Amortized Analysis of push() and pop() operations

For stack_b (dynamically-sized, min_capacity = 1024):

Let capacity = c;

For push() operation,

let us assume we have to insert N (very large) elements into the stack, so till number of push operations become c/2 – 1, the time taken for each push operation is constant (O(1)). In next push operation, the time taken will be (2c) + (c/2 – 1) + (1).

So, total cost of N pushes will be –

No. of pushes taking constant time + No. of pushes taking (2c + c/2 – 1 + 1) time

= 1*(N – N/1024) + (2c + c/2 – 1 + 1)*(N/1024)

= O(N)

Therefore, amortized cost for each push will be O(1).

For pop() operation,

if we have N (= capacity) elements in the stack, popping 3N/4 – 1 elements, the time taken will be constant (O(1)). The next pop will take c/2 + c/4 + 1 + 1 (assuming c/4 >= 1024) amount of time.

Total cost of N pops will be –

(Ignoring some of the constant terms)

3/4 * (N/4 + N/4^2 + N/4^3 + ………) + 1/4 * (N/4 + N/4^2 + N/4^3 +………) = N (1/(1-1/4))

= O(N)

Therefore, amortized cost for each pop will be O(1).