Amortized Analysis of Hysteresis used for stacks

implemented using
Dynamic resizing arrays.

-Ryan Azim Shaikh
2022MT11923

Algorithm:

In my solution for Stack_B (implementation of stack using dynamic resized array). I have used geometric expansion and shrinking of the internal array using hysteresis. I have used a growth factor of 2.

Initially the stack has a capacity of 1024.
When the array is full a new array is allocated with double (growth factor = 2) the size and the elements of the original array are copied to the new address. Then the old array is deallocated, and the internal array pointer is set to point to the new array.

When the number of elements (size) in the array become less than or equal to a fourth (growth factor * growth factor = 4) of the array length allocated (capacity) then a new array is allocated with half the size and the same copying of elements is done as above. A check is made to ensure that capacity of array is at least 1024.

Amortized Analysis:

I will use the method of potentials to give a proof of efficiency for the algorithm.
define the potential function ϕ(S) as

$$\phi(\mathrm{S}) = max((2 * size - capacity), \frac{capacity}{2} - size)$$

for non-empty array. And

$$\phi(\mathrm{S}) = 0$$

for an empty array.

Amortized time is given by

$$T_A = T + \phi(S') - \phi(S)$$

s = size before operation
c = capacity before operation

Push:
if a resize does not occur:

Case 1

$$s > \frac{c}{2}$$

$$\phi(S) = 2 * s - c$$

$$T_A = 2 + 2 * (s + 1) - c - 2 * s + c = 4$$

$$= O(1)$$

Case 2

$$s < \frac{c}{2}$$

$$\phi(S) = \frac{c}{2} - s$$

$$T_A = 2 + \frac{c}{2} - (s + 1) - \frac{c}{2} + s = 1$$

$$= O(1)$$

If expansion occurs:

$$s \ = \ c$$

$$\phi(S) = 2 * s - c$$

$$T_A = (s + 1) + 2 * (s + 1) - 2 * c - 2 * s + c$$

$$= (s + 1) + 2 * (s + 1) - 2 * s - 2 * s + s = 3$$

$$= O(1)$$

Pop:

If resize does not occur:

Case 1

$$s > \frac{c}{2}$$

$$\phi(S) = 2 * s - c$$

$$T_A = 2 + 2 * (s - 1) - c - 2 * s + c = 0$$

$$= O(1)$$

Case 2

$$s < \frac{c}{2}$$

$$\phi(S) = \frac{c}{2} - s$$

$$T_A = 2 + \frac{c}{2} - (s - 1) - \frac{c}{2} + s = 3$$

$$= O(1)$$

If shrinking occurs:

$$s = \frac{c}{4} + 1$$

$$\phi(S) = \frac{c}{2} - s$$

$$c = 4 * (s - 1)$$

$$T_A = (s + 1) + \frac{c}{4} - (s - 1) - \frac{c}{2} + s$$

$$= (s + 1) + (s - 1) - (s - 1) - 2 * (s - 1) + s = 3$$

$$= O(1)$$

Conclusion:

The algorithm is Amortized O(1) for both push and pop operations in all cases.

Hence it is an efficient algorithm.