CS280 – Data Structures

Motivation

What to expect from CS280?

- Classical **Abstract Data Types** (ADTs): stacks, queues, trees, graphs, etc.
- Basic **algorithm analysis**: complexity representation
- Low-level **implementation** (C++)
  - Basic data structures and their variations
  - Popular graph search algorithms: shortest path
  - etc..

About Data Structures

About Data Structures

- What are data structures
  - Virtual organization of memory
  - Memory, physically is linear. Think of it as an array.
  - All we do is manipulate memory to satisfy our means.
  - Data structures allow us to access and manipulate memory in different ways.

About Data Structures

- Algorithms:
  - An ***algorithm*** is a process organized in a set of steps that mean to solve particular problems.
  - Data structures support algorithms.
    - Some are better for certain algorithms, others are worse.
    - **Priority queue** supports **Dijkstra's algorithm**
    - **Disjoint-set** and union-find algorithm supports **Kruskal's minimum spanning tree algorithm**

About Data Structures

- This course is about the study of several data structures and their use in algorithms.
  - The similarities.
  - The pros and cons of each.
  - Time and space trade-off

Data Structures

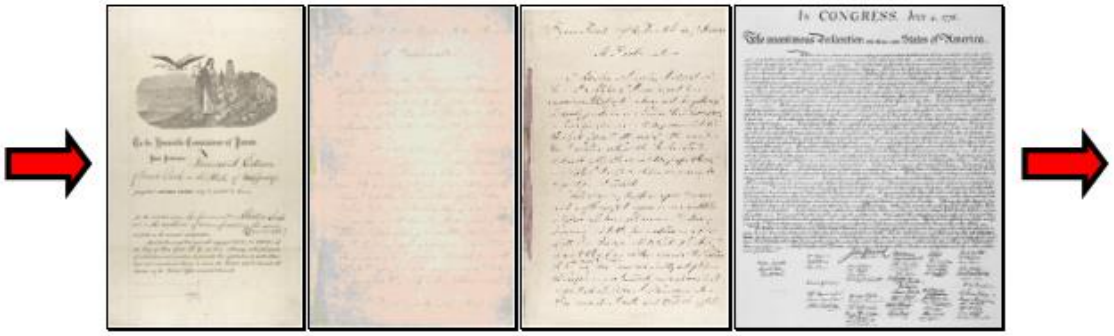
- Linked List
- Arrays
- Stacks
- Queues
- Trees (BST-Trees, AVL Trees, Splay Trees, Red Black Trees)
- Graphs
- Hash Maps
- Skip Lists
- Heaps

Data Structures

- The way in which the data is organized affects the performance of a program for different tasks.
- Computer programmers decide which data structures to use based on the nature of the data and the processes that need to be performed on that data.

Example: A Queue

- A *queue* is an example of commonly used simple data structure. A queue has **beginning** and **end**, called the *front* and *back* of the queue.

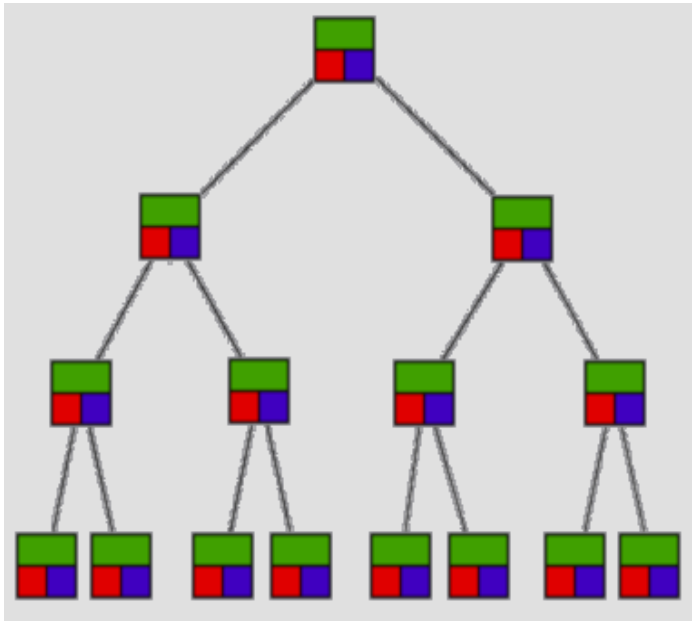


- Data enters the queue at one end and leaves at the other. Because of this, data exits the queue in the same order in which it enters the queue, like people in a checkout line at a supermarket.

