Datavetenskap
Datastrukturer och algoritmer
Johan Eliasson
johane@cs.umu.se



# Komplexitetsanalys

## Er uppgift

Att bestämma tidskomplexiteten för de nedanstående algoritmerna. Detta ska göras genom att räkna antalet *primitiva operationer* som utförs i varje algoritm. Utifrån detta ska sedan definitionen av *Ordo* användas för att ge uttryck för varje algoritms tidskomplexitet. Konstanterna $c$ och $n_0$ skall bestämmas. För ett exempel på hur en analys kan se ut titta på föreläsningsanteckningarna. Tänk på att fundera kring om det finns ett *worst-case* och ett *best-case* och hur de i så fall ser ut.

## Algoritmer

***Summera talen 1 till n***

```
Algorithm sumN(n)
input: A number n
output: The sum of the numbers 1 to n

    sum ← 0
    for i ← 1 to n do
        sum ← sum + i
    return sum
```

***Summera alla udda tal mellan 1 och n***

```
Algorithm sumN(n)
input: A number n
output: The sum of the numbers 1 to n

    sum ← 0
    i ← 1
    while i <= n do
        sum ← sum + i
        i ← i + 2
    return sum
```

***Linjär sökning***

```
Algorithm linearSearch(v, n, num)
input: A vector v containing numbers
      n is the length of the vector v
      A number num to be found in v
output: The index of num in the vector v or -1 if not found

    index ← -1
    i ← 0
    while (index == -1 and i < n)
        if v[i] == num then
            index ← i;
        i ← i + 1
    return index
```

***Naiv bubblesort***

```
Algorithm bubblesort(arr)
Input: An array to be sorted
Output: The sorted array
    repeat
        swapped ← false
        for j ← low(arr) to high(arr)-1 do
            if arr[j] > arr[j+1] then
                temp ← arr[j]
                arr[j] ← arr[j+1]
                arr[j+1] ← temp
                swapped ← true
    until not swapped
    return arr
```

***Beräkna $x^n y^m$***

Nedan finns det två olika algoritmer math1 och math2 som båda löser samma problem. Om du granskar de två algoritmerna (du behöver inte räkna operationerna i denna uppgift utan se på det mer övergripande), vilken av algoritmerna har lägst tidskomplexitet? Varför?

```
Algorithm math1(x, y, n, m)
Input: x, y number to be multiplied
       n, m how many multiplications that should be done
Output: x^n*y^m
    res ← 1
    for i ← 1 to n do
        res ← res * x
    for i ← 1 to m do
        res ← res * y
    return res


Algorithm math2(x, y, n, m)
Input: x, y number to be multiplied
       n, m how many multiplications that should be done
Output: x^n*y^m
    return pow(x, n)*pow(y, m)

Algorithm pow(x, n)
Input: x number to be multiplied
       n how many multiplications that should be done
Output: x^n
    if n = 0 then
        return 1
    temp ← pow(x, n/2)
    if (n%2 = 1) then
        return x*temp*temp
    else
        return temp*temp
```