UPPSALA UNIVERSITET

MA1: L03

Tom Smedsaas

Introduction to complexity analysis

UPPSALA UNIVERSITET

Compute x^n iteratively

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x \cdot x \cdot x \cdot \ \ldots \ \cdot x & \text{if } n > 0 \end{cases}$$

```
def power(x, n):
    result = 1
    for i in range(1, n+1):
        result *= x
    return result
```

The function makes *n* multiplications so the time grows linearly with *n* independently of *x*.

UPPSALA UNIVERSITET

Compute x^n *recursively*

```
def power(x, n):
    if n == 0:
        return 1
    else:
        return x*power(x, n-1)
```

Let *t*(*n*) stand for the time the call `power(x, n)` will take. Then

$$t(n) = \begin{cases} c & \text{if } n = 0 \\ d + t(n-1) & \text{if } n > 0 \end{cases}$$

$$t(n) = d + t(n-1) = d + d + t(n-2) = d + d + d + t(n-3) = \ \ldots$$

$$\ldots = d \cdot n + t(0) = d \cdot n + c$$

UPPSALA UNIVERSITET

A more efficient way to compute x^n

Suppose we want to compute x^{16}.

If we start by calculating x^8 then a squaring is enough to get x^{16}

and x^8 can be calculated by squaring x^4

and x^4 can be calculated by squaring x^2

and x^2 can be calculated by squaring x.

Thus, x^{16} can be calculated with 4 multiplications instead of 16.

How can we formulate this idea into a general algorithm?

UPPSALA UNIVERSITET

A more efficient way to compute x^n

Suppose that *n* is even and ≥ 0:

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ (x^{n/2})^2 & \text{if } n > 0 \end{cases}$$

If *n* is odd then is *n* -1 even:

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ (x^{n/2})^2 & \text{if } n > 0,\ n \text{ even} \\ x \cdot (x^{(n-1)/2})^2 & \text{if } n > 0,\ n \text{ odd} \end{cases}$$

A more efficient way to compute x^n

```
def power(x, n):

    def sqr(x):
        return x*x

    if n<0:
        return 1./power(x, -n)
    elif n==0:
        return 1
    elif n%2==0:
        return sqr(power(x, n//2))
    else:
        return x*sqr(power(x, (n-1)//2))
```

$$\begin{cases} 1 \\ (x^{n/2})^2 \\ x \cdot (x^{(n-1)/2})^2 \end{cases}$$

UPPSALA UNIVERSITET

How many multiplication is done by this algorithm?

If *n* is an even power of 2, i.e. if $n = 2^k$?

Requires $k = \log_2 n$ multiplications

If *n* not is en even power of two ?

Requires at most one extra multiplication before the problem is halved i.e. at most $2 \log_2 n$ multiplications.

Approximately how many multiplications are required to calculate x^{1000} and $x^{1000000}$ respectively?

UPPSALA UNIVERSITET

Searching a list

Searching for a value in a Python list is normally done with the `in` operator e.g. with the expression `if x in lst`

Even if we don't see it, behind the scene there must be code like this:

```
def search(x, lst):
    for e in lst:
        if e == x:
            return True
    return False
```

The work is thus, at least in the worst case, proportional to the length of the list.

The operator `in` and the function `search` has the same samma *complexity* even if the operator `in` for sure is much faster.

UPPSALA UNIVERSITET

More efficient searching

If the data in the list is sorted, the search can be made significantly more efficient :

if x < the value in the middle:
search in the left part
else:
search in the right part

The method is called binary search it is easy to implement both recursively and iteratively.

UPPSALA UNIVERSITET

Complexity for binary search

The method halves the search set for each iteration or recursive call.

$$n \longrightarrow \frac{n}{2} \longrightarrow \frac{n}{2^2} \longrightarrow \frac{n}{2^3} \longrightarrow \ldots \longrightarrow \frac{n}{2^k}$$

When is $\frac{n}{2^k} = 1$?

$$n = 2^k \Leftrightarrow k = \log_2 n$$

How many iterations/function calls are needed to search a list of $10^6, 10^9$ and 10^{12} elements?

UPPSALA UNIVERSITET

Algorithm- or complexity analysis

Studies how the time of an algorithm depends on the input data.

Results of the type:

time grows proportionally to the square of the number of elements

or

time is constant independent of input

or

time grows exponentially with problem size

UPPSALA UNIVERSITET

The end

.