Data Structures and Algorithms

Lab

Carolin Hannusch

DEIK

2026

Definition of algorithm

- An algorithm is a computational procedure, which creates a value or set of values (output) from another value or set of values (input)

Definition of algorithm

- An algorithm is a computational procedure, which creates a value or set of values (output) from another value or set of values (input)
- An algorithm is a procedure to solve a problem or to work out computations.

Definition of algorithm

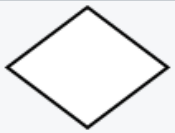
- An algorithm is a computational procedure, which creates a value or set of values (output) from another value or set of values (input)
- An algorithm is a procedure to solve a problem or to work out computations.
- The relation between input and output depends on the problem.

Flowchart

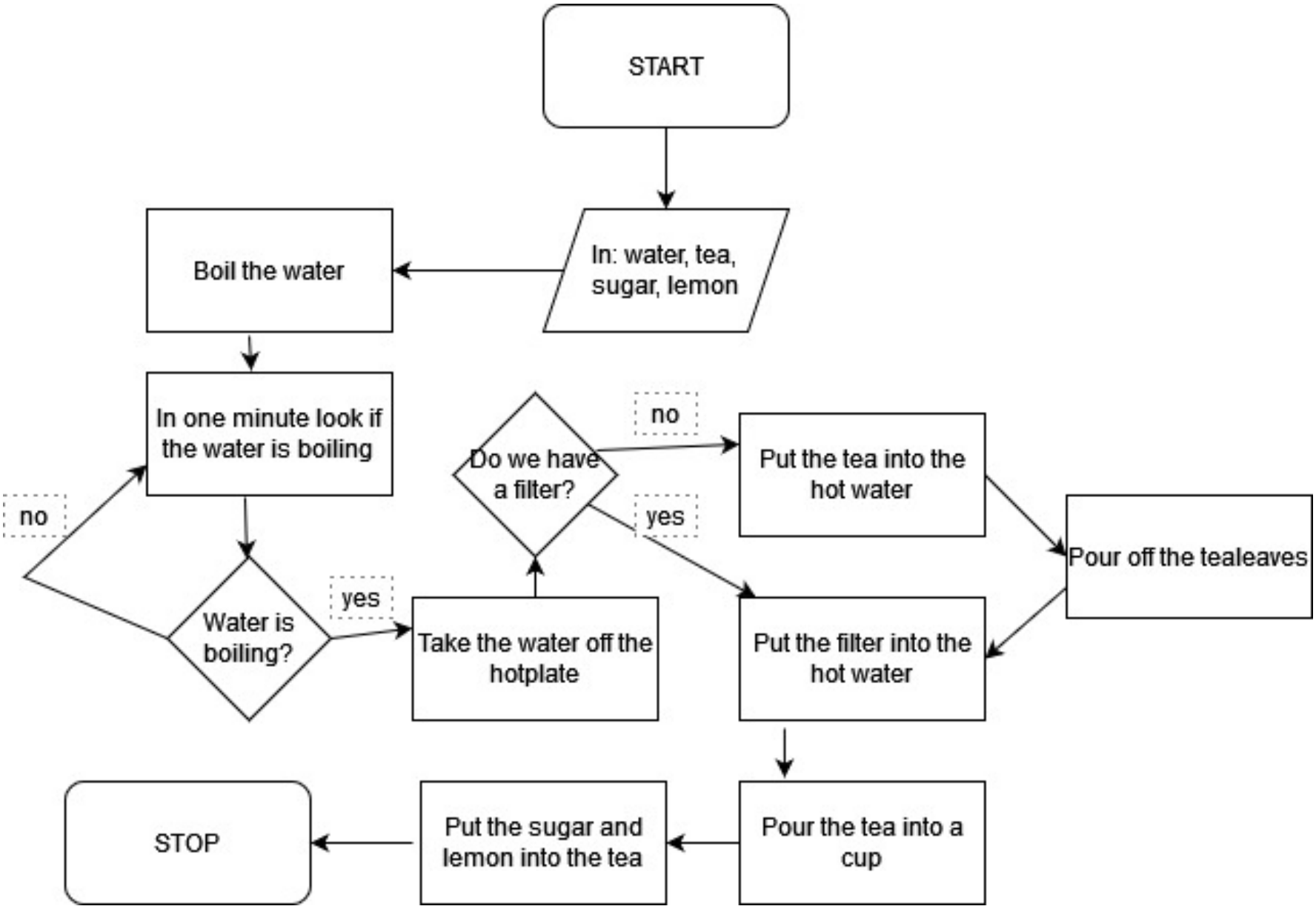
Building blocks [edit]

Common symbols [edit]

The American National Standards Institute (ANSI) set standards for flowcharts and their symbols in the 1960s.[14] The International Organization for Standardization (ISO) adopted
Generally, flowcharts flow from top to bottom and left to right.[17]

