Information, Computation, and Communication

Training

Representative Algorithms

Complexity	Example Behavior	Example Algorithm
$\Theta(\log n)$	Look at only log n elements of the input. Split the input into at least half (or more)	Binary search (Recherche dichotomique)
$\Theta(n)$	Look at each input element once	Linear search
$\Theta(n \cdot \log n)$	Split input into half (which can be done at most log n times) At each split look at each element once	Merge sort (tri fusion)
$\Theta(n^2)$	Look at each pair of elements (i,j)	Insertion sort
$\Theta(n^3)$	Look at each triple of element (i,j,k)	Floyd's shortest path algorithm
$\Theta(2^n)$	Look at all subset of a list or all permutation of a list or all paths in a graph	Create a list of all binary numbers of length n, e.g., input: 3 output:{{0,0,0},{0,0,1},{0,1,0},{0,1,1}, {1,0,0},{1,0,1},{1,1,0},{1,1,1}}

In this table "look at element x" means to do a constant number instructions for element x, e.g., compare x with 3, add x to another variable,…

Exercise: Execution

- What happens if you execute this algorithm with a = 32 and b = 48 and r =15?

gcd
input : a,b,r integers strictly greater than 0 output : *x*
repeat *r ← a mod b //rest of the division a/b* *a ← b* *b ← r* ***as long as*** *r > 0* ***return:*** *a*

Iter.	a	b	r
Init	32	48	15
1	48	32	32
2	32	16	16
3	16	0	0

Complexity: logarithmic in the value of the inputs (i.e., linear in the number of bits)
See https://en.wikipedia.org/wiki/Euclidean_algorithm#Algorithmic_efficiency
(You are not expected to compute the complexity of this algorithm yourself.)

Exercise: Execution

- What happens if you execute this algorithm with L={1,2,3,4,5} and n = 5

```
algorithm
input : list L of size n
output : x
  x ← 0
  for i from 1 to n
     x ← 10 · x + L[i]
   return: x
```

Iteration	x
Init	0
1	1
2	10 + 2 = 12
3	120 + 3 = 123
4	1230 + 4 = 1234
5	12340 + 5 = 12345

Complexity: Θ(n) because we have n iteration of the loop.

Exercise

- For each of the algorithms on the subsequent slides, answer the following four questions:
  1. What happens if you execute it with the input 6?
  2. Is this a recursive algorithm?
  3. What does it do in general?
  4. What is its asymptotic complexity?

Algo1

1. What happens if you execute it with the input 6?
2. Is this a recursive algorithm?
3. What does it do in general?
4. What is its asymptotic complexity?

algo1

input : an integer n
output : an integer

```
1  if n equals 0
2    return: 0
3  k ← 0
4  for j from 1 to 2n-1
5    if (j is odd)
6      k ← k + j
7  return: k
```

Line	n	k	j
Line 0	6	?	?
Line 3	6	0	?
Line 4	6	0	1
Line 6	6	0+1=1	1
Line 4	6	1	1+1=2
Line 4	6	1	2+1=3
Line 6	6	1+3=4	3
…			

1. k = 1 + 3 + 5 + 7 + 9 + 11 =36
2. Non recursive
3. Adding the odd numbers from 1 to 2n-1 (which is equal to n^2)
4. Complexity:
   - Instructions before the loop starts (Line 1-3): 3
   - Max. number of instructions per loop iteration (Line 4-6) = 4;
   - Number of loop iterations = 2n-1;
   - Total = 3 + 4 * (2n-1) = 8n - 1 = Θ(n) linear

Algo2

1. What happens if you execute it with the input 6?
2. Is this a recursive algorithm?
3. What does it do in general?
4. What is its asymptotic complexity?

