Bu-Ali Sina University

Report

Maximum Subarray Sum

Mehran Shahidi

supervised by
Dr. Mansori

May 7, 2020

Summary

In this report, I will explain how to solve the Maximum Sub-Array Sum with three different approaches, and in the end, you will see the comparison of those approaches. This report is written in LaTeX. For accessing to the LaTeX file and the source codes you can visit my github repo for the algorithm course.

https://github.com/m3hransh/algorithmt

I also have an algorithm course on youtube feel free to check that out too.

https://www.youtube.com/watch?v=n5rv7pbJpmM&t=256s

Contents

1 introduction

1.1 Defining problem

Let's define our problem(Maximum Sub-Array Sum). You are given a one-dimensional array that may contain both positive and negative integers , find the sum of a contiguous subarray of numbers which has the largest sum. For example, if the given array is {-2, -5, **6, -2, -3, 1, 5**, -6}, then the maximum subarray sum is 7 (see highlighted elements).

2 Brute Force

Our first solution is to check all the possible subarrays. Every subarray is consists of a start and an end. So there are $\binom{n}{2} + n$ possible sub-arrays that is needed to consider (n is for subarray of length 1). This can be implemented with two for-loops. The first for-loop chooses different possible values for the beginning of the sub-array and the second one values for the end of that sub-array. The implementation is in **Listing 1**.

```
def max_sum_sub_brute_force(A):
    '''Return a tuple(range i,j,sum).

    argument:
    A -- list of numbers.
    '''
    max_sum =float('-inf')
    index = (-1,-1)
    for i in range(len(A)):
        temp =0
        for j in range(i, len(A)):
            temp += A[j]
            if temp > max_sum:
                max_sum = temp
                index =(i, j)

    return (index, max_sum)
```

Listing 1: **Brute force implemented of maximum subarray sum**

As you can see, for-loops go through all the possible subarrays, and for each, only it takes $\Theta(1)$ to check if it is the maximum or not. So in total takes $\Theta(n^2)$ to find the maximum subarray.

3 Divide & Conquer