r = new Random();
5
6    for(int i=0;i<100000*(vel+1);i++){
7       prvi.add(r.nextDouble()*(r.nextInt()%100));
8       drugi.add(r.nextDouble()*(r.nextInt()%100));}
9    MjerenjeVremena obj = new   MjerenjeVremena(prvi, drugi);
10   long start1 = System.nanoTime();
11   System.out.println("Rezultat: "+obj.racunaj());
12   long end1 = System.nanoTime();
13   System.out.println("Vrijeme u nanosekundama: "+ (end1-
       start1));
14   long start2 = System.currentTimeMillis();
15   System.out.println("Rezultat: "+obj.racunaj());
16   long end2 = System.currentTimeMillis();
```

Mjerenje vremena izvršavanja izraza nad vektorima.

Računanje izraza nad dva vektora

```
1  System.out.println("Vrijeme u milisekundama: "+ (end2-
     start2));
2   Instant inst1 = Instant.now();
3   System.out.println("Rezultat: "+obj.racunaj());
4   Instant inst2 = Instant.now();
5   System.out.println("Vrijeme: "+ Duration.between(inst1,
     inst2).toString());
6   StopWatch stopWatch = new StopWatch();
7   stopWatch.start();
8   System.out.println("Rezultat: "+obj.racunaj());
9   stopWatch.stop();
10  System.out.println("Vrijeme: "+ stopWatch.getTime(TimeUnit
     .MILLISECONDS)+"\n"+stopWatch.formatTime());
11       vremena.add(stopWatch.getTime(TimeUnit.MILLISECONDS));
12   }
```

Mjerenje vremena izvršavanja izraza nad vektorima.

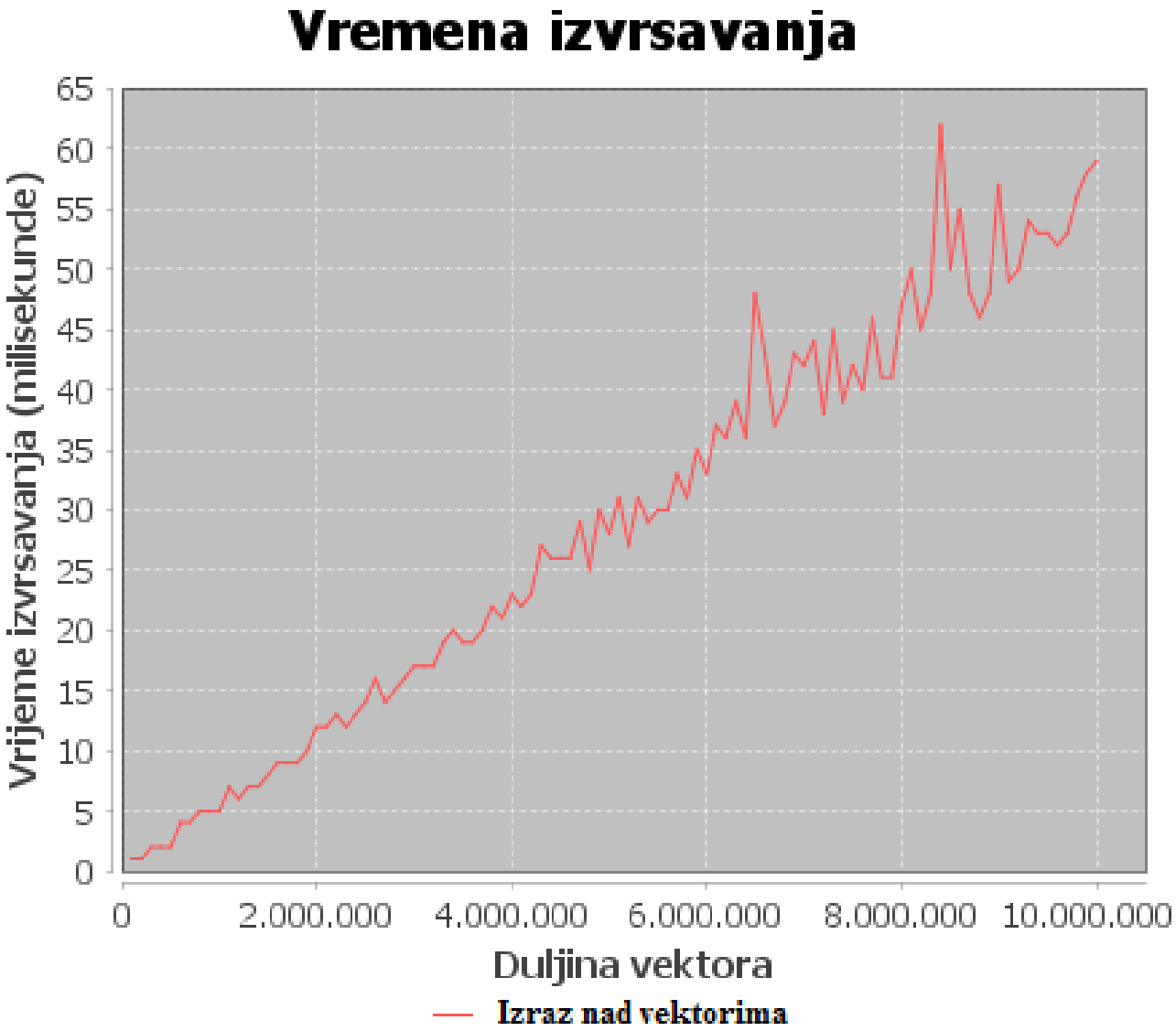