algo2
input : an integer n output : an integer
return: n^2

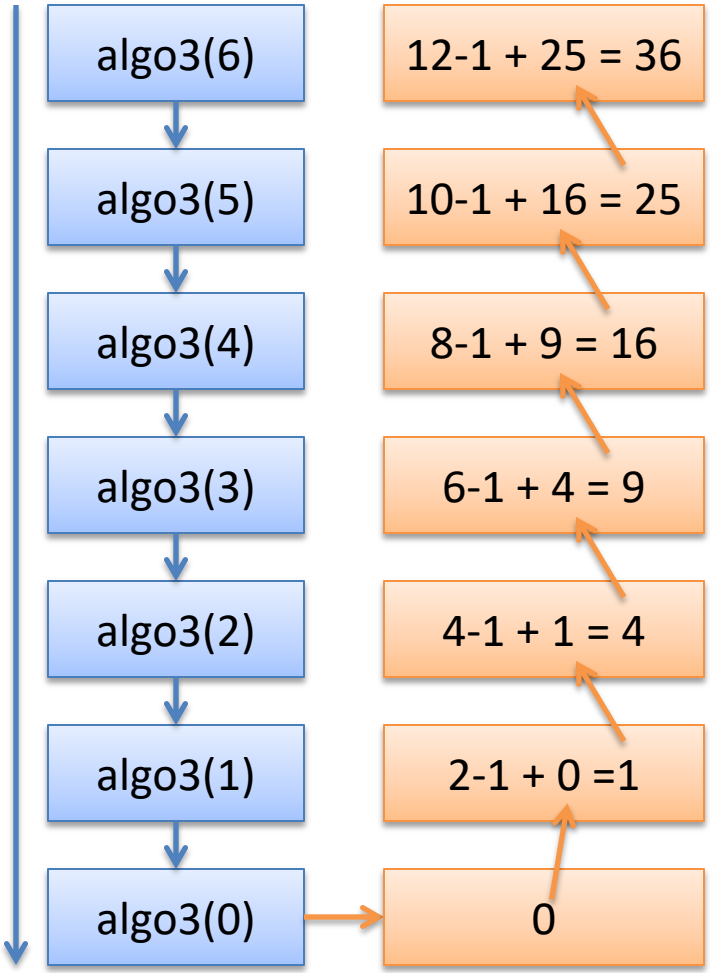
Line	n
Line 0	6
Line 1	36

1. $6^2 = 36$
2. Non recursive
3. Computes the square
4. Complexity: $\Theta(1)$ constant

Algo3

algo3
input : an integer n output : an integer
if *n equals 0*   ***return:*** *0* ***return:*** *2n-1* + **algo3**(n-1)

Height = n
#calls = n +1



Additional steps per level

3 (+,-,assign)

3

3

3

3

3

2

1. 0 + 1 + 3 + 5 + 7 + 9 + 11 =36
2. Recursive
3. Adding the odd numbers from 1 to 2n-1 (which is equal to n^2)
4. Complexity:
   - Max. number of instructions in addition to recursive call = 3
   - Height of the recursive stack n+1
   - Total = 3 * (n+1) = Θ(n) linear