Example: A Binary Tree

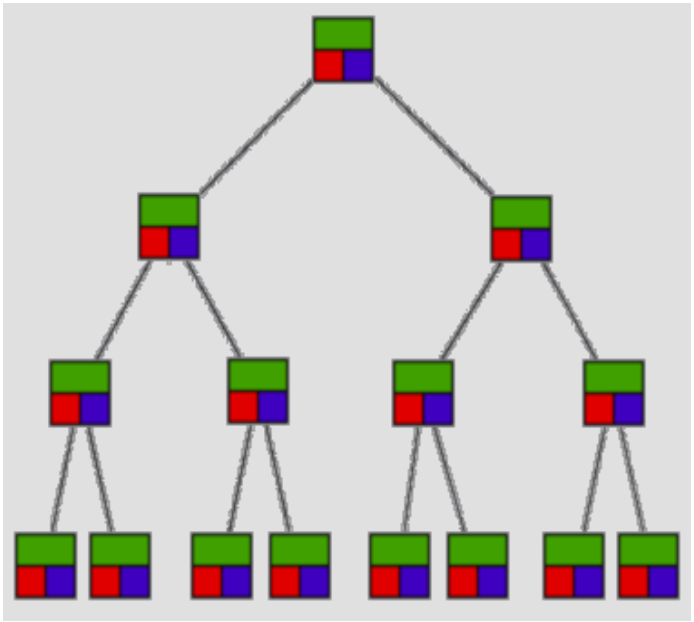
- A *binary tree* is another commonly used data structure. It is organized like an upside down tree.

- Each ***node***, holds an item of data along with a **left pointer** and a **right pointer**.



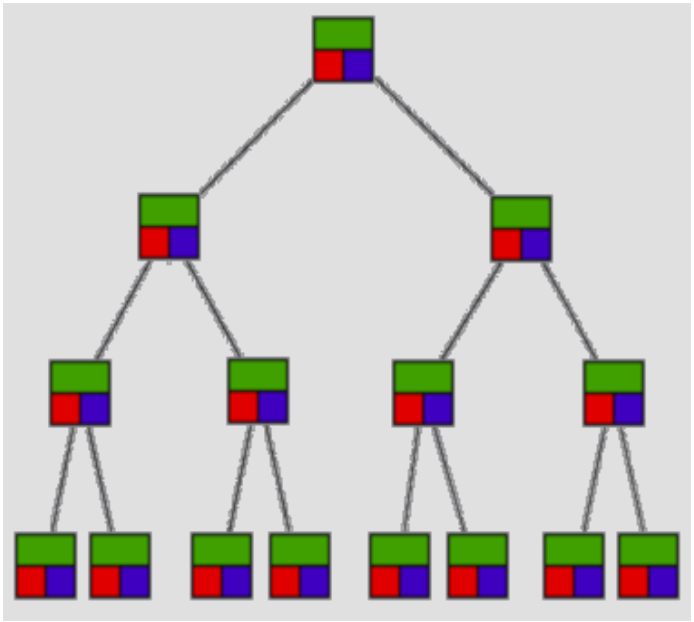
Choosing Data Structures

- By comparing the queue with the binary tree, you can see how the structure of the data affects what can be done efficiently with the data.



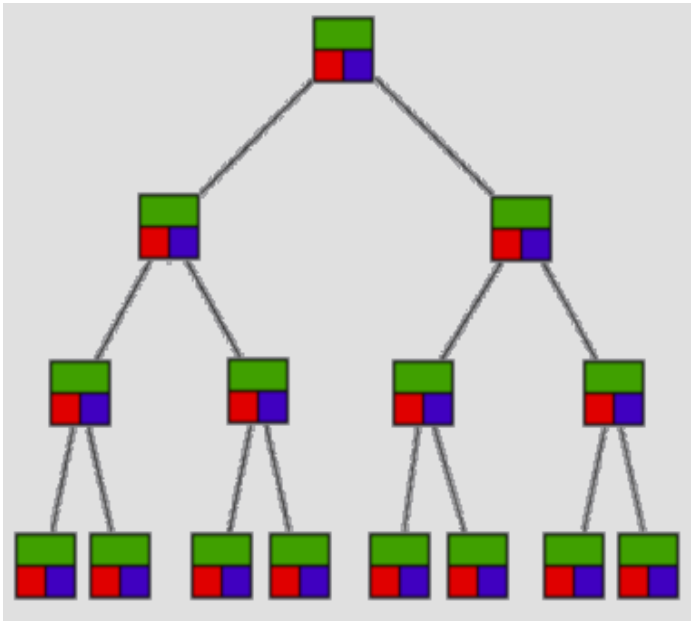
Choosing Data Structures

- A queue is a good data structure to use for storing things that need to be kept in order, such as documents waiting to be printed on a network printer.



