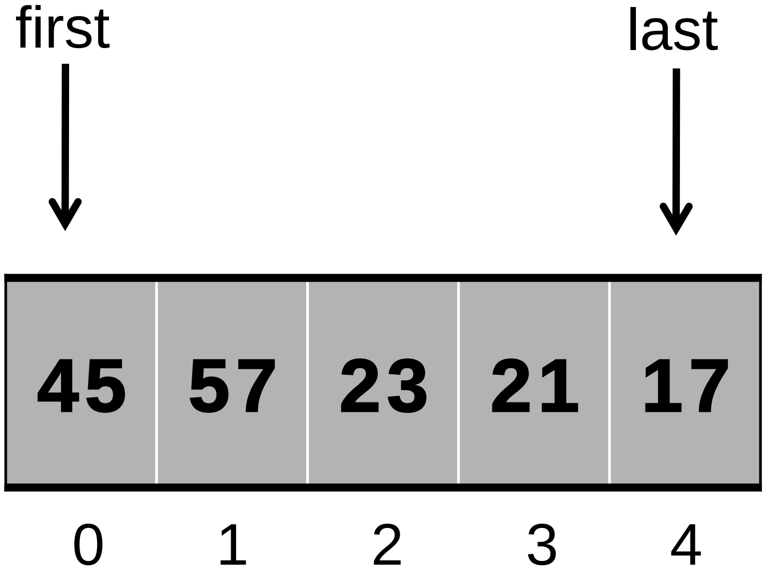
Implementing ADTs

Abstract Data Types

List

Interface:

```
void append(int x)
void prepend(int x)
void put(int x, int slot)
void remove(int x)
int getFirst()
int getLast()
int get(int slot)
boolean isEmpty()
```



```java
public class FixedLengthList{

      final int MAXSIZE = 100;
      int[] m_list= new int[100];
      int lastElement= -1;

      // Add new key to the list
      void append(int key){
         lastElement++;
         m_list[lastElement] = key;
      }

      // Search the list
      boolean contains(int key){
         // Linear search
         for (int i=0; i<=lastElement; i++){
            if (m_list[i] == key) return true;
         }
         return false;
      }
```

```
public class FixedLengthList{

      final int MAXSIZE = 100;
      int[] m_list= new int[100];
      int lastElement= -1;

      // Add new key to the list
      void append(int key){
         lastElement++;
         m_list[lastElement] = key;
      }

      // Search the list
      boolean contains(int key){
         // Linear search
         for (int i=0; i<=lastElement; i++){
            if (m_list[i] == key) return true;
         }
         return false;
      }
```

```
public class FixedLengthList{
      final int MAXSIZE = 100;
      int[] m_list= new int[100];
      int lastElement= -1;

      // Add new key to list
      void append(int key) {
         if (lastElement<MAXSIZE-1) {
            lastElement++;
            m_list[lastElement] = key;
         }
         else {
            System.out.println("Error: overfull list.");
         }
      }
      // Search list
      boolean contains(int key){
         // Linear search
         for (int i=0; i<=lastElement; i++){
            if (m_list[i] == key) return true;
         }
         return false;
      }
```

```
public class FixedLengthList{
      final int MAXSIZE = 100;
      int[] m_list = new int[100];
      int lastElement= -1;

      // Add new key to list
      void append(int key){
         if (lastElement<MAXSIZE-1) {
            lastElement++;
            m_list[lastElement] = key;
         }
         else{
            System.out.println("Error: overfull list.");
         }
      }
      // Search list
      boolean contains(int key) {
         // Linear search
         for (int i=0; i<=lastElement; i++){
            if (m_list[i] == key) return true;
         }
         return false;
      }
```

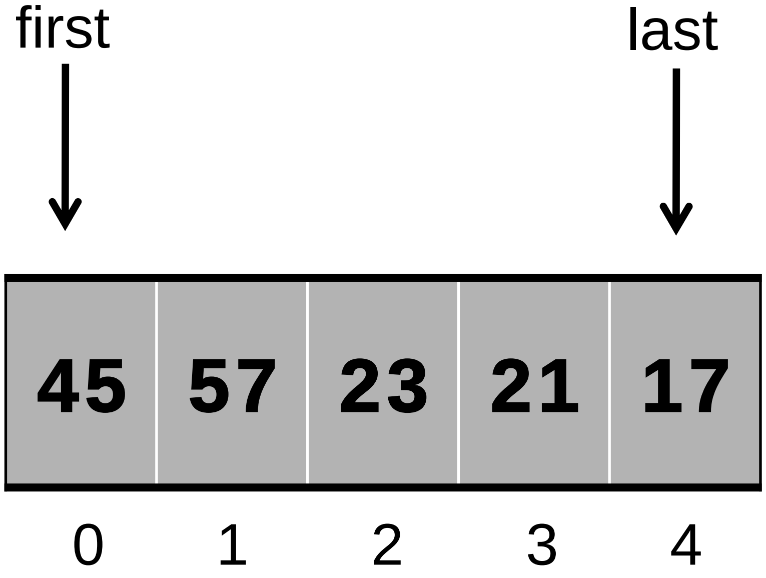
```
// Remove key in specified slot
public void remove(int elementNumber){

   // Do error checking
   ...

   // Move every item over by one
   for (int i=elementNumber; i<lastElement; i++){
      m_list[i] = m_list[i+1];
   }

   // Decrement lastElement and return
   lastElement--;
   return;
}
```

