Asymptotic Upper Bound (Big-O): Alternative Formulation

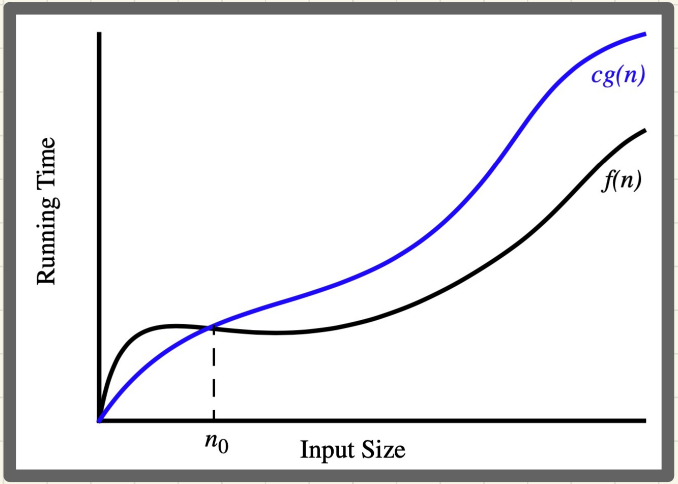
Known:

$f(n) \in O(g(n))$ if there are:

- A real *constant* $c > 0$
- An integer *constant* $n_0 \geq 1$

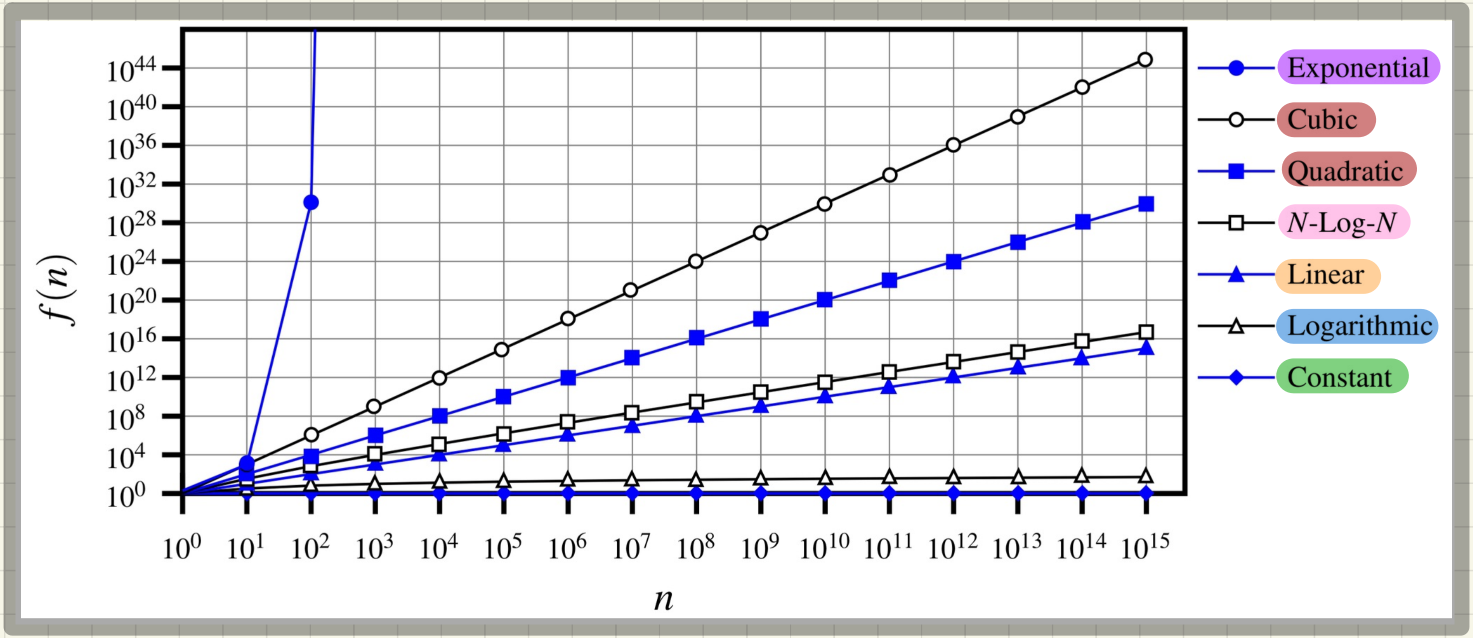
such that:

$$f(n) \leq c \cdot g(n) \quad \text{for } n \geq n_0$$



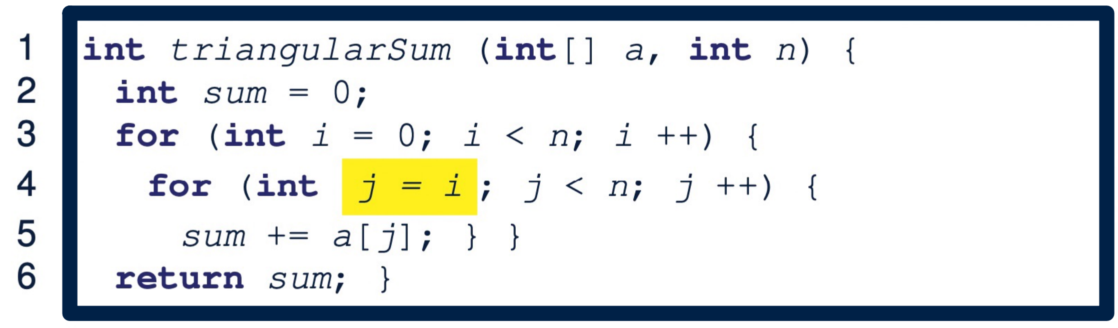
Q. **Formulate** the definition of "**f(n)** is order of O(**g(n)**)" using **logical** operator(s): ¬, ∧, ∨, ⇒, ∀, ∃