Algo4

algo4

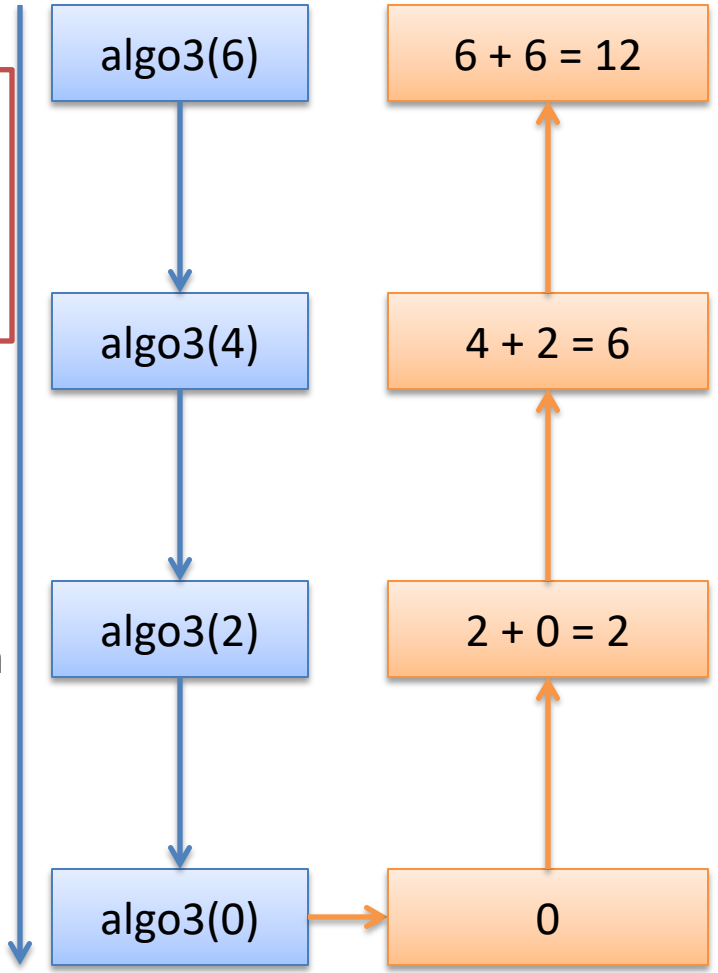
input : an integer n
output : an integer

```
if n equals 0
    return: 0
if n equals 1
    return: 1
return: n + algo4(n-2)
```

$$n - 2k \leq 1$$
$$n - 1 \leq 2k$$
$$k \geq \frac{n-1}{2}$$
$$\#\text{calls} = k + 1$$

Height=k
#calls = k+1

1. 0 + 2 + 4 + 6 = 12
2. Recursive
3. If n is even, it adds the even numbers from 0 to n. If n is odd, it adds the odd numbers from 0 to n.
4. Complexity:
   - Max. number of instructions in addition to recursive call = 3
   - Height of the recursive stack n/2 + 1
   - Total = 3 * ((n-1)/2 + 1) = Θ(n) linear



Additional steps per level

3(2 comp., +)

3

3

2

Exercise: Algo5

Consider the following algorithm

1. What happens if you execute it with input 4?
2. What does it compute?
3. Determine its order of complexity

algo5
input : an integer n ≥ 1 output : an integer
k ← *0* ***for*** *i* ***from*** *1* ***to*** *n* *s* ← *0* ***for*** *j* ***from*** *1* ***to*** *n* *s* ← *s* + *1* *k* ← *k* + *s* ***return:*** *k*

Exercise: Algo5

Consider the following algorithm

1. What happens if you execute it with input 4?
2. What does it compute?
3. Determine its order of complexity

algo5
input : an integer n ≥ 1 output : an integer
1 $k \leftarrow 0$ 2 ***for*** *i* ***from*** *1* ***to*** *n* 3 $s \leftarrow 0$ 4 ***for*** *j* ***from*** *1* ***to*** *n* 5 $s \leftarrow s + 1$ 6 $k \leftarrow k + s$ 7 ***return:*** *k*

Line	k	i	s	j
Line 0	?	?	?	?
Line 1	0	?	?	?
Line 2	0	1	?	?
Line 3	0	1	0	?
Line 4	0	1	0	1
Line 5	0	1	1	1
….				
Line 6	4	1	4	

Exercise: Algo5

Consider the following algorithm

1. What happens if you execute it with input 4?
2. What does it compute?
3. Determine its order of complexity

```
algo5
input : an integer n ≥ 1
output : an integer
    k ← 0
    for i from 1 to n
        s ← 0
        for j from 1 to n
            s ← s + 1
        k ← k + s
    return: k
```

