

# 1 Maximum Subarray Problem

---

Given an array of numbers (either positive or negative), we wish to calculate the subset of consecutive numbers whose sum is largest.

$$A : \boxed{3} \boxed{2} \boxed{-4} \boxed{-5} \underbrace{\boxed{6} \boxed{1} \boxed{-3} \boxed{7}}_{\text{sums to 11}} \boxed{-8} \boxed{2}$$

In this example, we see that the maximum sum is 11 with the indices of the subarray being  $A[5] : A[8]$ .

## 1.1 Cubic Time

### Pseudocode

---

**Algorithm 1** Maximum Contiguous Sum (*Cubic*)

---

```
1:  $M \leftarrow 0$ 
2: for  $i = 1$  to  $n$  do
3:   for  $j = i$  to  $n$  do
4:      $S \leftarrow 0$ 
5:     for  $k = i$  to  $j$  do
6:        $S \leftarrow S + A[k]$ 
7:     end for
8:      $M \leftarrow \text{MAX}(M, S)$ 
9:   end for
10: end for
```

---

## Analysis

From our algorithm above we can get the following analysis for it's complexity.

$$\begin{aligned}\sum_{i=1}^n \sum_{j=i}^n \sum_{k=i}^j 1 &= \sum_{i=1}^n \sum_{j=i}^n j - i + 1 \\ &= \sum_{i=1}^n \sum_{j=1}^{n-i+1} j \\ &= \sum_{i=1}^n \frac{(n-i+1)(n-i+2)}{2} \\ &= \frac{1}{2} \sum_{i=1}^n (n-i+1)(n-i+2) \\ &= \frac{1}{2} \sum_{n-i+1=1}^n i(i+1) \\ &= \frac{1}{2} \frac{n(n+1)(n+2)}{3}\end{aligned}$$

From this, we can see this first version of our algorithm is  $\Theta(n^3)$  or cubic time.

## 1.2 Quadratic Time

### Pseudocode

---

**Algorithm 1** Maximum Contiguous Sum (*Quadratic*)

---

```
1:  $M \leftarrow 0$ 
2: for  $i = 1$  to  $n$  do
3:    $S \leftarrow 0$ 
4:   for  $j = i$  to  $n$  do
5:      $S \leftarrow S + A[j]$ 
6:      $M \leftarrow \text{MAX}(M, S)$ 
7:   end for
8: end for
```

---

## Analysis

We then use sums to analyze this algorithm.

$$\begin{aligned}\sum_{i=1}^n \sum_{j=i}^n 1 &= \sum_{i=1}^n n - i + 1 \\ &= \sum_{i=1}^n i \\ &= \frac{n(n+1)}{2}\end{aligned}$$

So this more efficient version is  $\Theta(n^2)$  or quadratic time.

## 1.3 Linear Time

Using dynamic programming we can improve on this algorithm even further. Conceptually, the maximum sum is just the previous maximum sum plus the current value.

### Pseudocode

---

**Algorithm 1** Maximum Contiguous Sum (*Linear*)

---

```
1:  $M \leftarrow 0$ 
2:  $S \leftarrow 0$ 
3: for  $i = 1$  to  $n$  do
4:    $S \leftarrow \text{MAX}(S + A[i], 0)$ 
5:    $M \leftarrow \text{MAX}(M, S)$ 
6: end for
```

---

## Analysis

It follows that this algorithm is  $\Theta(n)$  or linear time. Correctness of this algorithm stems from proof by induction on  $S \leftarrow \text{MAX}(S + A[i], 0)$ , as this is the loop invariant.