IFT2125 - Introduction to algorithms

Introduction

Pierre McKenzie

DIRO, University of Montreal

Fall 2017

Documentation

Book required:

Brassard and Bratley, *Fundamentals of algorithmics,* Prentice Hall, 1996.

Other:

Cormen, Leiserson, Rivest, *Introduction to Algorithms,* 1994, or +.
English Edition of 2009 features 4th author, Stein

Kleinberg, Tardos, *Algorithm Design,* 2006

Transparencies, notes, point references placed on Studium

Web

Library (reserve, many books)

Evaluation

Homework :

- 4 or 5 assignments of up to 4 questions
- each question evaluated out of 10 regardless of its difficulty
- teams of 2 recommended

Intra and final examinations:

- Closed Book
- Cumulative Final

Scale:

- Intra 30%, final 40%, homework 30%.
  Threshold at 40%.
- Undergraduate students: intra 40%, final 60%.
  For doctoral students: homework does not count.

Page 4

Teaching Assistant

Maëlle Zimmermann
maelle.zimmermann@umontreal.ca

Tasks:

- practice sessions
- corrects homework
- answers questions
- available by appointment

Page 5

Book BB = B and B rassard Ratley

Page 6

Useful to have seen

Prerequisite IFT1065 - Discrete Mathematics:

- Mathematical Induction [BB 1.6]
- Logic, propositional, predicates [BB 1.4.5]
- Permutations, combinations [BB 1.7.3]
- Modular arithmetic, polynomials
- Definitions of $O\ (f\ (n))\ \Omega\ (f\ (n))\ \Theta\ (f\ (n))$ [BB 1.7.2, 3.2, 3.3]
- Simple and linear homogeneous recurrences [BB 4.7.2]

Prog. and concomitant IFT2015 - Data Structures:

- Dichotomous research
- Some sorts
- Python (?)

Concomitant IFT1978 - Probability and statistics:

- Basic probabilities (BB 1.7.4)

Page 7

Approximate Map

hours courses	material
4	Introduction and examples (partly outside book)
4	Supplements on orders and recurrences (Chapters 3 and 4)
5	Voracious Algorithms (Chapter 6)
5	Divide and rule (Chapter 7)
5	Dynamic programming (Chapter 8)
5	Exploring graphs (Chapter 9)
5	Probabilistic algorithms (Chapter 10)
2	Parallel Algorithms (Chapter 11)
2	Selected Topics

TOTAL: 37

Page 8

Take notice of the honor code of student DIRO . In particular,

cite all sources of information used in your work

assigning a duty as a team commits the responsibility of the team.

For more information on university regulations, consult
Integrity at the Université de Montréal .

Questions ?

What is algorithmics?

to design effective methods for solving computational problems

choose the appropriate method for a given problem

Much intelligence over the years devoted to algorithmics!

Page 11

Examples

Sorting a table

1. Selection
2. Insertion
3. Merge out - by merger
4. Quick sort - fast
5. Heap out - by heap
6. Radix released - by base
7. Bucket out - by packets
8. Alouette sort - by alouettes :-)
9. Sorting in parallel

Page 12

Examples

Determining an $m \times m$ matrix

$$\begin{vmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{vmatrix} = X_{11} x_{22} x_{33} - x_{11} x_{32} x_{23} - x_{12} x_{21} x_{33} + \cdots$$

$$= \sum_{\sigma \in S_m} (-1)^{\operatorname{sign} \sigma} \, {}_{1\,\sigma} x_{(1)} \, x_{\sigma\, 2\,(2)} \cdots x_{m\sigma\,(m)}$$

1. Stupid
   Make the sum of $m!$ terms.
2. Gauss-Jordan
   Bring it to the triangular shape.
   Multiply the elements of the diagonal.
3. Berkowitz (Samuelson)
   Reduce to matrix power calculation.

Searching: and the permanent of a matrix?

Page 13

Examples

Determining the primality

We want to determine if $\underbrace{xxxxxxxxxxxx}_{12 \text{ digits}}$ is a prime number.

An addictive problem for mathematicians.
Basic problem for cryptographers.

1 Stupid
Try $2, 3, 4, 5, 6, 7, 8,$ ... optionally up to 10^6

2 Erathostenes Screen
Eliminate alternately divisors of the list $2, 3, 4, 5, 6, 7, 8$ ..., 10^6

3 Miller-Rabin
Quick accepting a probability of error of less than $2^{-1000000}$.

4 Agrawal-Keyal-Saxena (2002)
Polynomial time with certainty but high degree.

Page 14

Examples

maximum Stable

Data: graph (S, A)

Determine: a stable maximum size.

1 Stupid
Try all $E \subseteq S$ descending size.

Searching: find a method that is not stupid.

Page 15

design effective solving computational problems methods

choose the appropriate method for a given problem

In fact, what is a problem?

Page 16

Problems and examples ("problems and instances")

A problem request

- calculating a value
  - eg sorting, determining
- or answer a yes / no question
  - eg the number is prime, there is a stable of size *k?*

from data input.

