Big-O Analysis

1. Pushing n items to the Stack – $O(n^2)$

Consider the algorithm for pushing one item (x) to the stack:

Shift all array elements to the right by one index and put x at the front.

For an array with m elements, pushing an item to the stack would need exactly m assignments to move all of them to the right by one. Putting x at the front needs one extra assignment. In total, this operation would require exactly m + 1 assignments, thus resulting in a linear time complexity of O(m).

Doing that for every single item of the n elements we need to push to a stack of size m:

Element number	Stack size	Number of assignments to push that element
1	m	m + 1
2	m + 1	m + 2
3	m + 2	m + 3
…	…	…
n - 1	m + n – 2	m + n – 1
n	m + n – 1	m + n

As can be seen, pushing n number of elements requires a total of $(m+1)+(m+2)+\cdots+(m+n)=\sum_{i=1}^{n}(m+i)=\sum_{i=1}^{n}m+\sum_{i=1}^{n}i=m\cdot n+\frac{n(n+1)}{2}\approx n^2+m\cdot n$ number of assignments.

For $m \leq n$, this is essentially $O(n^2)$ in time complexity, otherwise it's $O(m \cdot n)$, which are quadratic and multilinear, respectively.

Assuming the stack is initially empty (m == 0), pushing n items would take quadratic time $O(n^2)$.

2. Popping those n items from the Stack – $O(n^2)$

Consider the algorithm for popping one item from the stack:

Extract the first element to return. Shift all other elements to the left by one index.

For an array with m elements, popping a single item from the stack would need exactly m – 1 assignments to move all the others to the left by one. Returning the desired element requires an extra copy/move construction. If we assume the cost of an assignment to be equal to that of a construction, in total, this operation will require exactly m assignments, thus resulting in a linear time complexity of O(m).

Doing that for every single one of the n elements we need to pop from a stack of size m + n:

Element number	Stack size	Number of assignments to pop that element
1	m + n	m + n
2	m + n – 1	m + n – 1
3	m + n – 2	m + n – 2
…	…	…
n - 1	m + 2	m + 2
n	m + 1	m + 1

As can be seen, popping those n elements requires $(m+n)+(m+n-1)+\cdots+(m+1) = \sum_{i=1}^{n}(m+i) = \sum_{i=1}^{n} m + \sum_{i=1}^{n} i = m \cdot n + \frac{n(n+1)}{2} \approx n^2 + m \cdot n$ number of assignments.

For $m \leq n$, this is essentially $O(n^2)$ in time complexity, otherwise it's $O(m \cdot n)$, which are quadratic and multilinear, respectively.

Assuming n items were pushed to an initially empty stack (m == 0), popping those same n items would take a quadratic amount of time O(n^2).

Figure 1: Running time (z) as a function of number of items to be pushed/popped (x) and number of items currently in the stack (y)

