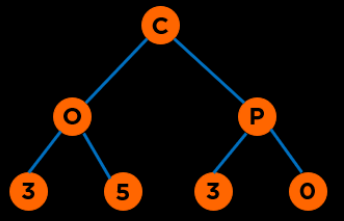
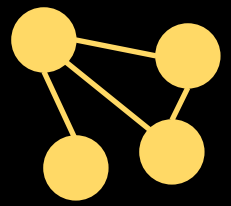
Exam 1 Review - A



Data Structures & Algorithm

Categories of Data Structures

Linear Ordered	Non-linear Ordered
Lists	Trees
Stacks	Heaps
Queues	



Agenda

- Record this Lecture
- Announcements
- Exam 1 Logistics
- Exam Review A: Module 1 and 2

Exam 1 Logistics

- Exam 1 is on Feb 21 (3-10 pm, Start by 8 pm)
  - 4 blank sheets of paper are allowed
  - Honorlock
  - Topics: Module 1-4 (includes Balanced Trees)
  - You can use both Iteration or Recursion to solve problems

- Read the Topics and Expectations Guide on Canvas

Exam 1 Topics and Expectations

Algorithm Analysis

- Analyze the Computational Complexity of a given code snippet
- Understand what is Big O notation and order of growth
- Identify functions that belong to the family of functions in Big O (we will not ask Big Theta, Big Omega or other notations in the exam)
- Know or infer the runtime in terms of Big O of algorithms and scenarios covered in Weeks 2-5 for best, average, and worst case
- Know how Linear and Binary Search Algorithms work
- Know the three methods of evaluating the time execution of an algorithm
- Comprehend and contrast the order of growth of a two or more functions

List, Stacks, and Queues

- Properties
- Insertion, Deletion, Traversal, Search for all types of List, Stacks, and Queues
- Ways of implementation
- Critically thinking when a certain type is better in terms of performance
- Pseudocodes for common operations or for solving a problem
- Use cases of Stacks to evaluate expressions, call stacks, balancing parentheses, and finding palindromes.
- Performance

Mini Review – Complexity

What is the computational complexity of adding an item to a stack in the worst case in terms of Big O notation?

Mini Review – Complexity

What is the computational complexity of adding an item to a stack in the worst case in terms of Big O notation?

Answer: O(1)

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
int k;
for (int i=1; i < n; i++)
      for (int j=n; j>1; j=j/2)
            k = i*j;
```

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
int k;
for (int i=1; i < n; i++)
      for (int j=n; j>1; j=j/2)
            k = i*j;
```

Answer: $O(n \log_2 n)$

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
int i = 1;
while (i < n)
{
   i = i * m;
}
```

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
int i = 1;
while (i < n)
{
   i = i * m;
}
```

Answer: $O(\log_m n)$

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
result = 0
for (int i = 0; i < n; i++)
  result += i;
for (int j = 1; j < m; j *= 2)
  result *= j;
```

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
result = 0
for (int i = 0; i < n; i++)
  result += i;
for (int j = 1; j < m; j *= 2)
  result *= j;
```

Answer: $O(n+\log_2 m)$

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
for (int i=n; i>0; i/=2)
      for (int j=1; j<i; j++)
            sum = sum + 1;
```

Mini Review – Complexity

What is the computational complexity of the following code snippet?

```
for (int i=n; i>0; i/=2)
       for (int j=1; j<i; j++)
              sum = sum + 1;
```

Answer: O(n)

Mini Review – Complexity

Which code snippet will take less time to execute on a computer?

A

```
for (int i=1; i < n; i++)
      for (int j=1; j < n; j++)
            sum = sum + 1;
```

B

```
for (int i=1; i < 2n; i++)
      for (int j=1; j < 2n; j++)
            sum = sum + 1;
```

Mini Review – Complexity

Which code snippet will take less time to execute on a computer?

A

```
for (int i=1; i < n; i++)
      for (int j=1; j < n; j++)
            sum = sum + 1;
```

B

```
for (int i=1; i < 2n; i++)
      for (int j=1; j < 2n; j++)
            sum = sum + 1;
```

Answer: A

Mini Review – Complexity

Which code snippet will have a higher growth rate asymtotically in terms of Big O notation?

A

```
for (int i=1; i < n; i++)
      for (int j=1; j < n; j++)
            sum = sum + 1;
```

B

```
for (int i=1; i < 2n; i++)
      for (int j=1; j < 2n; j++)
            sum = sum + 1;
```

Mini Review – Complexity

Which code snippet will have a higher growth rate asymptotically in terms of Big O notation?

A

```
for (int i=1; i < n; i++)
      for (int j=1; j < n; j++)
            sum = sum + 1;
```

B

```
for (int i=1; i < 2n; i++)
      for (int j=1; j < 2n; j++)
            sum = sum + 1;
```

Answer: A and B grow at the same rate asymtotically

Mini Review – Linked Lists

Consider a class List that implements an ordered list backed by a singly linked list with a head pointer. The invariant "ordered" is maintained always. Given that representation, what is the worst-case time complexity of the following operations? Assume the list is sorted in ascending order.

A. Insert an item
B. Finding the minimum element
C. Delete the largest element from list
D. Finding the largest element
E. Finding a random element, n
F. Deleting the minimum element in the list

Mini Review – Linked Lists

Consider a class List that implements an ordered list backed by a singly linked list with a head pointer. The invariant "ordered" is maintained always. Given that representation, what is the worst-case time complexity of the following operations? Assume the list is sorted in ascending order.

A. Insert an item : O(n)
B. Finding the minimum element : O(1)
C. Delete the largest element from list : O(n)
D. Finding the largest element : O(n)
E. Finding a random element, n : O(n)
F. Deleting the minimum element in the list : O(1)

Mini Review – Stacks

Postfix Evaluation “2 3 1 * + 9 -“. We scan all elements one by one.

Mini Review – Stacks

Postfix Evaluation “2 3 1 * + 9 -“. We scan all elements one by one.

1) Scan ‘2’, it’s a number, so push it to stack. Stack contains ‘2’
2) Scan ‘3’, again a number, push it to stack, stack now contains ‘2 3’ (from bottom to top)
3) Scan ‘1’, again a number, push it to stack, stack now contains ‘2 3 1’
4) Scan ‘*’, it’s an operator, pop two operands from stack, apply the * operator on operands, we get 3*1 which results in 3. We push the result ‘3’ to stack. Stack now becomes ‘2 3’.
5) Scan ‘+’, it’s an operator, pop two operands from stack, apply the + operator on operands, we get 3 + 2 which results in 5. We push the result ‘5’ to stack. Stack now becomes ‘5’.
6) Scan ‘9’, it’s a number, we push it to the stack. Stack now becomes ‘5 9’.
7) Scan ‘-‘, it’s an operator, pop two operands from stack, apply the – operator on operands, we get 5 – 9 which results in -4. We push the result ‘-4’ to stack. Stack now becomes ‘-4’.
8) There are no more elements to scan, we return the top element from stack (which is the only element left in stack).

Output Prediction / Coding Questions

https://onlinegdb.com/BJ4hyD7vP
https://onlinegdb.com/BJ6gewQDP

Mini Review – Coding questions

- Check whether a string is a Palindrome using a Stack.
- Write pseudocode for adding an element in the rear in a doubly linked list consisting of a head and tail.
- Write pseudocode or C++ code to pop an element from a Circular Queue implemented as an array.
- Design a Stack data structure that supports push, pop and min operations in O(1) time.

Questions