Page 17

A problem has an infinite number of copies

- ex: sorting
  - Table 1, Table 2, etc.
- ex: first number
  - *0, 1, 2, ...,* etc.
- ex: stable
  - ... ...

Page 18

The weather

An algo *A P* solves a problem.

A can take a different time on each copy

What when the execution time of *A?*

Simplification:

parameterize a function of size n copies

And what size n of a copy?

Ultimately, n = number of bits used to encode the copy.

In practice, depends on P and purpose of the analysis:

ex: sorting

- often n = number of array elements

eg evaluating an expression like $((28783 + 410) / 192) \times 159$

- often n = number of operands, here $n = 4$
- sometimes n = total number of digits, here $n = 14$.

ex: stable

- sometimes n = number $|S|$ of vertices in the graph (S, A)
- sometimes $n = |S|^2$ = number of bits required to represent the matrix adjacency of the graph

Various time measures

Worst Case (Most Used Measure):

$$t(n) = \max_{e \text{ copy of size } n} \{\text{time it takes } A \text{ on } e\}$$

On average

$$t(n) = \frac{\sum_{e \text{ copy of size } n} \{\text{time it takes } A \text{ on } e\}}{\text{number of copies of size } n}$$

Cushioning (case of A that acts on external data)

Averaged over several successive calls to A

Hoped (case of A that uses Shuffle)

Mathematical expectation of time before stopping.

4 slides borrowed from Sylvie Hamel

How to obtain the time $t(n)$ of an algorithm?

Method 1: Experimental studies

Implement the algorithm in Java (or other)

Run the program with size entries
and different composition

Use a method to get
a real measure of time
execution

Draw the graph of the results

9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0 50 100
Input Size

Limitation of this method

We must implement the algorithm

We want to know the time complexity of an algorithm before
to implement it, saving time and money

The results are not representative of all
the entrees

To compare two different algorithms for the same problem, the same environment must be used (hardware, software)

Method 2: Analytical

Including the elementary operations:

Basic operations performed by the algorithm

- Evaluate an expression
- Assign a value to a variable
- Calling a method
- Increment a counter
- etc.

Independent of the chosen programming language

Assume that each takes a constant execution time

Example: max of a table

Count basic operations

Inspecting the pseudocode of an algorithm can determine the maximum number of transactions performed by an algorithm, such as depending on the size of the entry

Algorithm	# operations	
Algorithm ***arrayMax (A, n)***		
currentMax ← ***A*** [0]	2	2
for *i* ← 1 **to** *n* - 1 **do**	2 ***n***	2 (n-1)
if *A [i]*> Then *currentMax*	2 (***n*** - 1)	2 (n-1)
currentMax ← ***A [i]***	2 (***n*** - 1)	2 (n-1)
{Increment counter ***i***}	2 (***n*** - 1)	
return *currentMax*	1	1
	Total = 8 ***n*** - 2	total = 6n - 3

The number of elementary operations in time

T is sought (n) = time in worst case, when n = array size

The algo executes *arrayMax* $6n - 3$ elementary operations in worst case

- a = execution time of the fastest elementary operation
- b = execution time of the slowest elementary operation

So the calculation time $t(n)$ *arrayMax* checks:

$$\forall n,\ a \times (6n - 3) \leq t(n) \leq b \times (6n - 3)$$

Often the asymptotic behavior is sufficient. Right here :

- of $\leq t$ is pulled $(n) \in O(n)$
- $\geq t$ of pulling $(n) \in \Omega(n)$
- $t(n) \in O(n) \cap \Omega(n)$ where $T(n) \in \Theta(n)$

arrayMax is particular in that the worst case is easy to identify

Page 27

Other examples of intelligence devoted to algorithms

Page 28

Multiplication of large integers

Several methods to calculate $xxxxxxxxxxxx \times YYYYYYYYYYYY$

1. classical
2. In the "Russian way"
3. In the "Arabic way"

Page 29

Multiplication of large integers

Several methods to calculate *xxxxxxxxxxxx* × *YYYYYYYYYYYY*

1. classical
2. In the "Russian way"
3. In the "Arabic way"
4. recursive
   Express *xxxxxxxxxxxx* = $A + B \times 10^{6}$
   Express *YYYYYYYYYYYY* = $C \times 10^{6} + D$
   Calculating $AC \times 10^{12} + (AD + BC) \times 10^{6} + BD$.

Page 30

Multiplication of large integers

Several methods to calculate *xxxxxxxxxxxx* × *YYYYYYYYYYYY*

1. classical
2. In the "Russian way"
3. In the "Arabic way"
4. recursive
   Express *xxxxxxxxxxxx* = $A + B \times 10^{6}$
   Express *YYYYYYYYYYYY* = $C \times 10^{6} + D$
   Calculating $AC \times 10^{12} + (AD + BC) \times 10^{6} + BD$.
5. Interpolation
   Calculate *xxxxxxxxxxxx* × *YYYYYYYYYYYY* modulo 2
   Calculate *xxxxxxxxxxxx* modulo 3 × *YYYYYYYYYYYY*
   Calculate *xxxxxxxxxxxx* × *YYYYYYYYYYYY* modulo 5
   ... ...

