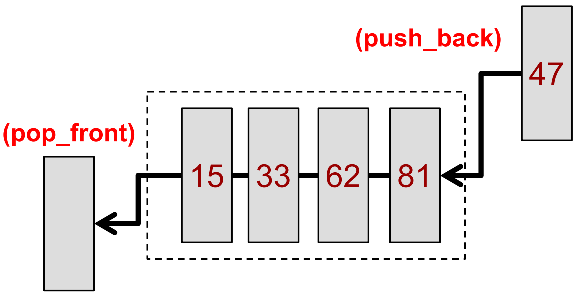
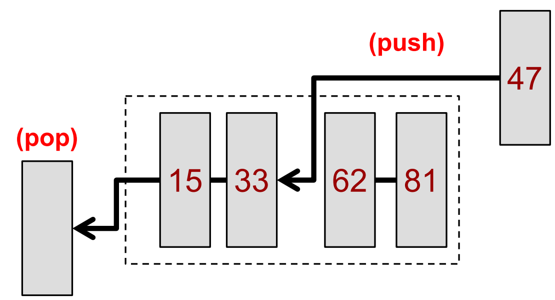
PRIORITY QUEUES

Traditional Queue

- Traditional Queues
  - Accesses/orders items based on POSITION (front/back)
  - Did not care about item's VALUE
- Priority Queue
  - Orders items based on VALUE
    - Either minimum or maximum
  - Items arrive in some arbitrary order
  - When removing an item, we always want the minimum or maximum depending on the implementation
    - Heaps that always yield the min value are called min-heaps
    - Heaps that always yield the max value are called max-heaps
  - Leads to a "sorted" list
  - Examples:
    - Think hospital ER, air-traffic control, etc.



Traditional Queue

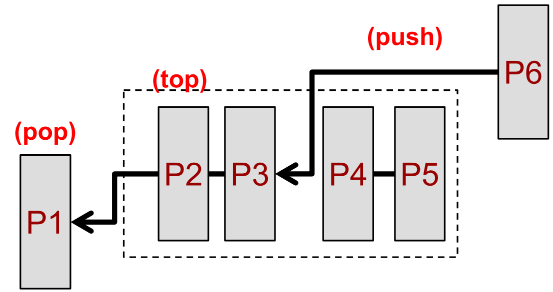


Priority Queue

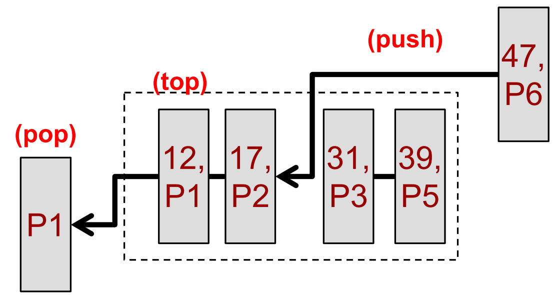
Priority Queue

- What member functions does a Priority Queue have?
  - push(item) – Add an item to the appropriate location of the PQ
  - top() – Return the min./max. value
  - pop() - Remove the front (min. or max) item from the PQ
  - size() - Number of items in the PQ
  - empty() - Check if the PQ is empty
  - [Optional]: changePriority(item, new_priority)
    - Useful in many algorithms (especially graph and search algorithms)
- Priority can be based on…
  - Intrinsic data-type being stored (i.e. operator<() of type T)
  - Separate parameter from data type, T, and passed in which allows the same object to have different priorities based on the programmer's desire (i.e. same object can be assigned different priorities)

```
class Patient {
public:
  bool operator<(...);
};
```



Priority Queue (Priority based on intrinsic property of the data)



Priority Queue (Priority based on separate priority parameter)

Priority Queue Efficiency

// back of array is min/max value (front)

- If implemented as a sorted array list
  - Insert() = O(n)
  - Top() = O(1)
  - Pop() = O(1)

n - # of items in pq

- If implemented as an unsorted array list
  - Insert() = O(1)
  - Top() = O(n)
  - Pop() = O(n)

Priority Queue Efficiency

- If implemented as a sorted array list
  - [Use back of array as location of top element]
  - Insert() = O(n)
  - Top() = O(1)
  - Pop() = O(1)
- If implemented as an unsorted array list
  - Insert() = O(1)
  - Top() = O(n)
  - Pop() = O(n)