RT Functions: Rates of Growth (w.r.t. Input Sizes)



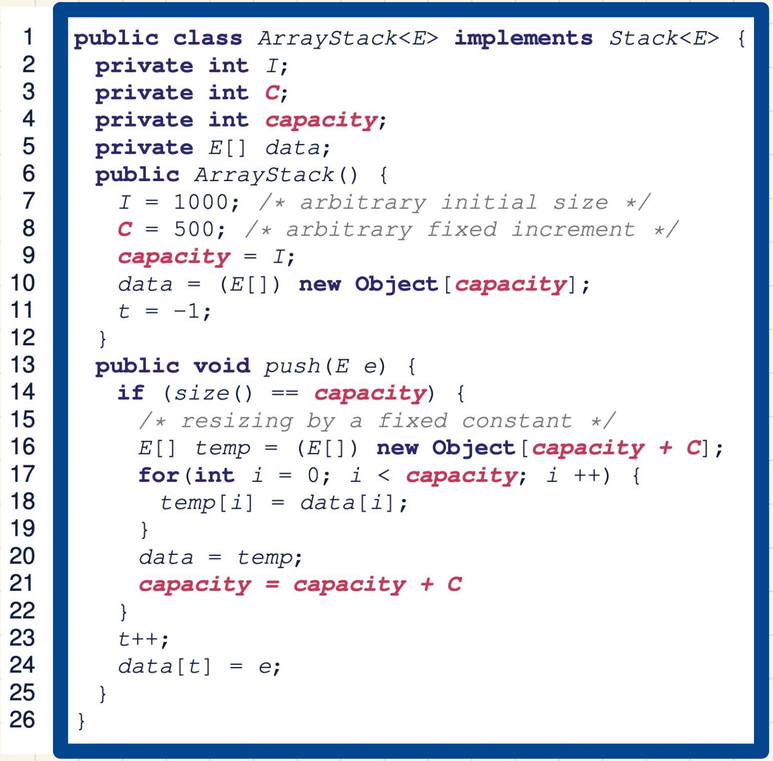
Asymptotic Upper Bound: Arithmetic Sequence/Progression

Determining the Asymptotic Upper Bound (3)

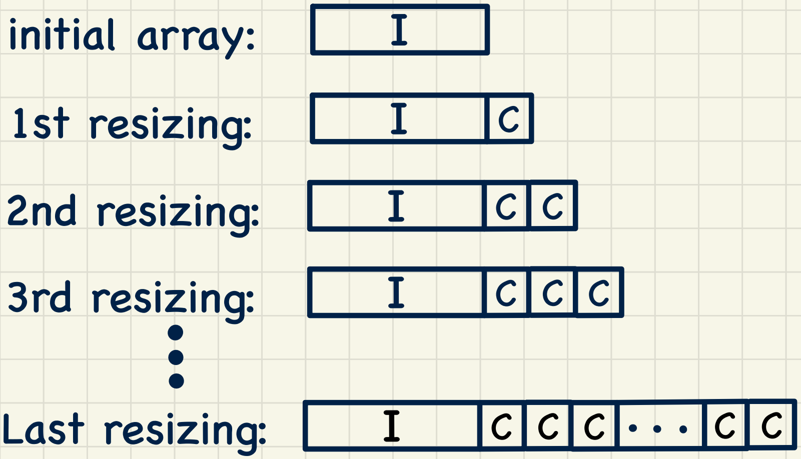


```
int triangularSum (int[] a, int n) {
  int sum = 0;
  for (int i = 0; i < n; i ++) {
    for (int j = i; j < n; j ++) {
      sum += a[j]; } }
  return sum; }
```

Amortized Analysis: Dynamic Array with Const. Increments



```java
public class ArrayStack<E> implements Stack<E> {
  private int I;
  private int C;
  private int capacity;
  private E[] data;
  public ArrayStack() {
    I = 1000; /* arbitrary initial size */
    C = 500; /* arbitrary fixed increment */
    capacity = I;
    data = (E[]) new Object[capacity];
    t = -1;
  }
  public void push(E e) {
    if (size() == capacity) {
      /* resizing by a fixed constant */
      E[] temp = (E[]) new Object[capacity + C];
      for(int i = 0; i < capacity; i ++) {
        temp[i] = data[i];
      }
      data = temp;
      capacity = capacity + C
    }
    t++;
    data[t] = e;
  }
}
```



initial array:

1st resizing:

2nd resizing:

3rd resizing:

⋮

Last resizing:

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

and the last push triggers the last resizing routine.

Amortized/ Average RT: