CSC 130: Data Structures and Algorithms Analysis

Anna Baynes

Spring 2018

Lecture 1

Today’s outline

- Introductions
- What is this course about?
- Admin
- Review: Queues and stacks
- Asymptotic Analysis

Introduction

- Who am I?
  - Anna Baynes
  - Assistant Professor
  - Past 5 years, Research and Dev Engineer at IBM – Data Science
  - Before that – Intel, University of Michigan, University of Washington

Class Overview

- Intro to many of the basic data structures in computer science
  - Understand the data structures
  - Analyze the algorithms that use them
  - Know when to apply them

Class Overview

- Practice design and analysis of data structures
- Practice using these data structures by writing programs
- Make the transformation from programmer to computer scientist

Goal

- You will understand
  - What the tools are for storing and processing common data types
  - Which tools are appropriate for which need
- So that you can
  - Make good design choices as a developer, project manager, or system customer

Roll Call

- Adding this class

Syllabus

- Canvas

Course mechanics

- Due 11:59pm electronically
  - Lates
- Programming Assignments
  - In Java
  - Submit Java Files!!
- Teamwork vs cheating

Homework 1 and Project 1

- Up online now

Data Structures

- “Clever” ways to organize information in order to enable efficient computation
  - What do we mean by clever?
  - What do we mean by efficient?

Picking the best Data Structure for the Job

- The data structure you pick needs to support the operations you need
- Ideally it supports the operations you will use most often in an efficient manner
- Abstract Data Type (ADT) - A data object and a set of operations for manipulating it
  - List ADT with operations insert and delete
  - Stack ADT with operations push and pop

Terminology

- Abstract Data Type (ADT)
  - Mathematical description of an object with set of operations on the object. Useful building block
- Algorithm
  - A high level, language independent, description of a step-by-step process
- Data structure
  - A specific family of algorithms for implementing an ADT
- Implementation of data structure
  - A specific implementation in a specific language

Why so many data structures?

- Ideal data structure:
  - “fast”, “elegant”, memory efficient
- Generates tensions
  - Time vs space Performance vs elegance
  - Generality vs simplicity
  - One operation's performance vs another's
  -

Review: Stacks and Queues

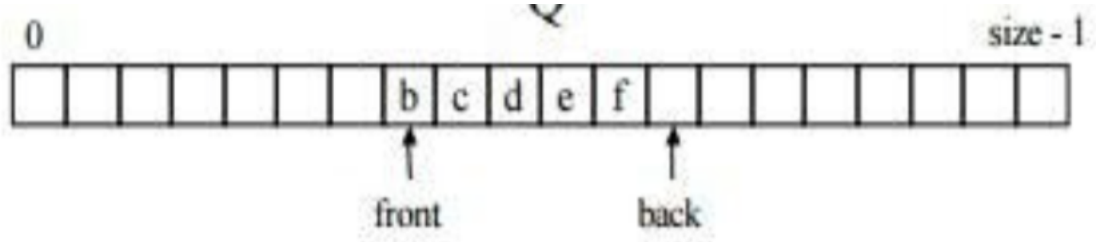
- Queue
- Stack
- Fifo/ Lifo
- What operations do they support?
- Implementation?

First Example: Queue ADT

- FIFO: First In, First Out
- Queue operations
  - create
  - destroy
  - enqueue
  - Dequeue
  - is_empty

Circular Array Queue Data Structure

- How to implement enqueue?
- How to implement dequeue?
- How to test if queue is empty?
- How to find K-th element in the queue?
- What is the complexity of these operations?
- What is the limitation of this data structure?

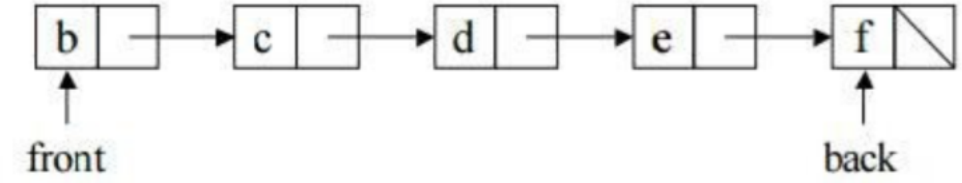


Circular Array Queue Data Structure

```
enqueue(Object x) {
  Q[back] = x;
  back = (back + 1) % size;
}

dequeue(){
  x = Q[front];
  front = (front + 1) % size;
  return x;
}
```

Linked List Queue Data Structure




```
Void enqueue(Object x)
  if(is_empty())
    front = back = new Node(x) ;
  else
    back.next = new Node(x) back = back.next;
}

bool is_empty() { return front == null; }
```

Circular Array vs. Linked List

- Circular Array
  - Too much space
  - Kth element accessed in O(1)
  - Not as complex
  - Could make array more robust
- Linked List
  - Can keep growing
  - No going back around to front
  - More complex code

Algorithm Analysis: Why?

- Correctness:
  - Does the algorithm do what is intended.
  - How well does the algorithm complete its goal
- Performance:
  - What is the running time of the algorithm
  - How much storage does it consume
- Different algorithms may correctly solve a given task
  - Which should I use?

Iterative Algorithm for Sum

- Find the sum of the first n integers stored in an array v

Iterative Algorithm for Sum

- Find the sum of the first n integers stored in an array v

```
sum(integer array v, integer n) returns integer
   let sum = 0 for    i      =
       1...n
   sum = sum + ith    number
   return  sum
```

Programming via Recursion

- Write a recursive function to find the sum of the first n integers stored in array v

Programming via Recursion

- Write a recursive function to find the sum of the first n integers stored in array v

```
sum(integer array v, integer n) returns integer
   if n = 0 then 0
   sum    =
   else
      sum = nth number + sum of first n-1 numbers
   return sum
```

- Next Class
- Read homework
- Start Project