Data structures and algorithms
CM n°1
Overview - Analysis of algorithms
(cost measures, growth rates, asymptotic analysis)
Review - Standard sorting functions in C
(insertion sort, selection sort, mergesort, quicksort)

Christophe Tollu

A209 (poste 3691) - ct@lipn.univ-paris13.fr

Friday 21 Septembre 2023

Course organization and schedule

3h or 4,5 h per week for eleven weeks

- This week : 3h lecture (CM) ;
- Next nine weeks : 1,5h CM + 1,5h TD + 1,5h TP ;
- Last week : 3h TP.

Timetable

- Likely to change.
- You must check it every day !

Assessment

Written exam (Pa)

- No midterm written exam
- Three-hour final written exam in January 2024

Other forms of assessment (EvC)

- Homework assignment (C programming)
- Lab reports and/or tests in class (starting next week)

Combined assessment

- (EvC + 2*Pa)/3

Warning

- Conditions for success : assiduity and hard work !

Assessment (2nd chance)

Only written exam (Pa2)

- Two-hour written exam in June 2024

No combined assessment

- Pa2

Make sure you avoid 2nd chance

- Attend lectures (CM), practical work sessions (TD) and labs (TP)
- Ask questions if you do not understand, practice regularly
- Develop your skills in C programming

SDA on line

ENT : Course website

`https://moodlelms.univ-paris13.fr/course/view.php?id=2217`

Online resource

We will post

- News and announcements
- Slides of lectures
- Practice sheets and (perhaps) solutions
- Assignments
- Pieces of advice
- ...

- Check website every other day

Syllabus

Aims

- Get a good command of some linear and tree-like data structures, their implementation and the basic algorithms to manipulate them
- Be able to analyze and implement classical algorithms
- Develop your skills in structured C Programming

Highlights

- Linear data structures : queues, stacks, chained lists...
- Tree-like data structures
- Algorithms for sorting and searching
- Hashing
- Asymptotic analysis of algorithms
- ...

Basic bibliography

Algorithms

- Donald E. Knuth, The Art of Computer Programming (4 volumes), Addison-Wesley, 1968–
  [Volumes 1-3 are in their 2nd or 3rd editions, Volumes 1 and 3 of particular interest to us]
  https://en.wikipedia.org/wiki/The_Art_of_Computer_Programm
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, MIT Press, 1990
  [Now in its fourth edition, 2nd edition translated into French]
  https://en.wikipedia.org/wiki/Introduction_to_Algorithms

Programming in C

- Claude Delannoy, Le livre du C premier langage, Eyrolles, 2002
- Brian Kernighan and Dennis Ritchie, The C Programming Language, Prentice Hall, 1978
  https://en.wikipedia.org/wiki/The_C_Programming_Language

Today

Overview of algorithm complexity

- Cost measure(s)
- Growth rates and scales
- Bachmann-Landau notation for comparing growth rates

Review of array sorting by key comparison

- Selection and insertion
- Mergesort
- Quicksort

Time and space complexity

Time complexity

- How long does it take to run a given program ?
- Answer is unreliable because depends too much on machine and environment
- Instead concentrate on the number of (specific, possibly weighted) instructions required by the execution w.r.t. some relevant parameters
- Usually, relevant parameters are size and/or values of inputs

Space complexity

- What is the amount required to run a given program ?
- More on space complexity later

Time complexity

Cost measure / function

- Decide operations / instructions to consider for cost
- Possibly assign specific weights to different operations
- Decide parameters to serve as entries of cost function

Examples

- When study searching algorithm, parameter = $\sharp$ values in search space and cost = $\sharp$ value accesses
- When study comparison-based sorting algorithms, parameter = $\sharp$ of values to be sorted and cost = $\sharp$ value comparisons and assignments (or exchanges)

Cost growth

- We focus on growth rate of cost function when parameters $\to \infty$
- We thus need tools to compare growth rates

Two main models

- Worst-case complexity : For arbitrary value(s) of parameter(s), consider maximum cost over all instances
- Average-case complexity : For arbitrary value(s) of parameter(s), consider mean of cost (for chosen probability distribution) over all instances

Big "O" notation (dominance)

Definition

- Let $f = (f_n)_{n\geq 0}$ and $g = (g_n)_{n\geq 0}$ two $\mathbb{R}_+$-valued sequences
- (Think f and g as cost functions, so generally $f_n, g_n \geq 0$)
- One writes $f \in O(g)$ (and reads "f is dominated by g when $n \to \infty$") if $(\exists C > 0)$ $(\exists n_0 \in \mathbb{N})$ $(\forall n \geq n_0)$ $(|f_n| \leq C\,|g_n|)$

