The importance of algorithms in Computer Science

- or -

how Manao learned how to defeat all the monsters.

Alexandra Silva

alexandra@cs.ru.nl

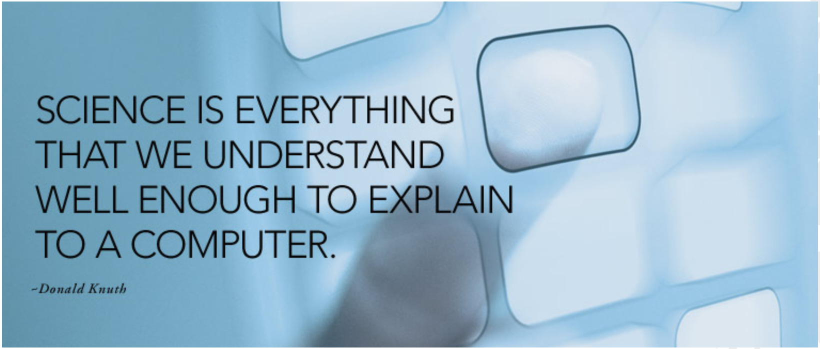
http://www.cs.ru.nl/~alexandra

Institute for Computing and Information Sciences
Radboud University Nijmegen

20th August 2013

Computers are everywhere

Computer Science



Algorithms

Algorithms = Human (high-level) way of teaching a computer what to do to solve a problem

One algorithm can change the world

One algorithm can change the world

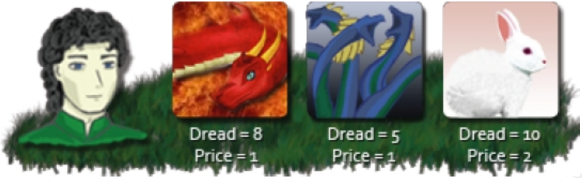
Why was the Google algorithm successful?

Solved a problem – search – in an *elegant*, *efficient*, *correct* and *scalable* way.

Manao and the monsters

- Manao is traversing a valley inhabited by monsters.
- During his journey, he will encounter several monsters.
- The scariness of each monster is a number *dread/scare*.
- Manao is not going to fight the monsters (*watje*).

Manao and the monsters

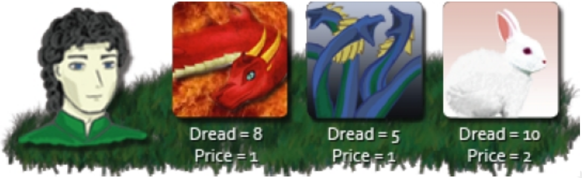


- Instead, he will bribe some of them and make them join him.
- The monsters are not too greedy (price $= 1$ or 2).

Manao and the monsters

- Manao meets monster: bribe or not bribe?
- No bribe, not scary enough: monster will attack!
- Bribe: monster will join the party!
- So: if monster is stronger, Manao bribes.
- If monster is not strong: choice.

Manao and the monsters: example



- Dragon: bribe (always for the first monster).
- Hydra: Manao can either pass or bribe her.
- Killer Rabbit:
  1. Bribed Hydra: total scariness $= 13 > 10$. Pass the rabbit! Total cost $= 1 + 1 = 2$ coins.
  2. Bribe the Rabbit. Total cost $= 1 + 2 = 3$ coins.

Manao and the monsters: version 1

- There are between 1 and 20 monsters.
- Their scariness level is between 1 and 2,000,000,000, inclusive.
- The bribe for each monster will be either 1 or 2.

Manao and the monsters: version 1

- There are between 1 and 20 monsters.
- Their scariness level is between 1 and 2,000,000,000, inclusive.
- The bribe for each monster will be either 1 or 2.

Goal

Minimize the amount of money Manao needs to pay.

How do we solve it?

- Monster 0: no option, we bribe.
- Monster 1: two options.
  1. Not bribe: we are scary enough, money and scariness $\Longleftrightarrow$.
  2. bribe: scariness and money spent $\Uparrow$.

  $\vdots$
- Monster p: we have passed the first p monsters and we have a total scariness level of *currentScare*.

How do we decide what to do?

How do we solve it?