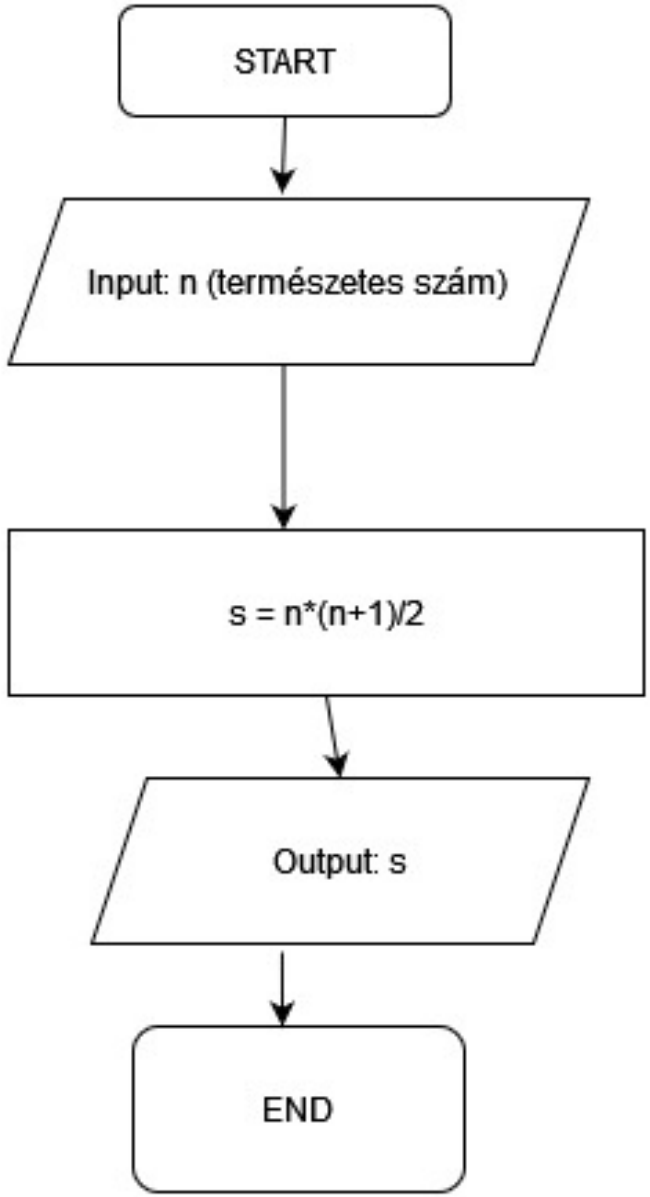
ANSI/ISO Shape	Name	Description
	Flowline (Arrowhead)[15]	Shows the process's order of operation. A line coming from one symbol and pointing at another.[14] Arrowheads are added if the flo
	Terminal[14]	Indicates the beginning and ending of a program or sub-process. Represented as a stadium,[14] oval or rounded (fillet) rectangle. T a process, such as "submit inquiry" or "receive product".
	Process[15]	Represents a set of operations that changes value, form, or location of data. Represented as a rectangle.[15]
	Decision[15]	Shows a conditional operation that determines which one of the two paths the program will take.[14] The operation is commonly a y
	Input/Output[15]	Indicates the process of inputting and outputting data,[15] as in entering data or displaying results. Represented as a parallelogram.
	Annotation[14] (Comment)[15]	Indicating additional information about a step in the program. Represented as an open rectangle with a dashed or solid line connec
	Predefined Process[14]	Shows named process which is defined elsewhere. Represented as a rectangle with double-struck vertical edges.[14]
	On-page Connector[14]	Pairs of labeled connectors replace long or confusing lines on a flowchart page. Represented by a small circle with a letter inside.[1
	Off-page Connector[14]	A labeled connector for use when the target is on another page. Represented as a home plate-shaped pentagon.[14][18]