1. (1+1+1+1)+(1+1+1+1)+(1+1+1+1)+(1+1+1+1) = 16
2. It computes n^2
3. Complexity:
   - Max. number of instructions in inner loop (over j): ca. 4
   - Max. number of iterations of the inner loop: n
   - Max. number of steps in the outer loop (over i): 5 + 4*n (inner loop)
   - Max. number of iterations of the outer loop (over i) = n
   - Total = n * (5 + 4n) = $5n + 4n^2 = \Theta(n^2)$ quadratic

Exercise: Algo6

Consider the following algorithm

1. What happens if you execute it with input 4?
2. What does it compute?
3. Determine its order of complexity

algo6
input : an integer n ≥ 1 output : an integer
1 $k \leftarrow 0$ 2 ***for*** *i* ***from*** *1* ***to*** *n* 3 $s \leftarrow 0$ 4 ***for*** *j* ***from*** *i* ***to*** *n* 5 $s \leftarrow s + 1$ 6 $k \leftarrow k + s$ 7 ***return:*** *k*

3. Complexity:
   - Instructions in the inner loop (line 4-5): ca. 3 (constant)
   - Iterations of the inner loop: (n-i)
   - Instructions in the outer loop without the inner loop: 5 (const)
   - Instructions of the algorithm = sum of instructions over all iterations of the outer loop (line 2-6) =

$$\sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n\cdot(n+1)}{2} = \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$$

