Amortized Analysis

Learning Objectives

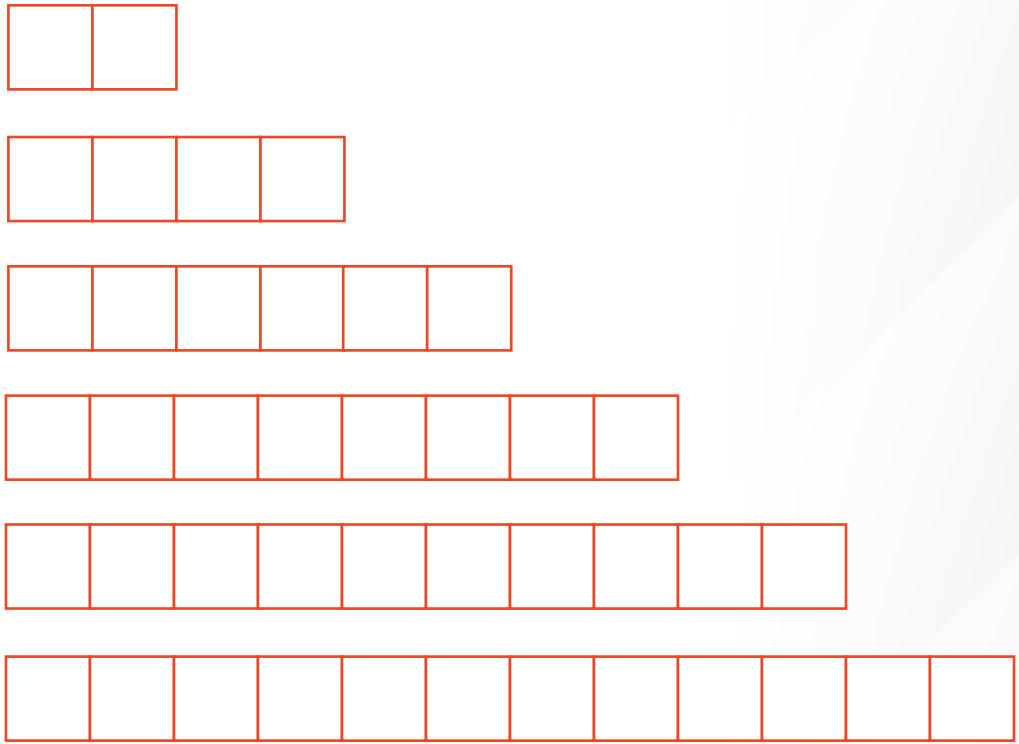
1. Define Amortized Analysis
2. Analyze the runtime of different implementations of Push_Back in an Array List

```
1  template <typename T>
2  void List<T>::push_back(const T & data){
3    if(size_ == capa_){
4      _addspace();
5    }
6
7    *size_++ = data;
8  };
9
10
11
12
13
14
15
```

Definition

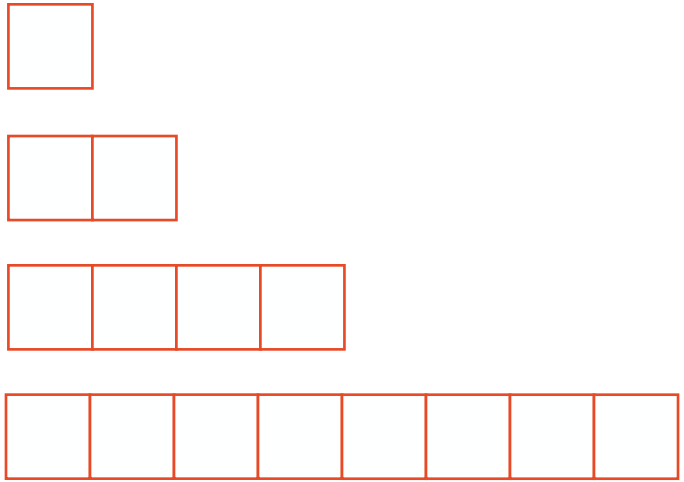
Given a sequence of operations, the amortized analysis gives the upper bound on the average run time per operation, which is equivalent to the total time divided by the number of operations.

Add_Space Strategy: +2



Add_Space Strategy: +2

Resize Strategy: x2



Resize Strategy: x2