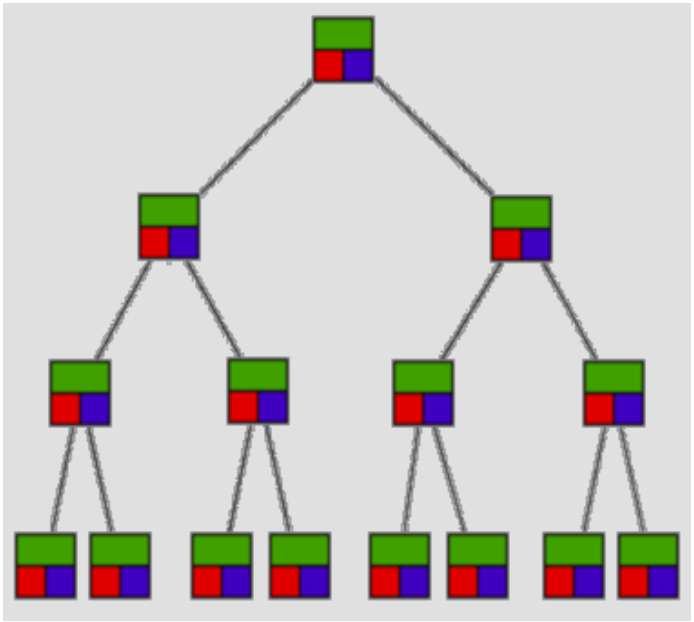
Choosing Data Structures

- The jobs will be printed in the order in which they are received.
- Most network print servers maintain such a *print queue*.



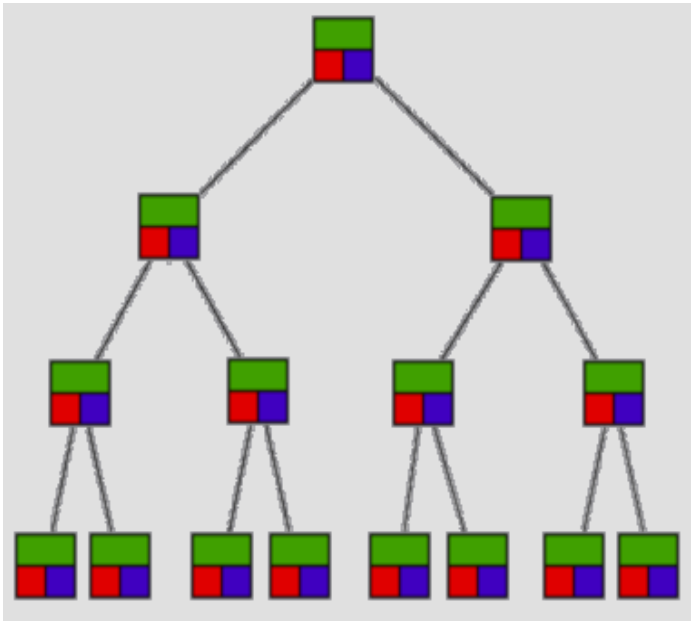
Choosing Data Structures

- A binary tree is a good data structure to use for **searching** sorted data.
- **Lesser** items to the left and **greater** items to the right.



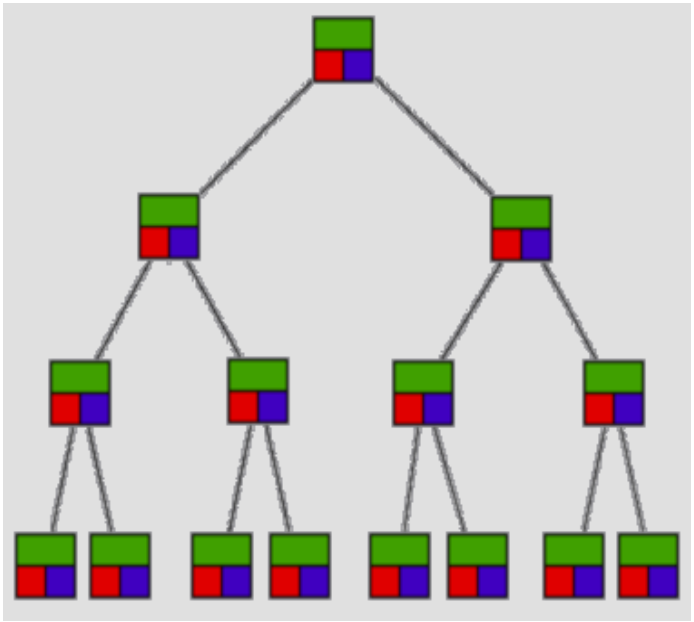
Choosing Data Structures

- A search begins at the root. It **either** find the data, **or** moves left **or** right, depending on the value for which it is searching.