1. (1+1+1+1)+(1+1+1)+(1+1)+(1) = 10
2. It computes sum of number up to n

Exercise: Algo7

Consider the following algorithm

1. What happens if you execute it with input 4?
2. Determine its order of complexity. Answer: Θ(n log n)

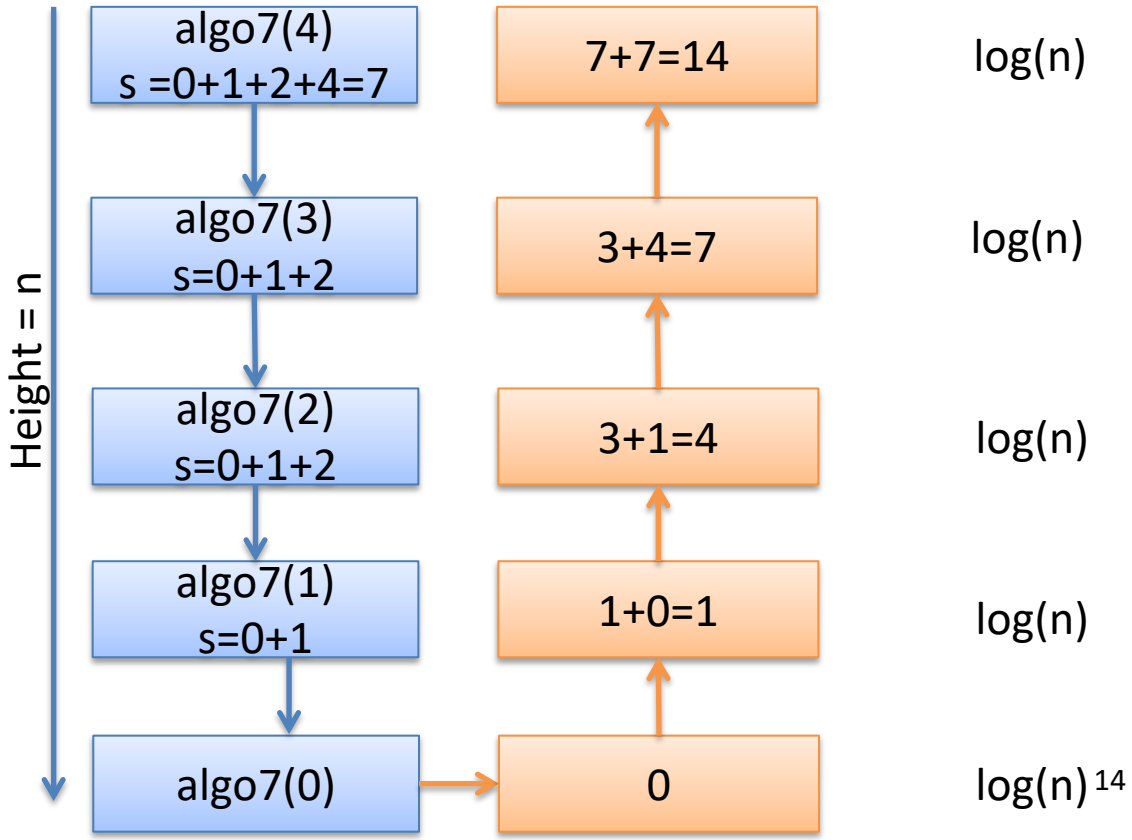
algo7

input : an integer n
output : an integer

```
if n equals 0
  return: 0
s ← 0
k ← 1
while k ≤ n
    s ← s + k
    k ← 2 * k
return: s + algo7(n-1)
```

Iterations of the while loop:

$$2^i > n$$
$$i > \log n$$



Complexity

- Consider a computer that needs 1s to execute any instructions. How much time does it take to execute one of the following algorithm on an input of the size n=1'000'000?
  - Algorithm A with complexity O(1)
  - Algorithm B with complexity O(n)
  - Algorithm C with complexity $O(n^2)$
  - Algorithm D with complexity $O(\log_2(n))$

1 day ≈ $8.5 \cdot 10^4$s
1 week ≈ $6 \cdot 10^5$s
1 year ≈ $3 \cdot 10^7$s

$2^{10} \approx$ 1'000 (kilo)
$2^{20} \approx$ 1'000'000 (mega)
$2^{30} \approx$ 1'000'000'000 (giga)

Complexity

- Consider a computer that needs 1s to execute any instructions. How much time does it take to execute one of the following algorithm on an input of the size n=1'000'000?
  - Algorithm A with complexity O(1) : 1s
  - Algorithm B with complexity O(n) : 10^6s ≈ 1.65 weeks
  - Algorithm C with complexity O(n^2): 10^{12}s ≈ 33'000 years
  - Algorithm D with complexity O($\log_2(n)$): 20s

1 day ≈ $8.5 \cdot 10^4$s
1 week ≈ $6 \cdot 10^5$s
1 year ≈ $3 \cdot 10^7$s

$2^{10} \approx$ 1'000 (kilo)
$2^{20} \approx$ 1'000'000 (mega)
$2^{30} \approx$ 1'000'000'000 (giga)

Write an Algorithm: Product

- Given a list L of numbers of size n, and an integer p, check if the product of any two numbers in L is equal to p.

Product

checkProduct
input : a list L, its size n, and an integer p output : true/false

for *i from 1 to n*
  for *j from 1 to n*
  if *(L[i]*L[j] = p)*
  return *true*
return *false*

Complexity: $\Theta(n^2)$ because we have n iterations of the loop over i, and for every iteration of i, we do n iterations of the loop over j.

Write an Algorithm: Sum of matrix elements

- Given a matrix M of size n x n, compute the sum of the elements below the diagonal.

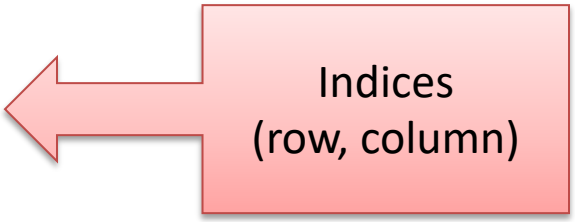
3	4	0	5
1	1	3	9
2	3	7	0
4	1	1	3

Sum = 12

Write an Algorithm: Sum of matrix elements

- Given a matrix M of size n x n, compute the sum of the elements below the diagonal.

