

PRIORITY QUEUES

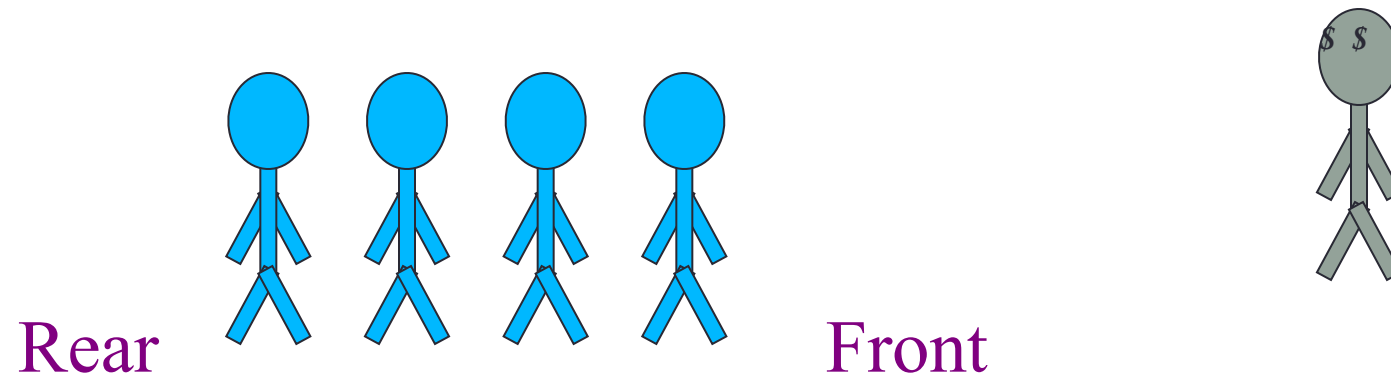
DATA STRUCTURE SELECTION

Final exam

- About the final exam: <https://ucsb-cs24-s19.github.io/exam/e03/>
 - PSYCH 1924 (Wed- June 12)
- Review session: Sunday, June 9th, 5p-7p
 - Phelps 2510

The Queue Operations

- A queue is like a line of people waiting for a bank teller.
- The queue has a front and a rear.



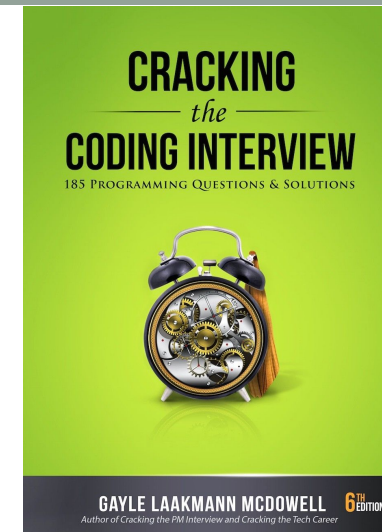
The Queue Class

- The C++ standard template library has a queue template class.
- The template parameter is the type of the items that can be put in the queue.

```
template <class Item>  
class queue<Item>  
{  
public:  
    queue( );  
    void push(const Item& entry);  
    void pop( );  
    bool empty( ) const;  
    Item front( ) const;  
    ...
```

Queue via stacks

Implement a MyQueue class which implements a queue using two stacks



Data structure Comparison

	Insert	Search	Min	Max	Delete min	Delete max	Delete (any)
Sorted array							
Unsorted array							
Sorted linked list (assume access to both head and tail)							
Unsorted linked list							
Stack							
Queue							
BST (unbalanced)							
BST (balanced)							
Min Heap							
Max Heap							

Selecting data structures

```
void mystery(vector<int>& a, int N){
    //Precondition: unsorted vector of size N

    for(int i =0; i<N; i++){ // N times
        int minElem = a[i];
        int index=i;
        for(int j = i+1; j<N;j++){
            if(a[j]<minElem){
                minElem = a[j];
                index = j;
            }
        }
        int tmp = a[i];
        a[i] = a[index];
        a[index]=tmp;
    }
}
```

Practice functors and PQs:

```
int main(){
    int arr[]={10, 2, 80};
    priority_queue<int*> pq;
    for(int i=0; i < 3; i++)
        pq.push(arr+i);

    while(!pq.empty()){
        cout<<*pq.top()<<endl;
        pq.pop();
    }
    return 0;
}
```

What is the output of this code?

A. 10 2 80

B. 2 10 80

C. 80 10 2

D. 80 2 10

E. None of the above

Sort array elements using a pq storing pointers

```
int main(){
    int arr[]={10, 2, 80};
    priority_queue<int*> pq;
    for(int i=0; i < 3; i++)
        pq.push(arr+i);

    while(!pq.empty()){
        cout<<*pq.top()<<endl;
        pq.pop();
    }
    return 0;
}
```

How can we change the way pq prioritizes pointers?

Write a comparison class to print the integers in the array in sorted order

```
int main(){
    int arr[]={10, 2, 80};
    priority_queue<int*, vector<int*>, cmpPtr> pq;
    for(int i=0; i < 3; i++)
        pq.push(arr+i);

    while(!pq.empty()){
        cout<<*pq.top()<<endl;
        pq.pop();
    }
    return 0;
}
```

Data structure Comparison

	Insert	Search	Min	Max	Delete min	Delete max	Delete (any)
Sorted array	$O(N)$	$O(\log N)$	$O(1)$	$O(1)$	$O(N)$ if ascending order, else $O(1)$	$O(1)$ if ascending, else $O(N)$	$O(\log N)$ to find, $O(N)$ to delete
Unsorted array	$O(1)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
Sorted linked list (assume access to both head and tail)	$O(N)$	$O(N)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(N)$ to find, $O(1)$ to delete
Unsorted linked list	$O(1)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$ to find, $O(1)$ to delete
Stack	$O(1)$ - only insert to top	Not supported	Not supported	Not supported	Not supported	Not supported	$O(1)$ - Only the element on top of the stack
Queue	$O(1)$ - only to the rear of the queue	Not supported	Not supported	Not supported	Not supported	Not supported	$O(1)$ - only the element at the front of the queue
BST (unbalanced)	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
BST (balanced)	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
Min Heap	$O(\log N)$	Not supported	$O(1)$	Not supported	$O(\log N)$	Not supported	$O(\log N)$
Max Heap	$O(\log N)$	Not supported	Not supported	$O(1)$	Not supported	$O(\log N)$	$O(\log N)$