Choosing Data Structures

- Items can be located very quickly in a tree*.
- Telephone directory information can be stored in a tree, so that a name and phone number can be found quickly.



Choosing Data Structures

- For some applications, a queue is the best data structure to use.
- For others, a binary tree is better.
- We choose from among many data structures based on how the data will be used by the program.

Data Structures: Another Example

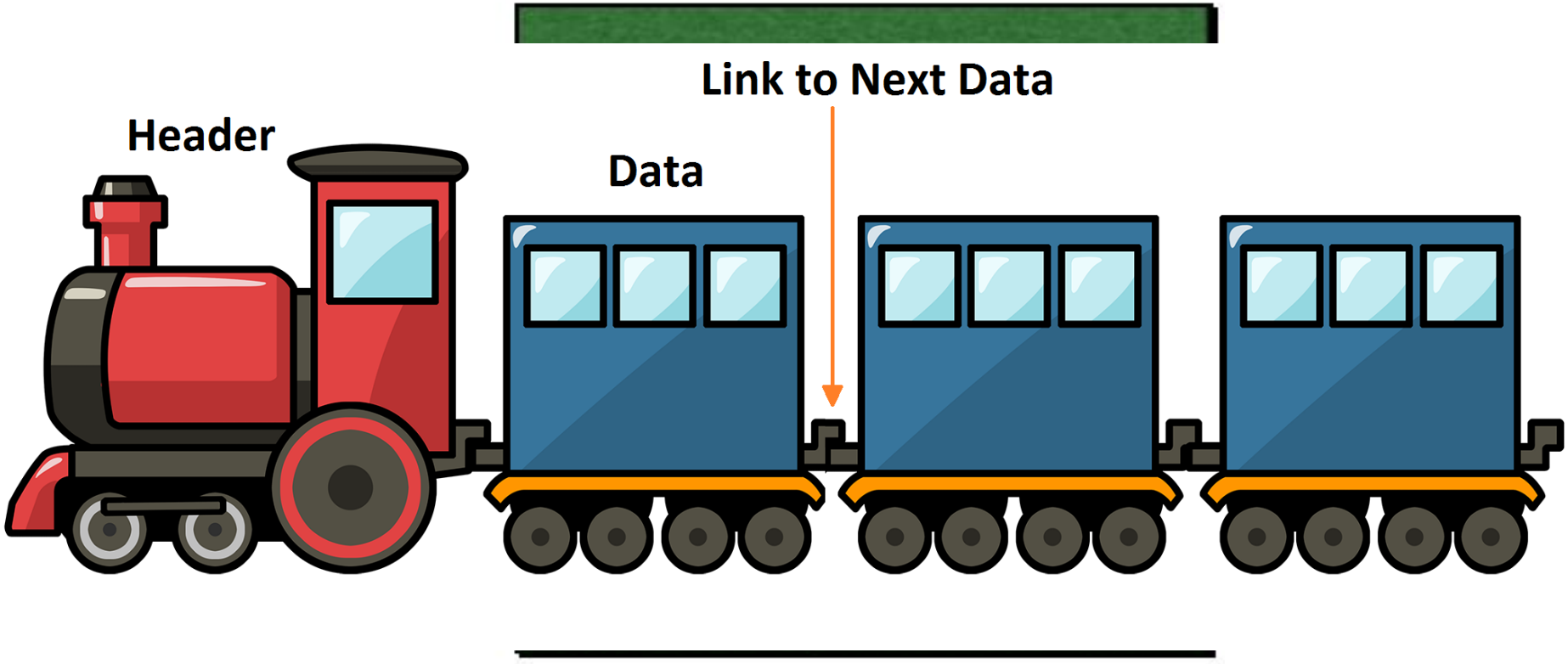
- Linked lists V.S. Arrays

Linked List

Arrays

Linked Lists

- It consists of a group of nodes which together represent a sequence.



Arrays

- It is an indexed set of variables, such as dancer[1], dancer[2], dancer[3],…
- It is like a set of boxes that hold things.

Linked List

Arrays

Linked Lists And Arrays

- You can see the difference between arrays and lists when you delete items.

