Welcome!

September 6, 2018

Github Setup

If you haven't already, set up a github account at: https://github.com/

Login, and join our Organization:
https://github.com/Algorithms-for-CP-and-Interviews

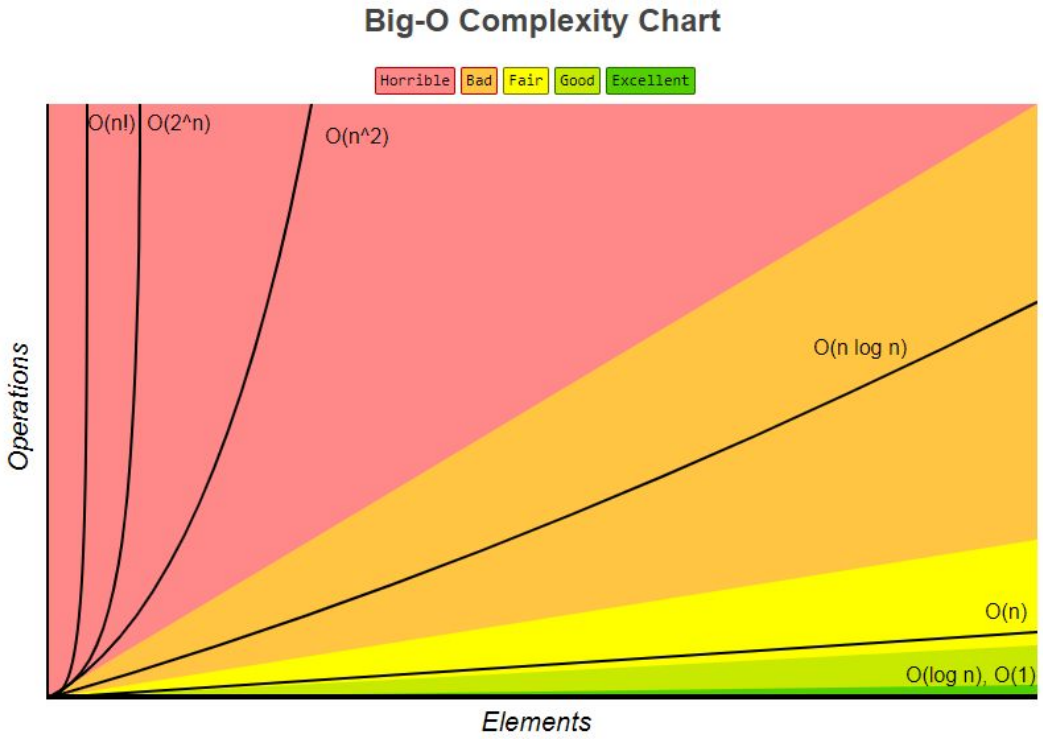
If you've joined the group for extra credit, this is how we will track your progress

Outline

- Growth of Functions/Asymptotic Concepts
- Insertion Sort & Merge Sort
- Basic ADTs

Growth of Functions/Asymptotic Concepts

- Use different symbols
  - Quantifies requirements (e.g. space, time, etc)
- Allows for definition of worst, average, best case



http://bigocheatsheet.com/

Growth of Functions/Asymptotic Concepts

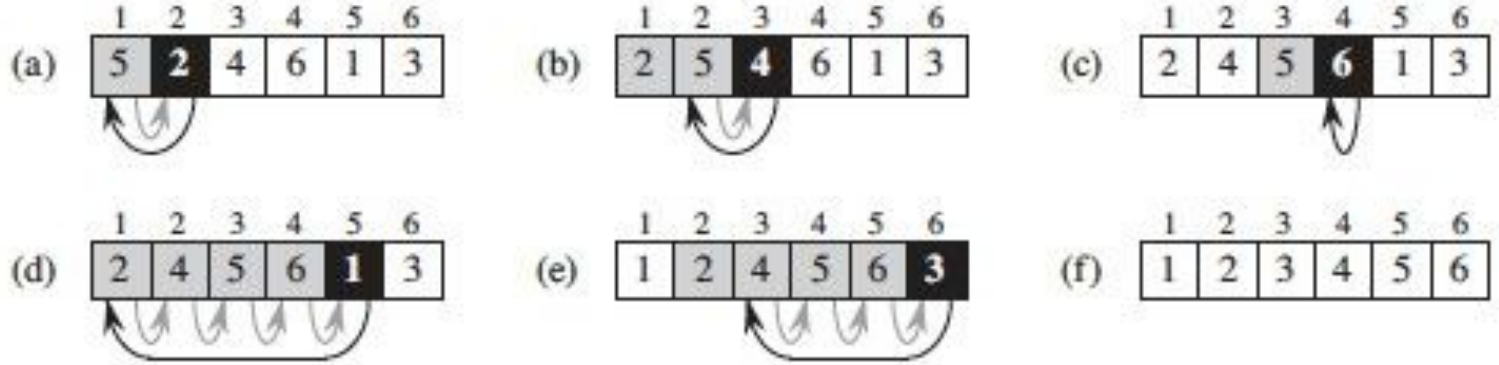
- $O \approx \leq$
  - Upper bound/equal to
  - Ex: $O(n^2) = n^2, n^2 + 4324, n^{1.999}$
- $\Omega \approx \geq$
  - Lower bound/equal to
  - Ex: $\Omega(n^2) = n^2, n^2 + 100, n^3$
- $\Theta \approx =$
  - Average bound
  - Ex: $\Theta(n^2) \rightarrow n^2, n^2 + 4329482$
- $o \approx <$
  - Strictly upper bound
  - Ex: $o(n^2) = n, n^{1.999}$
- $\omega \approx >$
  - Strictly lower bound
  - Ex: $\omega(n^2) = n^3, 4n^3 + nlgn$

