

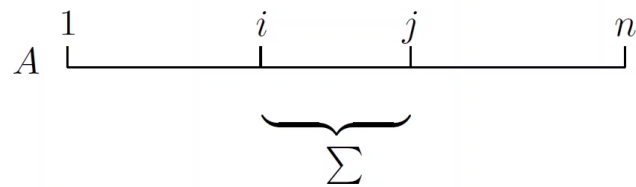
## Example 1: Maximum Subarray Sum

Given: Array  $A[1, \dots, n]$  of integers (also negative values)

Goal: continuous subarray  $A[i, \dots, j]$  with maximum sum

4	-5	4	2	-3
---	----	---	---	----

 $\Rightarrow \sum = 6$



- **Method 1:**  
Check all subsets of  $A$  and check for connectedness.

$n \dots$  sum of a subset

$2^n \dots$  number of subsets

$\mathcal{O}(n2^n)$  runtime

A computer with  $10^6$  Operations per second needs for  
 $n = 1000$  numbers  $\approx 10^{304}$  operations, i.e.,  $\approx 10^{290}$  years.

- Same computer: for  $n = 10^6$  numbers  $\approx 10^{300000}$  years.

## Method 2:

Only checking connected sequences:

$$\max_{1 \leq i \leq j \leq n} \left\{ \sum_{k=i}^j A[k], 0 \right\}$$

$i, j, k$  run for at most  $n$  steps.

Three nested loops  $\Rightarrow \mathcal{O}(n^3)$

For  $n = 1000 \Rightarrow 10^9$  steps  $\sim 16$  min.

For  $n = 10^6 \Rightarrow \approx 32000$  years.

- 

Method 3 (pseudo code):

Computing the sum 'online' with  $j$ .

```
max := 0; from := 0; to := 0;
for i := 1 to n do
  sum := 0;
  for j := i to n do
    sum := sum + A[j];
    if sum > max then max := sum; from = i; to = j;
  fi
od
od
output(" A[" , from, " - " , to, "] maximum sum = " , max)
```

- 

- almost equivalent to method 2
- does not recalculate existing sum
- instead adds next element to previous

$i, j$  go through at most  $n$  values  $\Rightarrow \mathcal{O}(n^2)$  steps

For  $n = 1000 \Rightarrow 10^6$  steps  $\sim 1$  sec

For  $n = 10^6 \Rightarrow \approx 11,5$  days.

-

#### Method 4:

Run through the input once with a 'scanline' and only consider the part of the input currently covered by the scanline.

Idea: Calculate for every index  $k$  the maximum sequence  $T_k$  ending at  $k$ .

From this get a  $k$  with a global maximum sequence.

Observe:

$$T_k \geq 0,$$
$$T_k = \max\{T_{k-1} + A[k], 0\}$$

•

#### Method 4 (pseudo code):

```
max := 0; from := 0; to := 0; f := 1; T := 0
for k := 1 to n do
  T := T + A[k]
  if T < 0 then T := 0; f = k + 1
  fi
  if T > max then max := T; from := f; to := k
  fi
od
output("A[", from, " - ", to, "] maximum sum = ", max)
```

$n = 1000 \Rightarrow \approx 1/1000$  second; for  $n = 10^6 \Rightarrow \approx 1$  second.

• Compare:  $10^{300000}$  years (Method 1) or 32000 years (M. 2).