LINEAR ALGEBRA AND VECTOR ANALYSIS

MATH 22B

Unit 12: Complexity

Seminar

12.1. We are heading to the first midterm. In a timed exam, **speed** is always an issue. As an exam writer it is therefore important to know whether a particular problem can be solved in the allocated time. Now, when solving a problem, one could be lucky and get a particular solution fast, by pure luck. As an engineer, it is important however to know how quickly the task can be finished in general. Estimating the speed is a rigorous game and usually asks for finding out how long the worst case takes. To do so we are rather rough and do not care whether an algorithm takes $100n$ steps or n steps. We care however whether the algorithm takes n steps (linear complexity) or n^2 (quadratic complexity) or e^n (exponential complexity).

Figure 1. Cheetah at Ree Park, Denmark, 24 April 2010, Malene Thyssen

12.2. Let us look at the **Landau big O** notation. Given two functions $f(n)$ and $g(n)$, we say $f(n) = O(g(n))$ as $n \to \infty$ if there exists a constant C such that $|f(n)| \leq C|g(n)|$ if n is large enough. If we say $f = O(g)$, we mean the limit $n \to \infty$. If $f = O(x^n)$ for some n, we say f has **polynomial growth** and if $f = O(e^x)$ but not better, we say f has **exponential growth**. A growth rate like $O(e^{\sqrt{x}})$ is an example of what one calls **sub-exponential growth**. We can also have **super exponential** growth, like $O(e^{x^2})$ or $O(e^{e^x})$. Sometimes one also writes $e^{O(x)}$ in order to characterize exponential growth. Now $e^{O(5x)} = e^{O(x)}$ but it is **not true** that $O(e^{5x})$ is equal to $O(e^x)$. If you see a statement using the big O notation, when in doubt, rewrite it using the definition.

12.3. For example $5x^2 + \sin(x) = O(x^2)$ or $7e^{5x} - 1000e^{3x} + x^{22} = O(e^{5x})$ or $5 + 1/(1 + x^2) + \cos(x^7) = O(1)$.

12.4. In **complexity theory**, the functions under consideration are usually a function of an integer n and $f(n)$ gives the number of computation steps which are needed to do the task for an object of size n. Let us first make sure we understand the big O notation:

Problem A: Determine from each of the following functions whether they show polynomial, exponential, sub-exponential (and super-polynomial), or super-exponential growth.
a) $f(n) = n^n$
b) $f(n) = e^{5\log(n)}$
c) $f(n) = \log(e^{n^7}) + \log(n)$.
d) $f(n) = n^3 \sin(e^n)$
e) $f(n) = e^{\sin(n)+n^7}$
f) $f(n) = 1/((1/n) + (1/\log(n))$
g) $f(n) = \log(\log(n^{10n}))$

12.5. How fast can we compute determinants of $n \times n$ matrices? Following the definition summing over all possible permutations is too costly as it requires to compute $n!$ products of n numbers. This appears to cost $O(n!n)$.

Problem B: Show that the Laplace expansion reduces the cost to $O(n!)$ multiplications. Is $O(n!)$ polynomial, sub-exponential, exponential or super-exponential?

12.6. In reality we can compute determinants much faster. Try out the following:

```
A=Table[Random[],{10000},{10000}]; Timing[Det[A]]
A=Table[Random[Integer,1],{1000},{1000}]; Timing[Det[A]]
```

Indeed, determinants can be computed in polynomial time. Let us explain:

Problem C: Why can we compute the determinant of a $n \times n$ matrix as fast as row reducing a $n \times n$ matrix? Can you estimate in the big O notation how fast determinants can be computed?

12.7. Related to determinants are **permanents**. The permanent of a matrix A is defined as

$$\mathrm{per}(A) = \sum_{\pi} A_{1\pi(1)} A_{2\pi(2)} \cdots A_{n\pi(n)} \ .$$

We see that they are defined like the by Leibniz defined determinants but the $\mathrm{sign}(\sigma)$ is missing. It looks like a small matter to leave away the -1 factors, but it changes the completely of the problem. The computation of permanents is believed to be **very hard**. An algorithm which does it in polynomial time is not unknown. If such an algorithm would exist, it would be a huge surprise or even a sensation.

12.8. Question: is there a fast way to determine whether a given orthogonal matrix has determinant 1 or -1? If you should find a fast way to do that let us know. It must be faster than just row reducing!

12.9.

Problem D: Time for Speed training! How fast can you compute the following determinants:

$$A = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 5 \\ 7 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 \\ 1 & 9 & 2 & 0 \\ 2 & 2 & 6 & 2 \end{bmatrix}, C = \begin{bmatrix} 7 & 2 & 5 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 2 & 2 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 3 & 4 \\ 2 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

12.10.

Problem E: Time for Speed training! How fast can you find the QR decomposition of the following matrices

$$A = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 5 \\ 7 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 3 & 4 & 2 & 1 \\ 0 & 5 & 3 & 1 \\ 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 \end{bmatrix}, C = \begin{bmatrix} 3 & -2 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 4 & -2 \end{bmatrix}, D = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 4 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 6 \end{bmatrix}.$$

12.11.

Problem F: Time for Speed training! How fast can you find a basis for the image of the following matrices?

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 2 & 4 & 6 & 8 \\ 4 & 8 & 12 & 16 \\ 8 & 16 & 24 & 32 \\ 16 & 32 & 48 & 64 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, D = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 8 & 10 \\ 4 & 8 & 10 & 12 \end{bmatrix}.$$

12.12.

Problem G: Time for Speed training! How fast can you find a basis for the kernel of the following matrices?

$$A = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \end{bmatrix}, B = \begin{bmatrix} 2 & 4 & 2 & 4 \\ 4 & 8 & 4 & 8 \\ 8 & 16 & 8 & 16 \\ 16 & 32 & 16 & 32 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, D = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 8 & 10 \\ 4 & 8 & 10 & 12 \end{bmatrix}.$$

Homework

Exercises A)-G) are done in the seminar. This homework is due on Tuesday:

Problem 12.1 There is a nice formula

$$A^{-1}_{ij} = (-1)^{i+j}\det(M(j,i))/\det(\mathrm{A}) \ ,$$

where $M(i,j)$ is the matrix in which the i'th row and jth column is deleted. The value $\det(M(i,j))$ is called the (i,j) **minor** of A.
a) First of all, this formula immediately follows from the Laplace expansion. Look at the formula at the bottom of the first page of lecture 11. Can you see it?
b) How fast can we compute the inverse of a $n \times n$ matrix using this formula?
c) Remember how we have computed the inverse of a matrix using row reduction. Does the determinant formula produce an advantage in terms of complexity?

Problem 12.2 How fast can we do the QR decomposition of a matrix? Is it polynomial or exponential?

Problem 12.3 In cryptology, it is important to be able to factor an integer x with n digits fast. Any significant progress there is monitored closely as somebody with a fast algorithm could bring down modern financial and communication systems.
a) How fast is the "Baby algorithm" which just tries out all factors up to $\sqrt{x}$. You have to give the answer in terms of $O(f(n))$ of the number n of digits of x.
b) You will find notions like $O(f(b))$, where b is the number of bits to represent an integer. Why can you replace the number of bits with the number of digits of a number?
c) Look up on the web how fast the fastest known integer factorization algorithm is.

Problem 12.4 The most important problem in computer science is the $P - NP$ problem. Describe this problem in one short paragraph.

Problem 12.5 What is "NP complete". List 3 examples of NP complete problems.