Make a tea

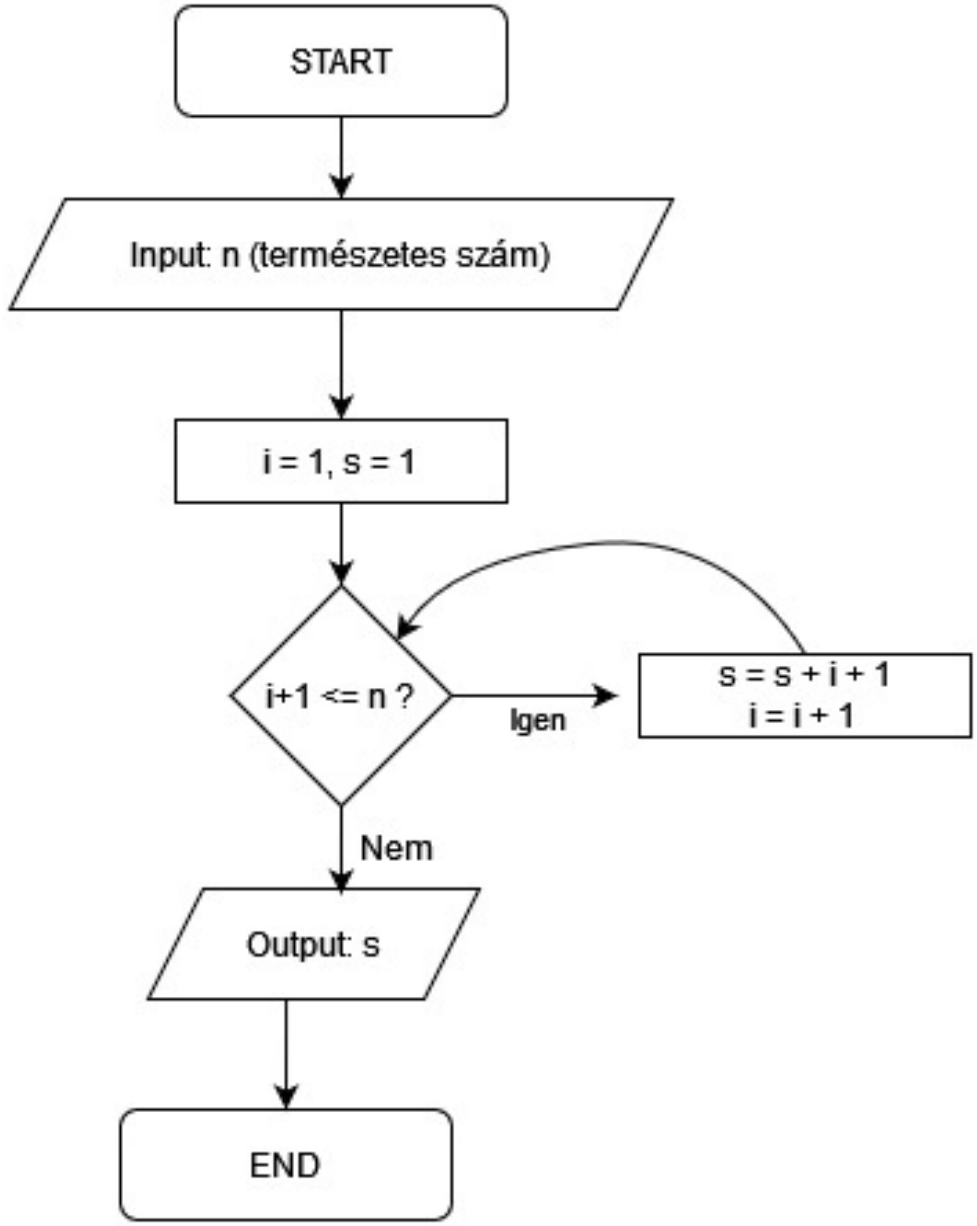

START
In: water, tea, sugar, lemon
Boil the water
In one minute look if the water is boiling
Water is boiling?
no
yes
Take the water off the hotplate
Do we have a filter?
no
yes
Put the tea into the hot water
Put the filter into the hot water
Pour off the tealeaves
Pour the tea into a cup
Put the sugar and lemon into the tea
STOP

Add the numbers from 1 to n

Add the numbers from 1 to n



...with n steps


START
Input: n (természetes szám)
i = 1, s = 1
i+1 <= n ?
Igen
s = s + i + 1
i = i + 1
Nem
Output: s
END

Python 3.14

```
>>> osszeadas(10)
55.0
>>> def osszeadrek(n):
...     x=0
...     i=1
...     while i<n+1:
...         x=x+i
...         i=i+1
...
>>> def osszeadrek(n):
...     x=0
...     i=1
...     while i<n+1:
...         x=x+i
...         i=i+1
...         return x
...
>>> osszeadrek(10)
1
>>> def osszeadrek(n):
...     x=0
...     i=1
...     while i<n+1:
...         x=x+i
...         i=i+1
...     return x
...
>>> osszeadrek(10)
55
>>>
```