- Monster p: Bribe or scare?
  We will compute the minimum price at step p given how scary we are – $M(p, CurrentScare)$ – by analyzing our two options:
  1. bribe: it will cost us $price_p$, we earn $scare_p$.
  $$P_1 = price_p + M(p+1, CurrentScare + scare_p)$$
  2. scare: $currentScare > scare_p$, not pay, not scarier.
  $$P_2 = M(p+1, CurrentScare)$$

Choose best option: $M(p, CurrentScare) = \min(P_1, P_2)$.

How do we solve it?

In the solution:

- We were using the minimum price of the *next* monster – recursive algorithm.

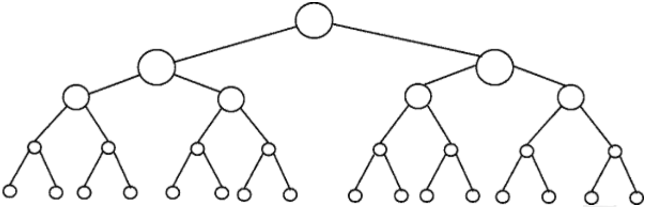
How do we solve it?

In the solution:

- We were using the minimum price of the *next* monster – recursive algorithm.
- When do we stop: $p = n$ – all monsters done, price is 0.

How *efficient* is this algorithm?

Let's measure how many options we have:



At each step we double our options. For n monsters we have

$$\underbrace{2 \times 2 \times \cdots \times 2}_{n \text{ times}}$$

How *efficient* is this algorithm?

$$\underbrace{2 \times 2 \times \cdots \times 2}_{n \text{ times}} = 2^n$$

For 20 monsters we have $2^{20} \sim 1.000.000$.

How *efficient* is this algorithm?

$$\underbrace{2 \times 2 \times \cdots \times 2}_{n \text{ times}} = 2^n$$

For 20 monsters we have $2^{20} \sim 1.000.000$.

Good or bad?

Manao and the monsters: version 2

A, dread = 10
B, dread = 2^10
D, dread = 10
E, dread = 10
F, dread = 10
H, dread = 10
J, dread = 10
L, dread = 10
M, dread = 10
P, dread = 10
P, dread = 10
R, dread = 10
T, dread = 10
.....

There can be now up to **50** monsters

Manao and the monsters: version 2

Is our previous algorithm ok?

Manao and the monsters: version 2

Is our previous algorithm ok?

$2^{50} =$ **VERY BIG NUMBER**

Manao and the monsters: version 2

Is our previous algorithm ok?

$2^{50} =$ **VERY BIG NUMBER**

Actually, with 2^{50} grains of rice you can cover the whole of the Netherlands with ca 2 meters of rice (calculations in Henk Barendregt's kitchen!).

Manao and the monsters: version 2

Can we do better?

Manao and the monsters: version 2

Can we do better?

Different perspective: what is the maximum price to pay for n monsters?

Manao and the monsters: version 2

Can we do better?

Different perspective: what is the maximum price to pay for n monsters? $2 \times n$

Manao and the monsters: version 2

Can we do better?

Different perspective: what is the maximum price to pay for n monsters? $2 \times n$

Instead of minimizing, ask...

For each price p from 0 to $2 \times n$, can we cross without paying more than p?

For $n \leq 50$ there are at most 101 values for p and we can answer the problem *fast* (in polynomial time).

Manao and the monsters: version 2

Can we do better?

Different perspective: what is the maximum price to pay for n monsters? $2 \times n$

Instead of minimizing, ask...

For each price p from 0 to $2 \times n$, can we cross without paying more than p?

For $n \leq 50$ there are at most 101 values for p and we can answer the problem *fast* (in polynomial time).

solution = chocolate bar!

Topcoder

Manao and the monsters appeared in a programming match of TopCoder. Both problems were solved in under 4 minutes!

Lessons to take home

- The parameters of the problem matter.
- Solutions often need to be adapted.
- A lot of improvement can be done before implementation.

Lessons to take home

- The parameters of the problem matter.
- Solutions often need to be adapted.
- A lot of improvement can be done before implementation.
- It is all about asking the right questions!

Lessons to take home

- The parameters of the problem matter.
- Solutions often need to be adapted.
- A lot of improvement can be done before implementation.
- It is all about asking the right questions!

A lot of Computer Science is like solving puzzles!