Variant definition and basic properties

- if g_n is ultimately > 0, then $f \in O(g)$ is equivalent to : f_n/g_n is ultimately bounded from above
- Definition can be generalized to multivariate sequences
- Transitivity : if $f \in O(g)$ and $g \in O(h)$, then $f \in O(h)$
- **As all sequences are positive**, dominance is compatible with addition and multiplication

Little "o" notation (negligibility)

Definition

- Let $f = (f_n)_{n \geq 0}$ and $g = (g_n)_{n \geq 0}$ two $\mathbb{R}_+$-valued sequences
- (Think f and g as cost functions, so generally $f_n, g_n \geq 0$)
- One writes $f \in o(g)$ (and reads "f is negligible w.r.t g when $n \to \infty$") if $(\forall \epsilon > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ (|f_n| \leq \epsilon\, |g_n|)$

Variant definition and basic properties

- if g_n is ultimately > 0, then $f \in o(g)$ is equivalent to : $f_n / g_n \to 0$ when $n \to \infty$
- Definition can be generalized to multivariate sequences
- if $f \in o(g)$, then $f \in O(g)$
- Transitivity : if $f \in o(g)$ and $g \in O(h)$, then $f \in o(h)$
- **As all sequences are positive**, negligibility is compatible with addition and multiplication

Another useful notation : Ω

Definition

- Let $f = (f_n)_{n\geq 0}$ and $g = (g_n)_{n\geq 0}$ two $\mathbb{R}_+$-valued sequences
- (Think f and g as cost functions, so generally $f_n, g_n \geq 0$)
- One writes $f \in \Omega(g)$ if
  $(\exists C > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ (|f_n| \geq C\,|g_n|)$

Variant definition and warning

- $f \in \Omega(g)$ is equivalent to : $g \in O(f)$
- if g_n is ultimately $\neq 0$, then $f \in \Omega(g)$ is equivalent to :
  $|f_n/g_n|$ is ultimately bounded from below by a constant > 0
- Notation introduced by Knuth in his pioneering work on analysis of algorithms, not to be confused with Ω notation used in number theory

Yet another useful notation : Θ

Definition

- Let $f = (f_n)_{n \geq 0}$ and $g = (g_n)_{n \geq 0}$ two $\mathbb{R}_+$-valued sequences
- (Think f and g as cost functions, so generally $f_n, g_n \geq 0$)
- One writes $f \in \Theta(g)$ if
  $(\exists C, c > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ (c\,|g_n| \leq |f_n| \leq C\,|g_n|)$

Variant definition and basic properties

- $f \in \Theta(g)$ is equivalent to : $f \in O(g)$ and $f \in \Omega(g)$
- Equivalence relation ("same order of asymptotic growth")

Asymptotic equivalence : $\sim$

Definition

- Let $f = (f_n)_{n \geq 0}$ and $g = (g_n)_{n \geq 0}$ two $\mathbb{R}_+$-valued sequences
- (Think f and g as cost functions, so generally $f_n, g_n \geq 0$)
- One writes $f \sim g$ if
  $(\forall \epsilon > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ (|f_n - g_n| \leq \epsilon, |g_n|)$

Variant definition and basic properties

- $f \sim (g)$ is equivalent to : $f - g \in o(g)$
- if g_n is ultimately $\neq 0$, then $f \sim g$ is equivalent to :
  $|f_n/g_n| \to 1$ when $n \to \infty$
- Equivalence relation
- if $f \sim g$, then $f \in \Theta(g)$
- Computer scientists often prefer the Θ notation

Growth rates

$\Theta(1)$	Constant
$\Theta(\log n)$	Logarithmic
$\Theta(n)$	Linear
$\Theta(n \log n)$	Linearithmic
$\Theta(n^2)$	Quadratic
$\Theta(n^k)$ with k constant	Polynomial
$\Theta(k^n)$ with $k > 1$ constant	Exponential

Échelles de comparaison

Définition

- Relation de prépondérance (sur l'ensemble des suites réelles positives) : g est **prépondérante par rapport à** f ssi $f \in o(g)$ (autres notations : $f \ll g$, $f \prec g$).
- Une **échelle de comparaison** est un ensemble de suites (réelles positives) de référence totalement ordonné par la relation de prépondérance.

Exemples

- $\{(n^\alpha)_{n\geq 0} \mid \alpha > 0\}$
- $\{(n^\alpha \log^\beta n)_{n\geq 0} \mid \alpha, \beta > 0\}$
- $\{(2^{Cn^\gamma} n^\alpha \log^\beta n)_{n\geq 0} \mid \alpha, \beta, \gamma > 0, C > 0\}$