Insertion Sort

- Iterative
- Two parts in thing to be sorted
  - Sorted & unsorted part
- Steps
  - Select first element from unsorted portion
  - Place element in appropriate spot in sorted portion
  - Repeat until all elements are sorted

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Insertion Sort



Insertion Sort

- Best case → O(n)
  - Elements already sorted
  - Process each element once
- Worst case → $O(n^2)$
  - Elements in reversed sorted order
- Average case
  - Usually the same as worst case

Merge Sort

- Divide and Conquer
  - Cut problem into smaller chunks
  - Solve smaller chunks first
  - Combine solved chunks
- Steps
  - Divide → Split array into 2 subarrays
  - Conquer → Recursively sort subarrays
  - Combine → Merge sorted subarrays

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

Merge Sort and Merge Procedure

6 5 3 1 8 7 2 4

Runtime of Merge Procedure: O(n)

```
MERGE(A, p, q, r)
 1  n_1 = q − p + 1
 2  n_2 = r − q
 3  let L[1 .. n_1 + 1] and R[1 .. n_2 + 1] be new arrays
 4  for i = 1 to n_1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n_2
 7      R[j] = A[q + j]
 8  L[n_1 + 1] = ∞
 9  R[n_2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

Merge Sort

- Best case time complexity → $\Omega(n \log(n))$
- Average case time complexity → $\Omega(n \log(n))$
- Worst case time complexity → $\Omega(n \log(n))$

Insertion & Merge Sort

- Insertion Sort

  Better on smaller arrays with a best case of O(n) or better in conditions where the input expected is sorted

- Merge Sort

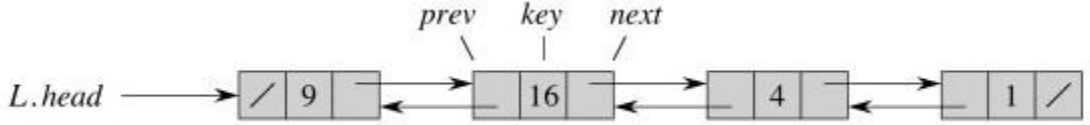
  Each case: Best, Average, Worst has the same asymptotic runtime complexity of O(nlog(n))

Basic ADTs ⟶ Linked Lists

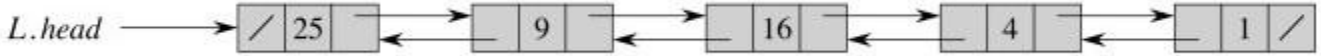
- Stores information in a linear order
- Can be sorted or unsorted
- Linked List implementation
  - Singular
    - Nodes have a next pointer pointing to next element
  - Doubly
