CSCM12 – Software concepts and efficiency February 10th 2025
Manjiri Joshi & Cécilia Pradic

Lab 2: complexity analysis

- For sign-off, if the exercise is code, you are expected to show one working copy of the code. Otherwise, write down a worked solution (i.e., as you would in an exam) to show us.
- The questions marked as **Challenge** are going beyond what we expect from you. Kudos if you manage them, but they won't count towards extra marks on the coursework.

1. **$\mathcal{O}$-notation** For this task, I mostly want you to try to plot some functions and get a sense of what their asymptotic behaviour is and understand the $\mathcal{O}$ notation. You can use whatever software you prefer for this, but here I will give guidance for using the web based version of wolfram alpha, a computer algebra system that you can run in the browser by going to the following link `https://wolframalpha.com`. You may for instance plot the two functions $n^{\frac{3}{2}} \log(n)$ and n for n ranging from 2 to 100 by typing `plot n^(3/2) * log2(n), n from 2 to 1000`.

   (a) Plot the following functions (each on their own)

   $$n^2 \qquad 10{-}2^{-n} \qquad \sqrt{n} \qquad \log(n) \qquad n\log(n) \qquad n^2{+}n{+}\sqrt{n} \qquad n \qquad \frac{2^n}{5^{37}}$$

   (b) For each pair of functions (f, g) with f and g taken from the functions listed above, determine whether we have $f = \mathcal{O}(g)$. You do not need to provide a mathematical proof; if you have a doubt, you can plot two functions together to try to get an intuition as to what the answer should be.

   (c) (optional, but might help you understanding the definitions) Recall from the slide that whether $f = \mathcal{O}(g)$ can be determined by computing the limit $\lim_{n\to+\infty} \frac{f(n)}{g(n)}$: if the limit is a finite number, then $f = \mathcal{O}(g)$. You can compute limits in computer algebra systems. For instance, to compute $\lim_{n\to+\infty} \frac{5n^2+\log(n)\sqrt{n}}{3n^2-70n}$ in wolfram alpha, you can query `limit ((5n^2 + log2(n) * sqrt(n)) / (3n^2 -70n)) as n -> infinity`. Check at some of your answers for the previous question by computing limits of the form $\frac{f(n)}{g(n)}$ in a computer algebra system.

   (d) Recall that by definition, assuming that f and g are increasing functions, $f(n) = \mathcal{O}(g(n))$ if and only if there exists some constant K such that $f(n) \leq K(g(n) + 1)$ for every n.
   Find some constant K that witnesses that $2n^2 + n + \sqrt{n} = \mathcal{O}(n^2)$.

2. **Assessing time complexity of some functions** For each of the java function below, assess its asymptotic time complexity in the worst case scenario with a $\mathcal{O}$. In each of the following case, you may assume that the size of the input is the initial value of the variable n.

(a)

```
static void func1(int[] a, int[] r)
{
  int n = a.length;
  for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
      r[(i+1)*(j+1)-1] = a[i] * a[j];
}
```

(b)

```
static void func2(int[] a, int[] r)
{
  int n = a.length;
  for(int i = 0; i < Math.sqrt(n); i++)
      r[i] = a[i*i];
}
```

(c)

```
static int slt(int[] a)
{
  final int n = a.length;
  int r = 0;
  for(int i = n-1; i >= 0; --i)
    for(int j = 0; j < i; ++j)
      r += a[i] * a[j];
  return r;
}
```

(d)

```
static int wfn(int[][] a)
{
  final int n = a.length;
  int r = 0;
  for(int i = 0; i < n; ++i)
    r += a[i%2] * slt(a);
  return r;
}
```

(e)

```
static double naivePow(double a, int n)
{
  double res = 1;
  while(n > 0)
  {
    res *= a;
    n--;
  }
  return res;
}
```

(f)

```
static double evalPoly(double[] p, double v)
{
  int n = p.length;
  double r = 0;
  for(int i = 0; i < n; ++i)
    r += p[i] * naivePow(v, i);
  return r;
}
```

(g) **Challenge** Prove that the $\mathcal{O}$s you have are actually Θs.

3. **Challenge task (coursework question two years ago)** Call a user a *star* on a social media if they follow no one, but everyone else follows them. We want to find an algorithm such that, assuming that we are given as input a $n \times n$ matrix with `true` in cell (i, j) if user i follows user j and `false` otherwise, returns a user who is a star, or -1 if there is not any (by convention, let us say that users can't follow themselves so cells (i, i) can only contain `false`).

   (a) Give two examples of possible inputs with $n \geq 3$ users, one in which there is a star, and another where there is no star.

   (b) Is it ever possible to have two stars? Why?

   (c) Write a java function that solves the problem. It should have the following signature:

   ```
   static int findStar(boolean[][] follows)
   ```

   (d) What is the asymptotic time complexity of your solution? (in function of either the size (number of cells) or the dimension (i.e. number of rows/columns) of the input matrix)