Računanje izraza nad dva vektora

```
1    //crtanje grafa izvrsavanja
2    var podaci1 = new XYSeriesCollection();
3    var series = new XYSeries ("Izraz nad vektorima");
4     int count =0;
5   for ( long i : vremena ) {
6    series.add ( 100000*(count+1) , i );// dodamo parove x,y
7    count++;
8    }
9   podaci1.addSeries( series );
10  JFreeChart chart = ChartFactory.createXYLineChart (
11    " Vremena izvrsavanja ", "Duljina vektora", "Vrijeme
     izvrsavanja (milisekunde)", podaci1 , PlotOrientation.
     VERTICAL , true , true , false );
12  ChartUtils.saveChartAsPNG(new File ("vremenaIzvrsavanja.
     png") , chart , 450 , 400);
13  }
14 }
```

Mjerenje vremena izvršavanja izraza nad vektorima.

Računanje izraza nad dva vektora

Rezultat: Intel Core $i3-5005U$, $2,00$GHz.



Računanje umnoška dvije matrice

Zadatak 2

Usporedite vrijeme množenja dvije matrice $M_1 \in \mathbb{R}_{m,n}$ i $M_2 \in \mathbb{R}_{n,k}$ ukoliko se računa sekvencijalno (koristeći jednu dretvu) i u višedretvenom okruženju na način da jedan zadatak predstavlja računanje jednog retka rezultantne matrice $M_3 \in \mathbb{R}_{m,k}$. Eksperiment provedite za $m = 10000$ i $n \in \{10, 50, 250, 1250, 6250\}$ i $k = 100$. Koristite $2, 4, 6, 8$ dretvi kod višedretvenog programa. Vremena sekvencijalnog i višedretvenih izvođenja nacrtajte kao usporedni linijski graf. Graf spremite kao sliku u `.png` formatu.