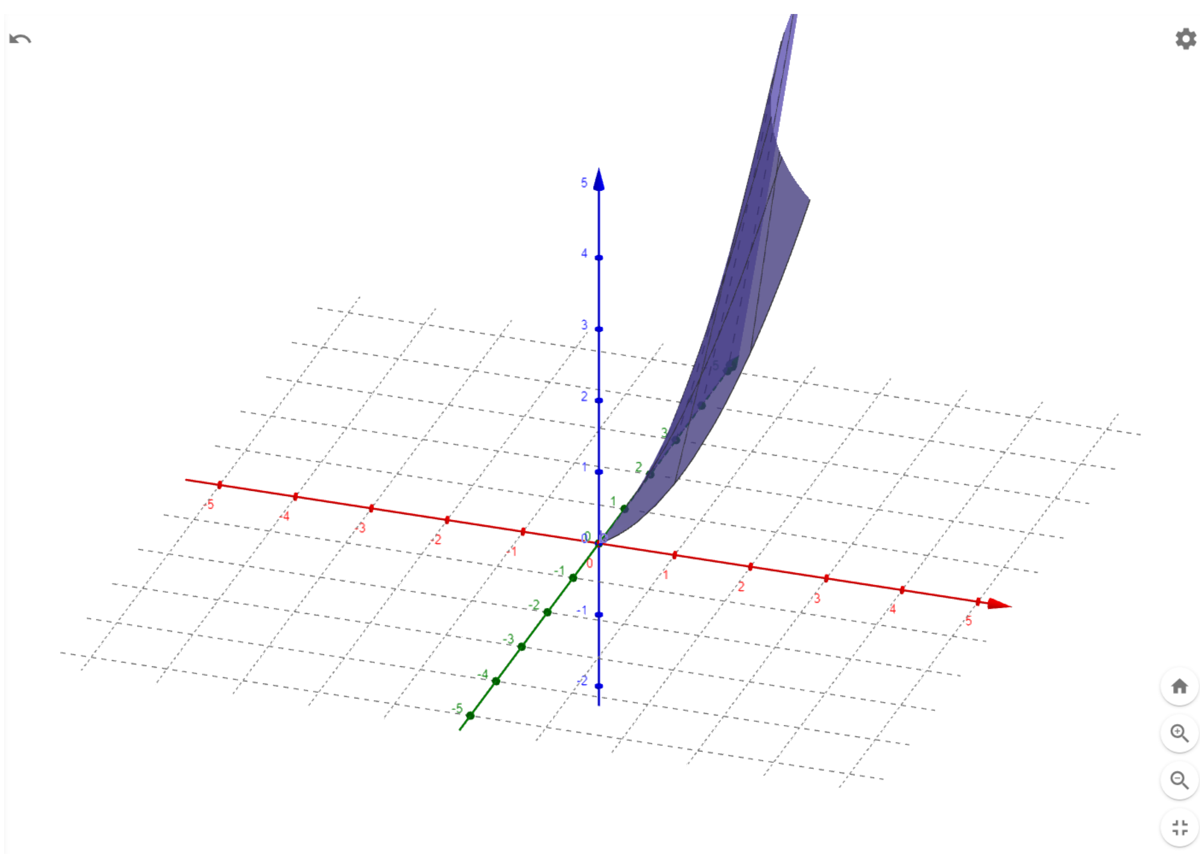
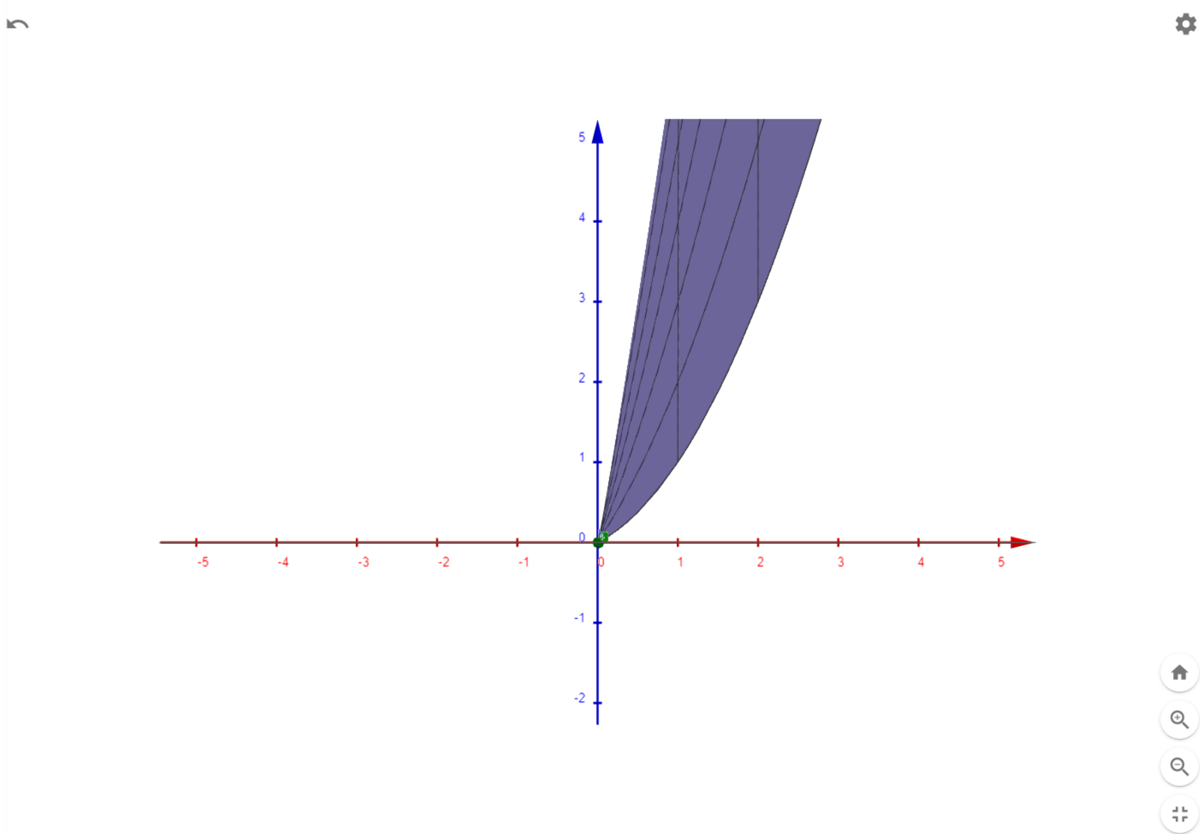
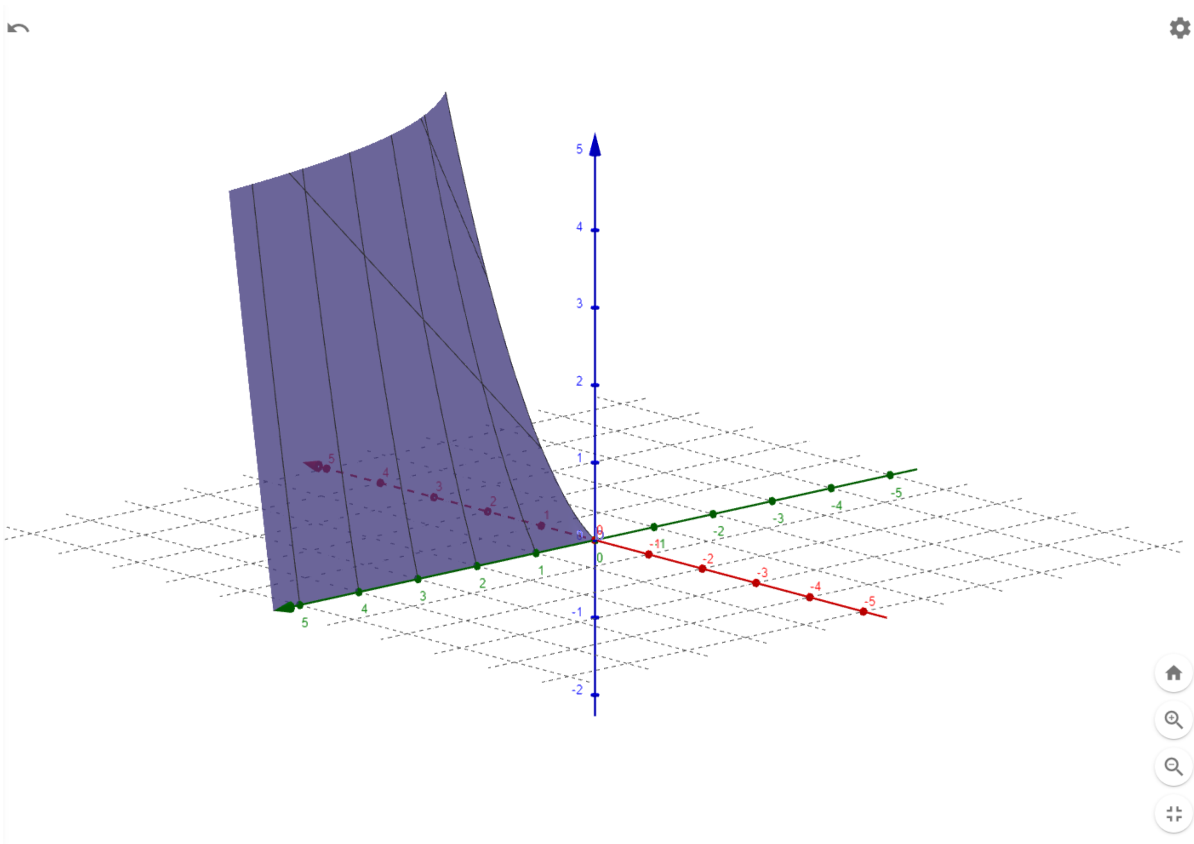
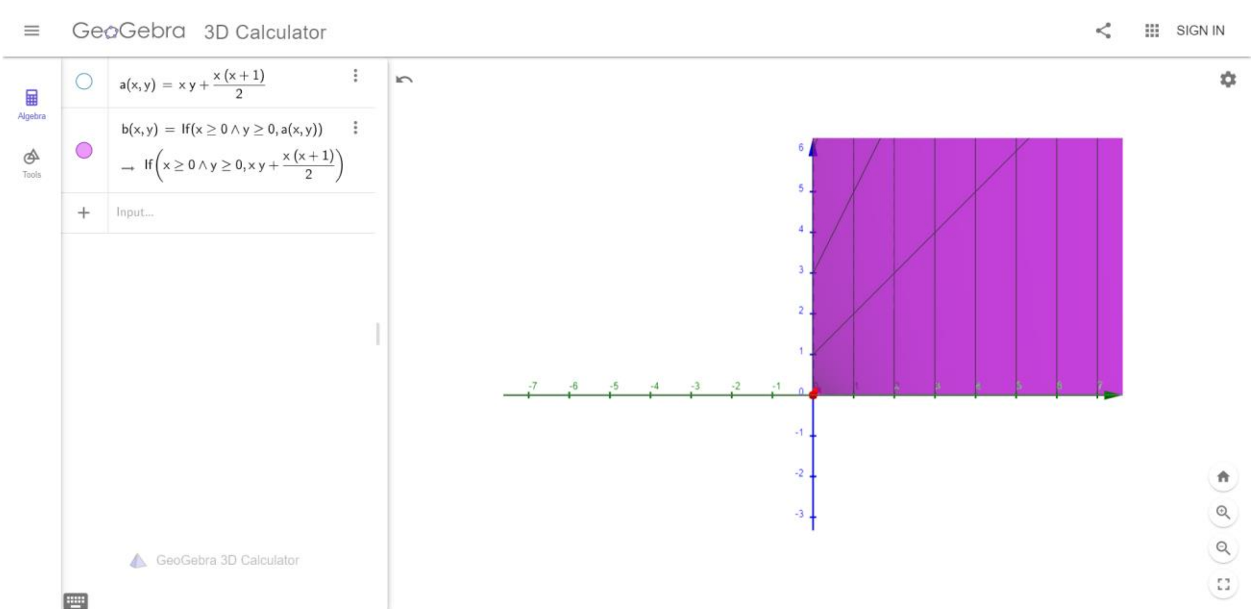


Figure 2: The rightmost curve represents the time needed to push n items to an initially empty stack






GeoGebra 3D Calculator
SIGN IN
Algebra
Tools
$a(x, y) = x\,y + \frac{x\,(x+1)}{2}$
$b(x, y) = \text{If}(x \geq 0 \land y \geq 0, a(x, y))$
$\rightarrow \text{If}\left(x \geq 0 \land y \geq 0, x\,y + \frac{x\,(x+1)}{2}\right)$
Input...
GeoGebra 3D Calculator