Proving that stack B has amortized O(1) time complexity

Ayush Gupta
2022CS11114

1 Aim

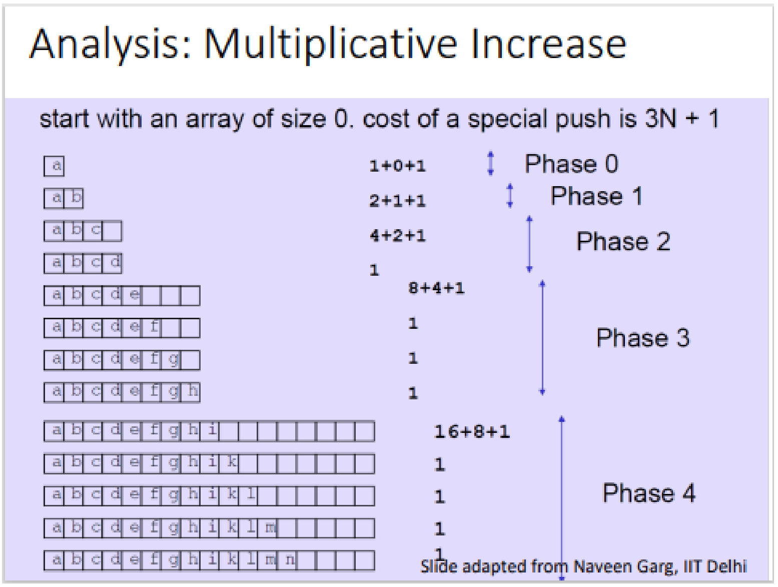
To show that push and pop of stack B has amortized O(1) average *amortized analysis*.

2 Checking Push Operation

If we do a push operation, there are two possibilities-

1. Size of stack is equal(or greater) than allocated capacity.
2. Size of stack is less than allocated capacity

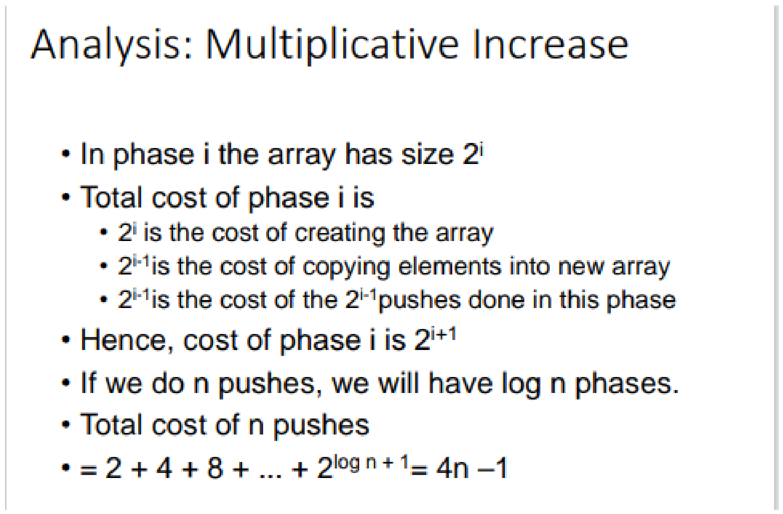
In the first case, we will need to increase the capacity allocated and make it double so that the element can be pushed. Now, if we make stack bigger than before, it takes a bit more time because here we have to copy each element from older stack to newer stack. But the main thing to note here is that we do this very rarely, as we are doing multiplicative increase here. It comes out to be of O(1). Below is the screenshot of the proof(taken from Lecture 6) of how it comes out to be of O(1).



Analysis: Multiplicative Increase

start with an array of size 0. cost of a special push is 3N + 1

Slide adapted from Naveen Garg, IIT Delhi



Analysis: Multiplicative Increase

- In phase i the array has size 2^i
- Total cost of phase i is
  - 2^i is the cost of creating the array
  - $2^{i-1}$is the cost of copying elements into new array
  - $2^{i-1}$is the cost of the 2^{i-1}pushes done in this phase
- Hence, cost of phase i is 2^{i+1}
- If we do n pushes, we will have log n phases.
- Total cost of n pushes
- $= 2 + 4 + 8 + \dots + 2^{\log n + 1} = 4n - 1$

3 Checking Pop Operation

Similarly, when we pop something from the stack it takes very less time. But sometimes we also have to make our stack smaller, which takes a bit of time. But same as push , it occurs very rarely in pop too.

4 Conclusion

So finally, we can conclude that on average, each push and pop operation doesn't take too long. Even though resizing the stack sometimes takes a bit more time, it doesn't happen all the time. This is why we say our stack is *amortized O(1)*.