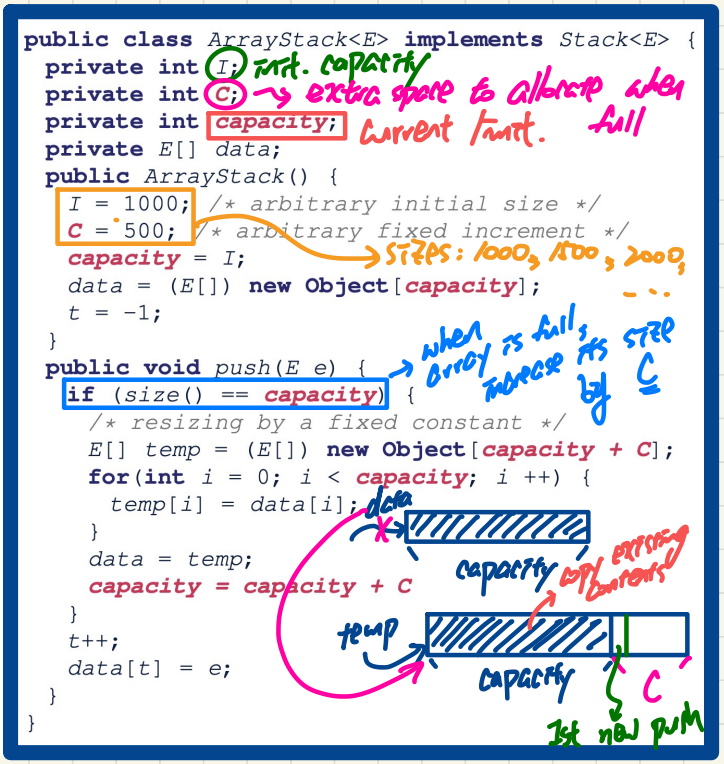
# Amortized Analysis: Dynamic Array with Const. Increments

```
1  public class ArrayStack<E> implements Stack<E> {
2    private int I;
3    private int C;
4    private int capacity;
5    private E[] data;
6    public ArrayStack() {
7      I = 1000; /* arbitrary initial size */
8      C = 500; /* arbitrary fixed increment */
9      capacity = I;
10     data = (E[]) new Object[capacity];
11     t = -1;
12   }
13   public void push(E e) {
14     if (size() == capacity) {
15       /* resizing by a fixed constant */
16       E[] temp = (E[]) new Object[capacity + C];
17       for(int i = 0; i < capacity; i ++) {
18         temp[i] = data[i];
19       }
20       data = temp;
21       capacity = capacity + C
22     }
23     t++;
24     data[t] = e;
25   }
26 }
```




initial array: I

1st resizing: I C

2nd resizing: I C C

3rd resizing: I C C C

Last resizing: I C C C ··· C C

Amortized/ Average RT:

W.L.O.G, assume: n pushes

and the last push triggers the last resizing routine.

# Deriving the Sum of a Geometric Sequence

Initial Term: I

Common Factor: r

Number of Terms: k

# Worst-Case RT: BST with Linear Height


Example 1: Inserted Entries with Decreasing Keys

<100, 75, 68, 60, 50, 1>

Example 2: Inserted Entries with Increasing Keys

<1, 50, 60, 68, 75, 100>

Example 3: Inserted Entries with In-Between Keys

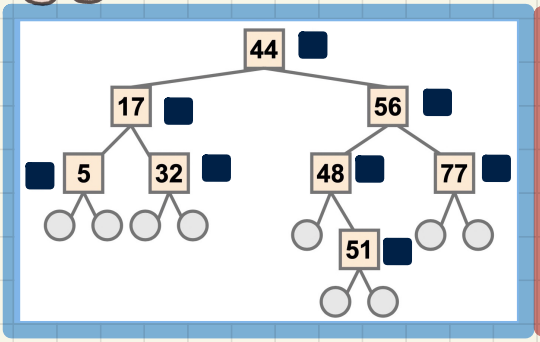
<1, 100, 50, 75, 60, 68>

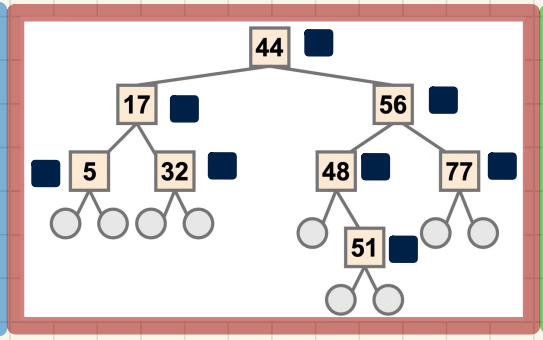
# Balanced BST: Definition

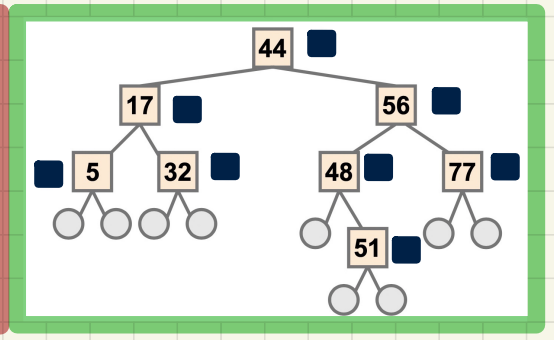
- internal node
- height
- height balance

Given a node $p$, the ***height*** of the subtree rooted at $p$ is:

$$\textit{height}(p) = \begin{cases} 0 & \text{if } p \text{ is } \textit{external} \\ 1 + \textbf{MAX}\ (\{\ \textit{height}(c) \mid \textit{parent}\ (c) = p\ \}) & \text{if } p \text{ is } \textit{internal} \end{cases}$$




Q. Is the above tree a **balanced BST**?

Q. Still a **balanced BST** after inserting **55**?

Q. Still a **balanced BST** after inserting **63**?