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

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

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

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 $N, N-1, \cdots, 2, 1, 0$ 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. $\frac{x^2+1}{x+1} \leq \frac{x^2+2x+1}{x+1} = \frac{(x+1)^2}{x+1} = x+1 \leq x+x = 2x$, whenever $x > 1$. Therefore, with $C = 2$ and $k = 1$ as witnesses, $f(x)$ is $\mathcal{O}(x)$. □

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

Proof. $\frac{x^3+2x}{2x+1} \leq \frac{x^3+2x}{2x} \leq \frac{x^2+2}{2} = \frac{x^2}{2} + 1 \leq 3x$. With $C = 3$ and $k = 1$ as witnesses, $f(x)$ is $\mathcal{O}(x^2)$. □

Problem 8.
a) $f(x) = 2x^2 + x^3\log(x) = \mathcal{O}(x^2) + \mathcal{O}(x^4) = \mathcal{O}(x^4) \Rightarrow n = 3$.
b) $f(x) = 3x^5 + (\log x)^4 = \mathcal{O}(x^5) + \mathcal{O}(x^4) = \mathcal{O}(x^5) \Rightarrow n = 5$.
c) $f(x) = \frac{x^4+x^2+1}{x^4+1} = \mathcal{O}(\frac{x^4}{x^4}) = \mathcal{O}(1) \Rightarrow n = 0$.
d) $f(x) = \frac{x^3+5\log x}{x^4+1} = \mathcal{O}(\frac{x^4}{x^4}) = \mathcal{O}(1) \Rightarrow n = 0$.

Problem 14.
a) No. Assume $x^3 \leq Cx^2$ for all $x > k$, where $k \in \mathcal{Z}^+$. Then $x \leq C$. For any chosen value of C, $\exists x \in \mathcal{Z}^+ \mid x > C$, since the set of positive integers is unbounded. Thus x^3 is not $\mathcal{O}(x^2)$.
b) Yes. $x^3 \leq Cx^3$, whenever $C = 1$ and $k = 1$.
c) Yes. Consider $x^3 \leq x^2 + x^3 \leq 2x^3 \leq Cx^3$. $x^3 \leq Cx^3$ for $C = 1$ and $k = 1$.
d) Yes. Consider $x^3 \leq x^2 + x^4 \leq 2x^4 \leq Cx^4$. $x^3 \leq Cx^4$ for $C = 1$ and $k = 1$.
e) Yes. $x^3 \leq C3^x$, whenever $C = 1$ and $k = 1$.
f) Yes. $x^3 \leq \frac{C}{2}x^3$, whenever $C = 2$ and $k = 1$.

Problem 19.
a) $(n^2+8)(n+1)$ is $\mathcal{O}(n^2 \cdot n) = \mathcal{O}(n^3)$.
b) $(n\log n + n^2)(n^3+2)$ is $\mathcal{O}(n^2 \cdot n^3) = \mathcal{O}(n^5)$.
c) $(n! + 2^n)(n^3 + \log(n^2+1))$ is $\mathcal{O}(n! \cdot n^3) = \mathcal{O}(n^n \cdot n^3) = \mathcal{O}(n^{n+3})$.

Problem 20.
a) $(n^3 + n^2\log n)(\log n + 1) + (17\log n + 19)(n^2+2)$ is
$\mathcal{O}(\max(n^3\log n, n^2\log n)) = \mathcal{O}(n^3\log n)$.
b) $(2^n + n^2)(n^3 + 3^n)$ is $\mathcal{O}(2^n \cdot 3^n) = \mathcal{O}(2^n 3^n)$.
c) $(n^n + n2^n + 5^n)(n! + 5^n)$ is $\mathcal{O}(n^n \cdot n!) = \mathcal{O}(n^n \cdot n^n) = \mathcal{O}(n^{2n})$.

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

Proof. Consider $f(n) = n\log n$ and $g(n) = \log n!$. $f(n)$ is $\mathcal{O}(g(n))$ if $f(n) \leq Cg(n)$ for all $n > k$, where $C, k \in \mathcal{Z}^+$.
$g(n) = \log n! = \log 1 + \log 2 + \cdots + \log n \leq \log n + \log n + \cdots + \log n = n\log n$.
Therefore, $f(n) = n\log n$ is $\mathcal{O}(\log n!)$, with $C = 1$ and $k = 1$ as witnesses. □