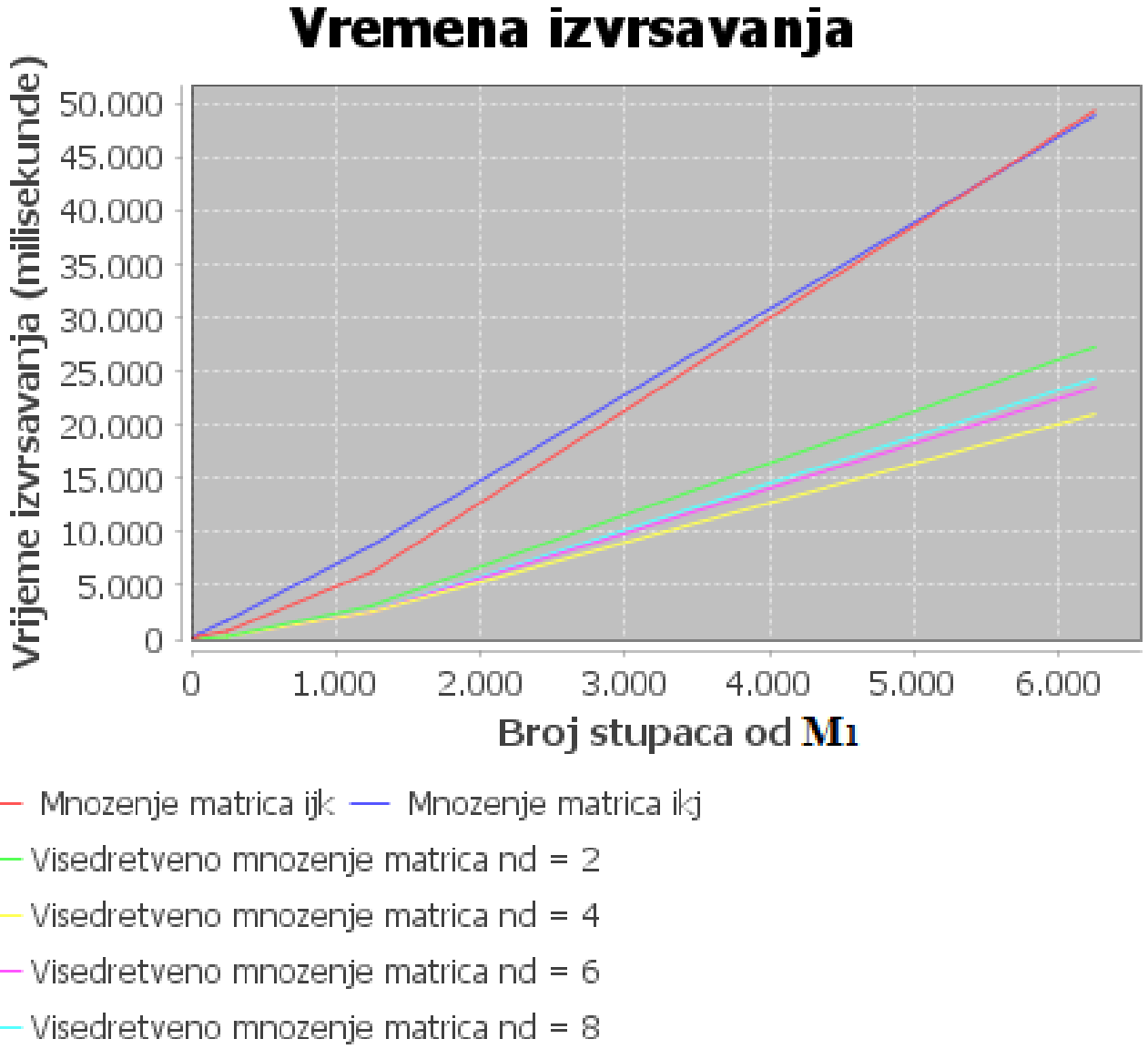
Računanje umnoška dvije matrice

Ideja rješenja:

- Implementiramo normalno dva načina sekvencijalnog množenja (i,j,k) - prva petlja po redcima matrice M_1, druga petlja po stupcima matrice M_2, treća petlja po stupcima matrice M_1 i (i,k,j) - prva petlja po redcima matrice M_1, druga petlja po stupcima matrice M_1, treća petlja po stupcima matrice M_2.
- Definiramo klasu `MnoziRedak` koja će sadržavati reference sve tri matrice kao i *id* retka matrice M_1 nad kojem će raditi. Unutar klase se provodi računanje retka *id* matrice M_3 (odnosno računa se skalarni umnožak *id*-tog retka od M_1 i svih stupaca matrice M_2).