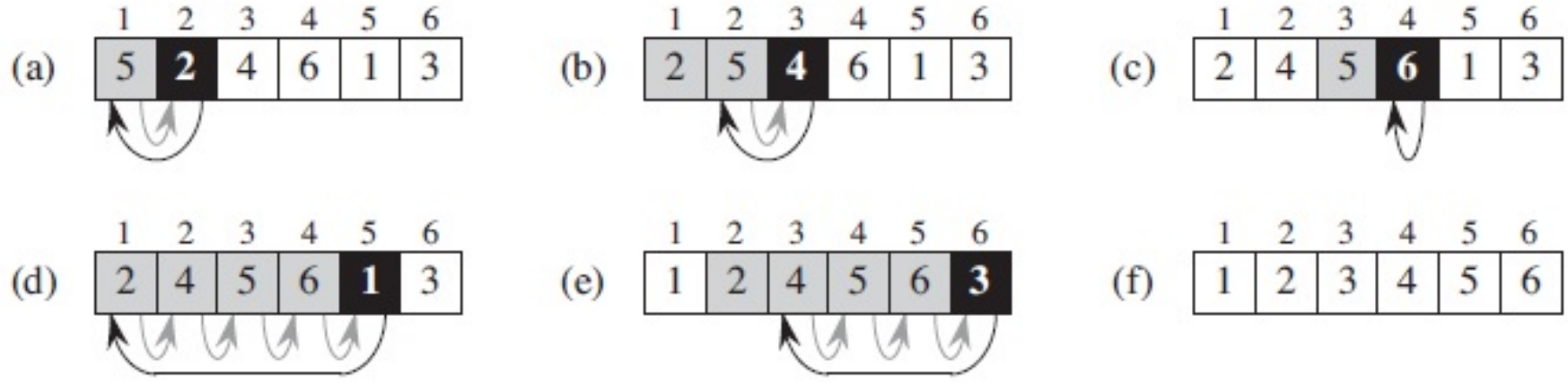
Python 3.14

```
0.0001583099365234375
>>> import time
...
... # Save timestamp
... start = time.time()
...
... print(osszead(1000))
...
... # Save timestamp
... end = time.time()
...
... print(end - start)
...
500500.0
9.3221664428710 94e-05
>>> import time
...
... # Save timestamp
... start = time.time()
...
... print(osszeadrek(1000))
...
... # Save timestamp
... end = time.time()
...
... print(end - start)
...
500500
0.0001654624938964843 8
>>>
```

Algoritmus rendezésre

Input: A=<5,2,4,6,1,3>

Output: A'=<1,2,3,4,5,6>



Pseudocode

```
for j = 2 to A.length do
    key = A[j]                          ▷ Put A[j] into A[1..j-1].
    i = j − 1
    while i > 0 do
        if A[i] > key then
            A[i + 1] = A[i]
            i = i − 1
            A[i + 1] = key
        end if
    end while
end for
```

Linear search

Input: vector

Linear search

Input: vector
Output: the coordinate of a given value in the vector

Linear search

Input: vector
Output: the coordinate of a given value in the vector

The algorithm starts at the first coordinate of the vector and proceeds along. At each coordinate it investigates if the coordinate coincides with the given value.

Linear search

Input: vector
Output: the coordinate of a given value in the vector

The algorithm starts at the first coordinate of the vector and proceeds along. At each coordinate it investigates if the coordinate coincides with the given value. If the length of the vector is increased by 1, then the number of steps in the algorithm also increases by 1. The running time is at most n.

Binary search

Input: ordered vector

Binary search

Input: ordered vector
Output: coordinate of a given value in the vector

Binary search

Input: ordered vector
Output: coordinate of a given value in the vector

The algorithm divides the vector into two parts in the middle. The value we search is either greater or smaller than the value in the middle. Therefore, we can cut one half of the vector.

Binary search

Input: ordered vector
Output: coordinate of a given value in the vector

The algorithm divides the vector into two parts in the middle. The value we search is either greater or smaller than the value in the middle. Therefore, we can cut one half of the vector. The running time of the algorithm is at most $log_2 n + 1$.

```python
def linear_search(A, x, n):
    for i in range(n):
        if A[i] == x:
            return i
    return unsuccessful


print(linear_search([1, 2, 3, 4, 5], 3, 5))
```

```
> def binary_search(A, n, x):
.       L = 0
.       R = n-1
.       while L <= R:
.           m = math.floor((L+R)/2)
.           if A[m] < x:
.               L = m +1
.           elif A[m] > x:
.               R = m - 1
.           else:
.               return m
.       return unsuccesful
...
>>>
>>> binary_search([1,2,3,4,5],5,2)
1
>>> binary_search([1,2,3,4,5],5,1)
0
>>> binary_search([1,3,4,5,7,9,33],7,9)
5
```

Exercises I

1. Given a list of integers and an integer k, determine the index of k using linear search. If k is not in the list, return -1.
2. Given a list and a value k, print all indices where k occurs in the list.
3. Given a list of integers, find the first even number using linear search.
4. Given a non-empty list, find the maximum element and its index using linear search.

Exercises II

1. Given a sorted list and a value k, find the index of k using binary search. If k is not in the list, return -1.
2. Given a sorted list and a value k that is not in the list, determine the index where k can be inserted so that the list remains sorted.
3. Given a sorted list where an element may occur multiple times, find the index of the first occurrence of k using binary search.
4. Explain briefly why binary search cannot be applied to an unsorted list.