Abstract Data Types

List

Interface:

```
void append(int x)
void prepend(int x)
void put(int x, int slot)
void remove(int x)
int getFirst()
int getLast()
int get(int slot)
boolean isEmpty()
```



45	57	23	21	17
0	1	2	3	4

What is the cost of adding an item to the beginning of the list in this implementation?

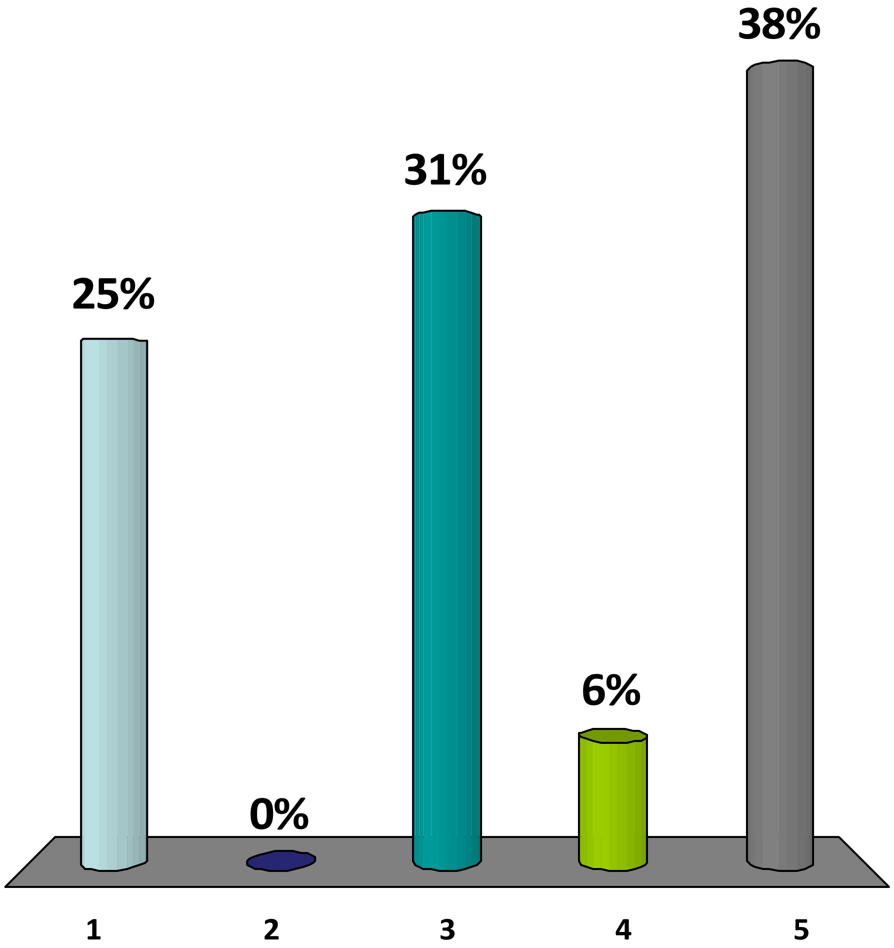
1. O(log n)

✓2. O(n)

3. O(n log n)

4. $O(n^2)$

5. $O(2^n)$



Implementing a Stack

Stack (of integers) :

```
class Stack{
    int[1000] stackArray;
    int top = 0;
```

```
boolean empty()
    return (top==0);
```

```
void push(int x)
    top++;
    stackArray[top] = x;
```

```
int pop()
    int i = stackArray[top];
    top--;
    return i;
```

Implementing a Stack

Stack (of integers) :

```
class Stack{
    int[1000] stackArray;
    int top = 0;
```

```
boolean empty()
    return (top==0);
```

```
void push(int x)
    top++;
    stackArray[top] = x;
```

What if stack is empty?

```
int pop()
    int i = stackArray[top];
    top--;
    return i;
```

Implementing a Stack

Stack (of integers) :

What if stack has 1001 elements?

```
class Stack{
   int[1000] stackArray;
   int top =0;
```

```
boolean empty()
   return (top==0);
```

```
void push(int x)
   top++;
   stackArray[top] = x;
```

```
int pop()
   int i= stackArray[top];
   top--;
   return i;
```

Implementing a Queue…

Queue

Interface:

- void enqueue(element x)
- element dequeue()

Exercise…