- Višedretvena računanje postižemo korištenjem `ThreadPoolExecutor` s predefiniranim brojem dretvi, svaki redak matrice M_1 reprezentira poseban zadatak koji se daje na obradu izvršivaču s fiksnim brojem dretvi.
- Unutar aplikacije eksperimentiramo s različitim brojem dretvi unutar izvršivača grupe dretvi i različitim brojem stupaca matrice M_1 odnosno redaka matrice M_2.

Računanje umnoška dvije matrice

Rezultat: Intel Core $i3 - 5005U$, $2,00$GHz.



Računanje memorijske potrošnje programa

Ukoliko u danom trenutku *Java virtualni stroj* na našem računalu izvodi **samo jedan program**, možemo **ugrubo** izračunati memorijsku potrošnju toga programa. Memorijsku potrošnju procjenjujemo kao **razliku između ukupne memorije alocirane za izvršavanje** *Java virtualnog stroja* i **ukupne slobodne memorije** za *Java virtualni stroj*. Da bi dohvatili informacije o okruženju u kojem se aplikacija izvodi koristimo funkcije klase `java.lang.Runtime`. Pozivom `Runtime.getRuntime()` dohvaćamo klasu koja sadrži informacije o okruženju, a pozivima metoda `totalMemory()` i `freeMemory()` na objektu tipa `Runtime` dohvaćamo količinu ukupne memorije i slobodne memorije instance *Java virtualnog stroja* koji izvršava aplikaciju.

Računanje memorijske potrošnje programa

Zadatak 3

Izračunajte ukupnu potrošnju programa pri alokaciji matrica $M \in \mathbb{R}_{m,n}$, gdje $m = 10000$, a $n \in \{10, 100, 1000, 4000, 8000, 10000\}$. Ispišite ukupnu količinu memorije dodijeljenu instanci *Java virtualnog stroja*, slobodnu količinu memorije te instance te procijenjenu memorijsku potrošnju programa.

Računanje memorijske potrošnje programa

```
1  public class MemorijskaPotrosnja {
2      public static void main(String args[]){
3           Runtime rt = Runtime.getRuntime();
4           long totalMemory, freeMemory;
5          int m = 10000;
6          int dims[] = {10, 100, 1000, 4000, 8000, 10000};
7          double res[][];
8
9          System.out.println("Dimenzije    Ukupno M
       Slobodno M      Koristena M");
10         for(int i=0;i<dims.length;i++){
11             res = new double[m][dims[i]];
12             totalMemory = rt.totalMemory();
13             freeMemory = rt.freeMemory();
14             System.out.println(m+" "+dims[i]+"     "+
       totalMemory+" B"+"     "+freeMemory+" B"+"     "+(
       totalMemory - freeMemory)+" B");
15      rt.gc();//mozemo predloziti poziv sakupljaca smeca
16          }     }    }
```

Mjerenje memorijske potrošnje programa.

Računanje memorijske potrošnje programa

Jedan mogući ispis:

```
Dimenzije     Ukupno M       Slobodno M      Koristena M
10000 10      67108864 B      64736944 B       2371920 B
10000 100     39845888 B      29672144 B       10173744 B
10000 1000    192937984 B     108687416 B      84250568 B
10000 4000    536870912 B     206080152 B      330790760 B
10000 8000    1052770304 B    394323224 B      658447080 B
10000 10000   1052770304 B    243004344 B      809765960 B
```

Zadatak za zadaću

Zadatak dz

Napišite program koji uspoređuje vrijeme izvršavanja programa koji računa $M^k, k \in \mathbb{N}$ i matricu $M \in \mathbb{R}_n, , \ n \in \mathbb{N}$. Napravite sekvencijalnu i višedretvenu verziju, izmjerite vrijeme izvršavanja i memorijsku potrošnju za razne dimenzije matrice M. Ponovite eksperiment za svaku dimenziju s puta i nacrtajte graf prosječnog vremena izvršavanja i memorijske potrošnje. Usporedite ta vremena s rezultatima kada $s = 1$.