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4



From row 2 to n.
From col 1 to row-1

Sum of Matrix elements (n x n)

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4

From row 2 to n.
From col 1 to row-1

matrixSum

input : a table M of n x n
output : sum of elements below the diagonal

```
s ← 0
for row from 2 to n
    for col from 1 to row-1
        s ← s + M[row][col]
return s
```

Complexity: $\Theta(n^2)$ because we iterate over n-2 rows, and for every row r, we go over r-1 columns. $\sum_{r=2}^{n} r - 1 = \sum_{s=1}^{n-1} s = \frac{(n-1)\cdot n}{2} = \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$

Sum of Matrix elements (n x m)

- Does this algorithm also work for a Matrix of size n x m, where n ≠ m?

matrixSum
input : a table M of n x n output : sum of elements below the diagonal

s ← *0*
for *row from 2 to n*
  for *col from 1 to row-1*
  s ← *s + M[row][col]*
return *s*

Sum of Matrix elements (n x m)

- Does this algorithm also work for a Matrix of size n x m, where n ≠ m?

Element 6,5 is missing!

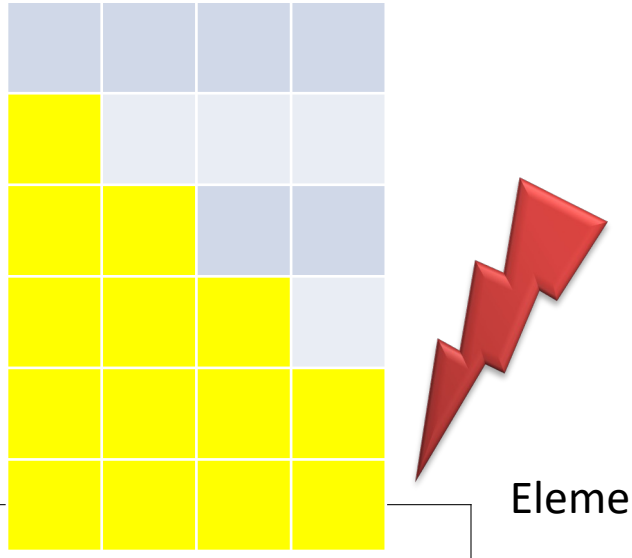
matrixSum

input : a table M of n x n
output : sum of elements below the diagonal

```
s ← 0
for row from 2 to n
    for col from 1 to row-1
        s ← s + M[row][col]
return s
```

Sum of Matrix elements (n x m)

- Does this algorithm also work for a Matrix of size n x m, where n ≠ m?



Element 6,5 is missing!

matrixSum

input : a table M of n x n
output : sum of elements below the diagonal

```
s ← 0
for row from 2 to n
    for col from 1 to row-1  min(row-1,m)
        s ← s + M[row][col]
return s
```

Write an Algorithm: Trains

- Given a list of cities named 1,2,3,..,n and a table T of size n x n that stores the direct train connections between these cities, i.e., if T(i,j) is 1, then there is a direct train going from city i to city j, if T(i,j) is 0, then there is no direct train from i to j.
- Given an algorithm that returns yes, if there is a city that does not have any train connection? (Note that we need to check that there is no train from and to this city.)

Trains

allCityHaveATrainConnection
input : a Table T of size n x n output : true or false

```
hasConn ← list with n entries.
for i from 1 to n
  hasConn[i] ← 0
for i from 1 to n
  for j from 1 to n
    if (T[i][j] = 1)
        hasConn[i] ← 1;
        hasConn[j] ← 1;
for i from 1 to n
  if (hasConn[i] == 0)
    return true
return false
```

Initialize all elements in hasConn to 0
The number of instruction in this loop is proportional to n.

Go through all the connections (i,j) in the table and mark that city i and city j have a connection
The number of instructions in this loop is proportional to n^2.

Check is there is one city without a connection. If yes, then return true.
The number of instruction in this loop is proportional to n.

Complexity: $n + n^2 + n = \Theta(n^2)$

Trains

Other algorithms you can write:

- Find a city with no outgoing train connections
- Find a city with no incoming train connections
- Find a city with the same number incoming as outgoing connections

Questions ?