

# Maximum Subarray Sum

لينك ال جيت هب فيه كل الملفات اللي تخص المشروع بطريقه منظمه جدا  
<https://github.com/SamahCodes/Algorithm>

## Team Information:-

### Team Members:

Samah Mohamed Salah — ID: 1000287603

Sahar Osama El-sieed — ID: 1000287997

Sahar Reda Helmy — ID: 1000288192

Abdelrahman Mohamed — ID: 1000287467

---

## Project Overview:-

[git hub على Readme file](#) لينك ال جيت هب عليه Readme file

This project presents a complete study of the Maximum Subarray Sum Problem, focusing on comparing a Naive (Brute Force) solution with an Optimized solution using Kadane's Algorithm.

The project includes:

Problem identification

Algorithm implementations in JavaScript

Theoretical analysis

Empirical analysis

Performance comparison

Documentation and presentation materials

## Problem Identification:-

The Maximum Subarray Sum problem aims to find the maximum possible sum of a contiguous subarray within a given array of integers (which may contain both positive and negative values).

يعنى array فيها ارقام موجبه وسالبها بنطلع ال اللى مجموعها اكبر مجموع شرط ان يكون عناصرها متصلين

Input:

An array of integers (positive and/or negative)

Output:

The maximum sum of any contiguous subarray

## Algorithms Implemented:-

1-Naive Solution (Brute Force): ليناك الفايل اللي بيشرح كل حاجه تخص النايف

Checks all possible contiguous subarrays

Computes their sums

Selects the maximum sum

Time Complexity:  $O(n^2)$

Space Complexity: O (1)

2- Optimized Solution (Kadane's Algorithm) : [لينك فايل ال optamized](#)

Uses dynamic programming

Processes the array in a single pass

Maintains current and maximum sums efficiently

Time Complexity: O(n)

Space Complexity: O(1)

## Implementation Details:-

نفذه بال java script

عشان خلفيتنا اڪتر لک فرونت اند

ف طبقنا ال algorithm

. والرابط موجود بردو ممكن تجرب فيه امثاله جاهزه کويس ui و ربطناها ب

<https://samahcodes.github.io/Algorithm/>

Programming Language: JavaScript

Execution Environment: GUI as a website

Files:

[algorithm1.js](#) → Naive implementation

```

1  function maxSubarrayNaive(arr) {
2      let maxSum = -Infinity;
3
4      for (let i = 0; i < arr.length; i++) {
5          let currentSum = 0;
6          for (let j = i; j < arr.length; j++) {
7              currentSum += arr[j];
8              maxSum = Math.max(maxSum, currentSum);
9          }
10     }
11
12     return maxSum;
13 }
14

```

[algorithm2.js](#) → Optimized (Kadane) implementation

```

1  function maxSubarrayKadane(arr) {
2      let currentSum = arr[0];
3      let bestSum = arr[0];
4
5      for (let i = 1; i < arr.length; i++) {
6          currentSum = Math.max(arr[i], currentSum + arr[i]);
7          bestSum = Math.max(bestSum, currentSum);
8      }
9
10     return bestSum;
11 }
12

```

## Theoretical Analysis:- التحليل

Theoretical analysis focuses on time and space complexity:

Algorithm	Time Complexity	Space Complexity
Naive	$O(n^2)$	$O(1)$
Optimized (Kadane)	$O(n)$	$O(1)$

The optimized algorithm is theoretically superior and more scalable for large input sizes.

## **Empirical Analysis:-** الإثنين

Empirical analysis was conducted by running both algorithms on datasets of increasing sizes and measuring execution time.

Independent Variable: Input size ( $n$ )

Dependent Variable: Execution time (milliseconds)

Measurement Method: JavaScript timing functions

Environment: Same machine and runtime for all tests

Observations:

Kadane's algorithm shows linear growth in execution time.

The naive approach becomes impractical for large input sizes.

Empirical results confirm theoretical expectations.

## **Comparison Summary:-**

Aspect	Naive	Optimized
<b>Performance</b>	Slow	Fast
<b>Scalability</b>	Poor	Excellent
<b>Practical Use</b>	Limited	Recommended

 Live Version :

<https://samahcodes.github.io/Algorithm/>

 Video Link:

[https://drive.google.com/drive/folders/1cWC8\\_G2K1O8l5F3kcUSSveSenw5kVH7L?usp=sharing](https://drive.google.com/drive/folders/1cWC8_G2K1O8l5F3kcUSSveSenw5kVH7L?usp=sharing)

## Conclusion: -

This project demonstrates how algorithmic optimization significantly improves performance.

Kadane's Algorithm proves to be both theoretically and empirically superior to the naive approach, making it the preferred solution for real-world applications.