    - Nodes have a prev (previous) and next pointer
  - Has a head and possibly tail pointer

Basic ADTs ⟶ Linked Lists

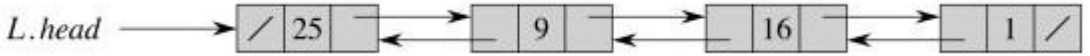
Linked List Implementation



Inserting 25



Deleting 4

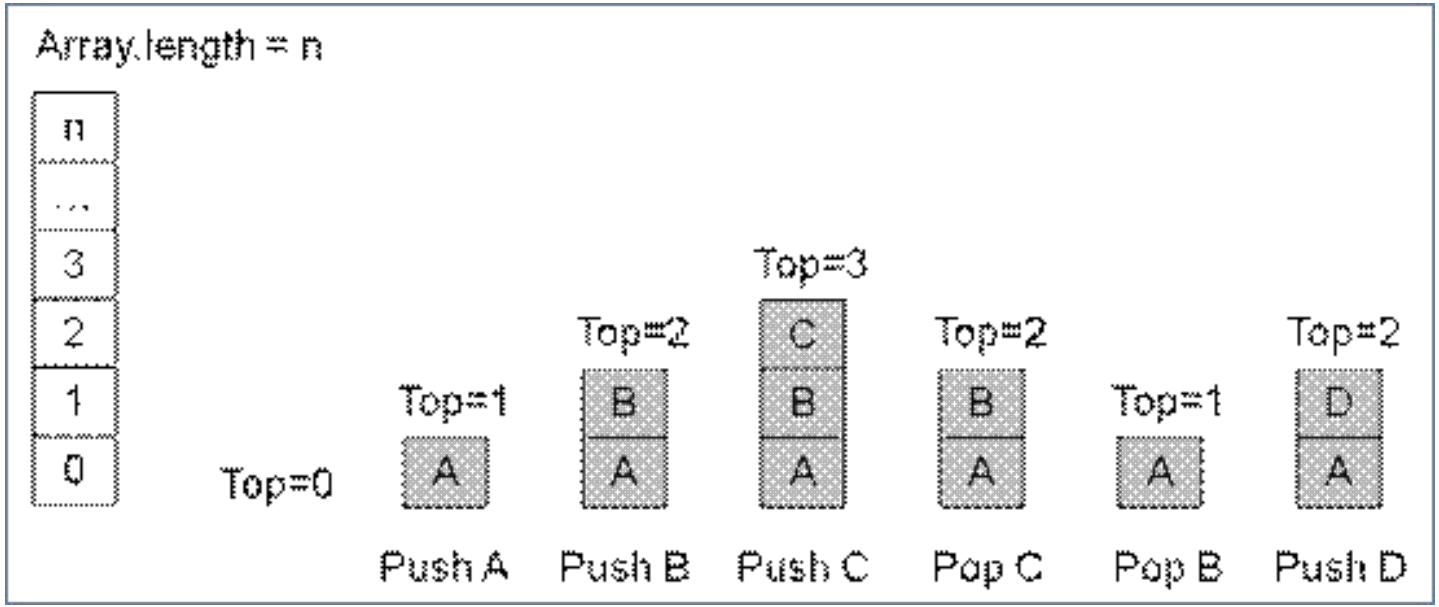


Basic ADTs ⟶ Stacks

- public Stack();
  - Creates an instance of a stack and initializes
- public void push(Object o);
  - Inserts o at the top of the stack
- public Object pop();
  - Removes and returns top of the stack
- public int size();
  - Returns the number of elements in the stack
- public boolean isEmpty();
  - Returns true if stack is empty
- public Object top();
  - Returns top of stack without removal
  - Returns error if isEmpty() == true

Basic ADTs ⟶ Stacks

- LIFO (last in first out)



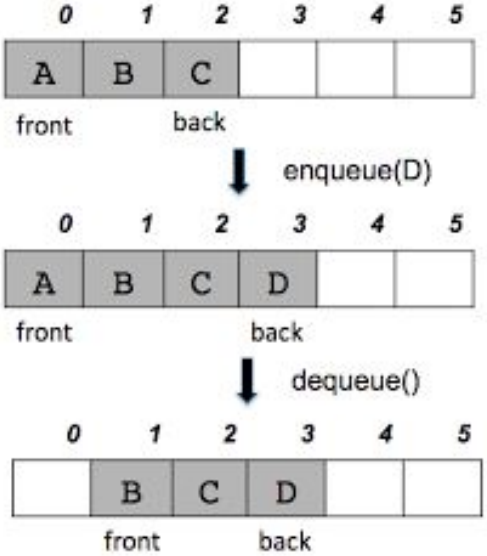
https://cs.uwec.edu/~stevende/cs245/labs/lab04-StackQDS/stackQds.htm

Basic ADTs ⟶ Queues

- public Queue();
  - Create an instance of ADT Queue and initialize it to the empty queue.
- public void enqueue(Object o);
  - Insert o at the end of the queue
- public Object dequeue();
  - Removes & returns object at front of the queue
  - Error if queue is empty
- public int size();
  - Returns number of objects in queue
- public boolean isEmpty();
  - Returns true if queue is empty
- public Object front();
  - Returns front of the queue without removing it

Basic ADTs ⟶ Queues

- FIFO (First in first out)



https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTQ0kNoyWO53CcaWl-IHeJyl3MwZrpyhW5uW8qnG2uMtLv0tJTo

Practice

- https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/practice-problems/algorithm/chandu-and-his-girlfriend/

Interview Problem Practice

Question: Given two sorted arrays, find the number of elements in common. The arrays are the same length and each has all distinct elements.

Let's start with a good example. We'll underline the elements in common.

```
A: 13  27  35  40  49  55  59
B: 17  35  39  40  55  58  60
```