   Calculate $m \leq 2 \times 3 \times 5 \times \cdots$ satisfying these congruences.
   Theorem ("Chinese remainders"): This m (positive) is unique.

Page 31

Greatest common divisor

GCD is sought *(xxxxxxxxxxxx, YYYYYYYYYYYY)*.
EG: *gcd (140,* 98) × 2 = *gcd (70,* 49) = 2 × 7 × *gcd (10,* 7) = 2 × 7 = 14.

1 Stupid
Try *xxxxxxxxxxxx*, *xxxxxxxxxxxx - 1*, ... possibly up to 2

2 Euclid

```
def pgcd (a, b):
    while b! = 0:
        a, b = b, a% b
    return (a)
```

Searching: effective method in parallel?

Greatest common divisor

How to ensure that

```
def pgcd (a, b):
    while b! = 0:
        a, b = b, a% b
    return (a)
```

is a correct algorithm?

rarely simple
ingenuity demand

Draw more from Euclid's algorithm

Extended Euclidean algorithm

Page 34

Fourier Transform

Given a particular form of matrix M and a vector x, we want calculate Mx and $M^{-1}x$ vector.

1. Stupid
   Multiply without worrying about the particular shape.
2. Divide and rule
   A revolutionized telecommunications and signal processing.
   Underpins the JPEG format.

Page 35

Belonging to a group of permutations

Reminder: permutations of a set $\{1, 2, 3, 4, 5, 6\}$ "points"

$$\varepsilon = \begin{pmatrix} 1\,2\,3\,4\,5\,6 \\ 1\,2\,3\,4\,5\,6 \end{pmatrix} \in S_6 \text{ is the identity permutation}$$

$$\begin{pmatrix} 1\,2\,3\,4\,5\,6 \\ 4\,2\,6\,5\,1\,3 \end{pmatrix} \in S_6 \text{ is also shown } (145)\,(36)$$

The product of two permutations:
(145) (26) * (134) (256) = (1) (2) (3465) = (3465)

The inverse of a permutation:
$[(145)\,(26)]^{-1} = (541)\,(62) = (154)\,(26)$

Page 36

The problem of belonging

Data: $p, p_1, \ldots, p_k \in S_m$
Determine if $p \in \underbrace{\langle P_1, \ldots, {}_k p \rangle}_{\text{all permutations generated by composition of } p_i}$.

1 Stupid
2 Clever
Next transparencies.
3 Super-smart
Fast in parallel, relies on 5000 pages of mathematics.

Page 37

Belonging: the stupid algo

$S \leftarrow \varnothing$
$S \leftarrow \{p_{1}, p_{2}, \ldots, p_{k}\}$

while $S = S$
$S \leftarrow S$
$S \leftarrow S \cup \{s * t: s, t \in S\}$

if $p \in S$ Then TRUE else FALSE

Page 38

Membership: intelligent algo

Heart of the algorithm: screening a permutation in a table construction

T: $m \times m$ array of permutations of the points $\{1, 2, \ldots, m\}$
r: permutation to be treated

sift (r)
while ε $r =$
$i \leftarrow$ min $\{i: i = i\}_{r}$ $\{I \leftarrow$ smallest point moved $r\}$
$\leftarrow i\, j_{r}$ $\{J \leftarrow$ the point where r i send$\}$
if $T[i, j] ==$ Then ε
$T[i, j] \leftarrow r$ {insert} r in table
else
$r \leftarrow r * (T[i, j])^{-1}$

Page 39

Membership: the complete intelligent algorithm

Data: $p, p_1, \ldots, p_k \in S_m$
Determine if $p \in \langle p_1, \ldots, p_k \rangle$.

fill $m \times m$ table T everywhere with ε

for $i = 1, \ldots, k$
    sift (p_i)

while there exist q, r in table $T * r\ q$ Such That Was never sifted
    sift $(q * r)$

if sift (p) Then modified T
    "P is not the group $\langle p_1, \ldots, {}_k\ p \rangle$"
else
    "Belongs to the group $p \langle p_1, \ldots, {}_k\ p \rangle$"

Membership: the intelligent algorithm

Is this algorithm correct? ? ?

Membership

Main property of *Dr.* screening, $r = \varepsilon$

Suppose:

r has been sifted

s was not the smallest point set by r

$T\ [a, j]$ = entry was last discussed at the *actual* screening.

So :

$s \leq t$

r now expressed as

$T[t, j] * T[t-1, j_{t-1}] * \cdots * T[s+1, j_{s+1}] * T[i, j_s]$.

Proof: induction on the number of revolutions of the while while sieving.

Problem of the order of a group of permutations

It is free from Table T

Data: $_1$ permutations $p, p_2, \ldots, p_k$

Determined: number of permutations of the group $<p_1, p_2, \ldots, p_k>$

form T sieving $p_1, p_2, \ldots, p_k$ and then by "closing" T

N ← 1

for $i = 1, \ldots, m$

    $l \leftarrow |\{j: T[i, j] = \varepsilon\}|$

    N ← N x $(l + 1)$

return N