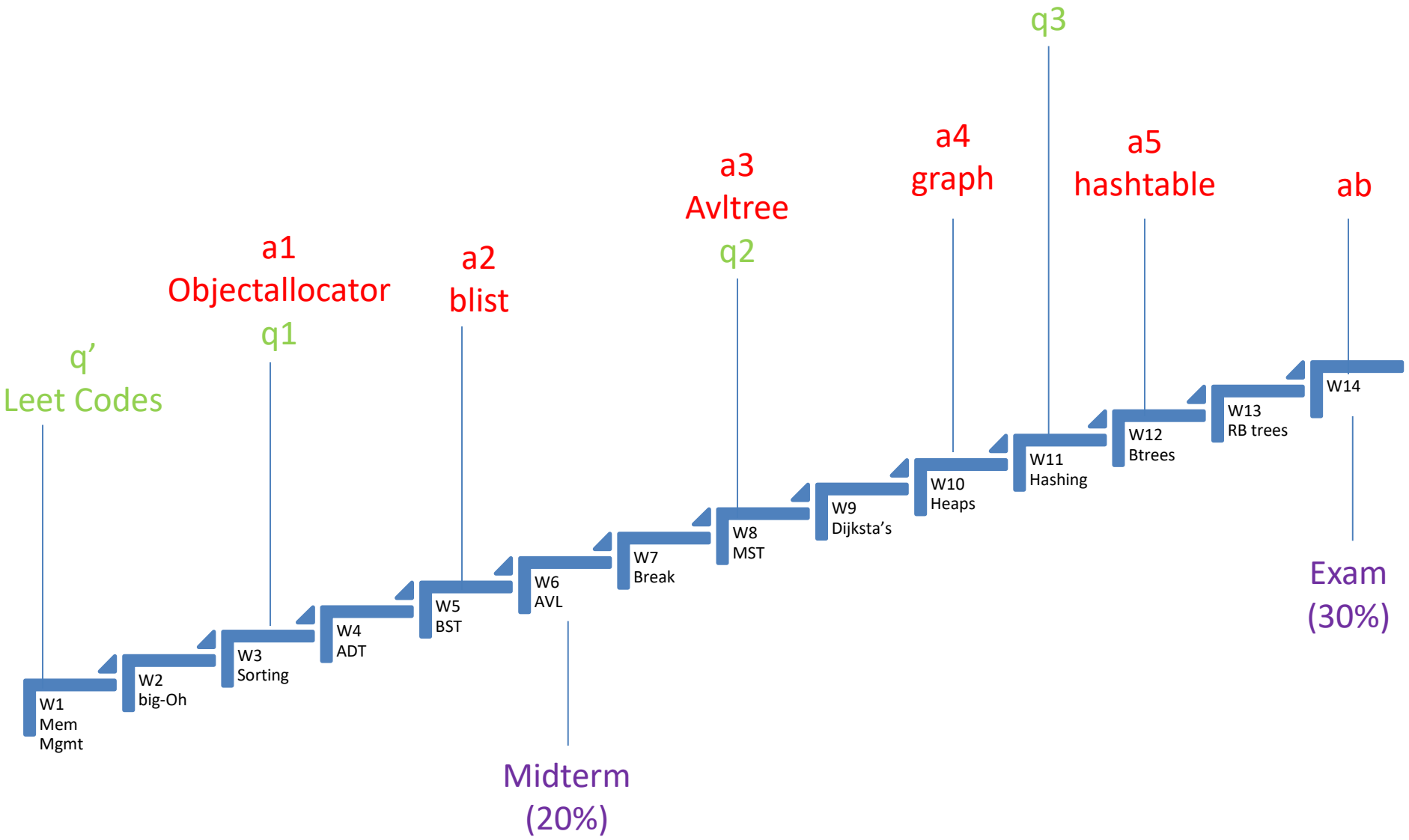
Linked Lists And Arrays

- In a linked list, the missing spot is filled in when something is deleted.

Linked Lists And Arrays

- In an array, an empty location is left behind when something is deleted.

Overview



Each assignment – 7% **+ Bonus – 3.5%** **Quizzes – 15%** **Kahoot – 1 %**

Extra Quiz

- Leet codes

- Choose 10 Problems – related to Data Structures

Slack

- https://join.slack.com/t/sit-dp-cs280s21/shared_invite/zt-kum0syls-K6VAKQvLd4mfOeVV9lMjTQ

References

- Algorithms in C++
  - Section 3.1: Elementary Data Structures – Building blocks