

Jean-Baptiste Kwizera

September 17, 2019

1 Part 1

Algorithm 1. Cubic bruteforce algorithm finds the maximum sum of a contiguous subarray by checking all subarrays and comparing their sums.

```
def cubic(nums):
    N = len(nums)
    maxs = float('-inf')
    for i in range(N+1):
        for j in range(i):
            maxs = max(maxs, sum(nums[j:i]))
    return maxs
```

Proposition A. The running time of **alg. 1** is $\mathcal{O}(N^3)$.

Proof: The inner loop takes $\sum_{j=0}^i j$ operations each time to compute the sum of subarray j to i . From $i = 0$ to $i = N$, the inner loop takes $\sum_{i=0}^N \sum_{j=0}^i j$ operations in total.

Therefore, the exact total number of operations done is

$$\sum_{i=0}^N \sum_{j=0}^i j = \sum_{i=0}^N \frac{i}{2} (i+1) = \frac{1}{2} \left(\sum_{i=0}^N i^2 + \sum_{i=0}^N i \right) = \frac{N}{6} (N+1)(N+2) \sim \frac{N^3}{6}$$

and $\frac{N^3}{6}$ is $\mathcal{O}(N^3)$.

Algorithm 2. Quadratic bruteforce algorithm finds the max. sum by checking all subarrays. However, unlike **alg 1.**, **alg 2.** compares a running sum from $j = i$ to $j = N - 1$ against the maximum sum so far given by some subarray located anywhere between 0^{th} and i^{th} indexes.

```

def quadratic(nums):
    N = len(nums)
    maxs = float('-inf')
    for i in range(N):
        curr = 0
        for j in range(i, N):
            curr += nums[j]
            maxs = max(maxs, curr)
    return maxs

```

Proposition B. The running time complexity of **alg. 2** is $\mathcal{O}(N^2)$.

Proof: Inner loop runs 2 operations for $N, N-1, \dots, 2, 1, 0$ times. Therefore, total number of operations run is

$$2 \sum_{i=0}^N i = 2 \left(\frac{N}{2} (N+1) \right) = N(N+1) \sim N^2 \text{ and } N^2 \text{ is } \mathcal{O}(N^2).$$

Algorithm 3. Linear optimal algorithm.

```

def linear(nums):
    N = len(nums)
    maxs = float('-inf')
    curr = float('-inf')
    for i in range(N):
        curr = max(nums[i], curr + nums[i])
        maxs = max(maxs, curr)
    return maxs

```

alg 3. extends the idea of **alg 2.** of keeping a running sum. It keeps a running sum for the maximum sum ending *at* current index— i^{th} index. The maximum at the i^{th} may contain $nums[i]$ or not, in which case it's just $nums[i]$. Then, it updates the maximum sum if $curr > maxs$.

Proof of correctness by induction: At $i = 0$, $maxs = nums[0]$, since $-\infty < nums[0]$ and $curr = nums[0]$, and is the maximum so far. For $i < N - 1$, $curr$ is the sum of a subarray including $nums[i]$ or just $nums[i]$ if $nums[i] > curr + nums[i]$. $curr = nums[i]$ when $curr < 0$ and $|curr| > nums[i]$. Now with $curr$ updated, it's checked against $maxs$ to update $maxs$ if $curr > maxs$. Thus each time the loop ends, $maxs$ is the maximum possible of some contiguous subarray between indexes 0 and i . Therefore, by induction, when $i = N - 1$ and the loop terminates, $maxs$ is the maximum sum given by a contiguous subarray between indexes 0 and $N - 1$.

Proposition C. The running time of **alg. 3** is $\mathcal{O}(N)$.

Proof: **alg. 3** consists of one *for* loop that does a constant time operation (finding a maximum of two numbers) twice N times. So, **alg. 3** takes exactly $2N$ operations and is therefore linear.

2 Part 2

Problem 5. Show that $\frac{x^2+1}{x+1}$ is $\mathcal{O}(x)$.

Proof. $\frac{x^2+1}{x+1} \leq \frac{x^2+2x+1}{x+1} = \frac{(x+1)^2}{x+1} = x+1 \leq x+x = 2x \Rightarrow \frac{x^2+1}{x+1}$ is $\mathcal{O}(x)$. \square

Problem 6. Show that $\frac{x^3+2x}{2x+1}$ is $\mathcal{O}(x^2)$.

Proof. $\frac{x^3+2x}{2x+1} \leq \frac{x^3+2x}{2x} \leq \frac{x^2+2}{2} = \frac{x^2}{2} + 1 \sim \frac{x^2}{2} \Rightarrow \frac{x^3+2x}{2x+1}$ is $\mathcal{O}(x^2)$. \square

Problem 8.

a) $f(x) = 2x^2 + x^3 \log(x) \Rightarrow n = 3$.

b) $f(x) = 3x^5 + (\log x)^4 \Rightarrow n = 5$.

c) $f(x) = \frac{x^4+x^2+1}{x^4+1} \Rightarrow n = 1$.

d) $f(x) = \frac{x^3+5 \log x}{x^4+1} \Rightarrow n = 0$.

Problem 14.

- a) No
- b) Yes
- c) Yes
- d) No
- e) No
- f) Yes

Problem 19.

a) $(n^2 + 8)(n + 1) = n^3 + n^2 + 8n + 8 \sim n^3 \longrightarrow \mathcal{O}(n^3)$.

b) $(n \log n + n^2)(n^3 + 2) = n^5 + n^4 \log n + 2n^2 + 2n \log n \sim n^5 \longrightarrow \mathcal{O}(n^5)$.

c) $(n! + 2^n)(n^3 + \log(n^2 + 1)) = n!n^3 + 2^n n^3 + n! \log(n^2 + 1) + 2^n \log(n^2 + 1) \sim n!n^3 \longrightarrow \mathcal{O}(n!n^3)$.

Problem 20.

a) $(n^3 + n^2 \log n)(\log n + 1) + (17 \log n + 19)(n^2 + 2) \longrightarrow \mathcal{O}(n^3 \log n)$.

b) $(2^n + n^2)(n^3 + 3^n) \longrightarrow \mathcal{O}(2^n 3^n)$.

c) $(n^n + n2^n + 5^n)(n! + 5^n) \longrightarrow \mathcal{O}(n!n^n)$.

Problem 62. Show that $n \log n$ is $\mathcal{O}(\log n!)$.

Proof. We show that $\log n!$ is $\mathcal{O}(n \log n)$ and it follows directly that $n \log n$ is $\mathcal{O}(\log n!)$. For $f(n) = \log n! = \log 1 + \log 2 + \cdots + \log n$. For $0 < i \leq n$, $\log i$ is bounded above by $\log n$. Therefore, $f(n)$ is bounded above by $f(n) = \log n + \log n + \cdots + \log n = n \log n$. Thus $\log n!$ is $\mathcal{O}(n \log n)$. So it follows that $n \log n$ is $\mathcal{O}(\log n!)$. \square