What is an Algorithm?

CAC 210

Amber Wagner

Today's Agenda

- Data Structures
- Algorithms
- and how they are related

What is a data structure?

Data Structures

Data structure	Description
Record	A ***record*** is the data structure that stores subitems, often called fields, with a name associated with each subitem.
Array	An ***array*** is a data structure that stores an ordered list of items, where each item is directly accessible by a positional index.
Linked list	A ***linked list*** is a data structure that stores an ordered list of items in nodes, where each node stores data and has a pointer to the next node.
Binary tree	A ***binary tree*** is a data structure in which each node stores data and has up to two children, known as a left child and a right child.
Hash table	A ***hash table*** is a data structure that stores unordered items by mapping (or hashing) each item to a location in an array.
Heap	A ***max-heap*** is a tree that maintains the simple property that a node's key is greater than or equal to the node's childrens' keys. A ***min-heap*** is a tree that maintains the simple property that a node's key is less than or equal to the node's childrens' keys.
Graph	A ***graph*** is a data structure for representing connections among items, and consists of vertices connected by edges. A ***vertex*** represents an item in a graph. An ***edge*** represents a connection between two vertices in a graph.

What is an Algorithm?

- You've already been using algorithms…but you knew that.
- "In mathematics and computer science, an algorithm is an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing and automated reasoning tasks." Wikipedia
  - Must solve a problem correctly and efficiently
- Understanding algorithms and knowing when to apply them is the key

A Little History

- Donald Knuth, a founder of CS, traced CS origins to al-Khwarizmi (9th century CE mathematician)
- Because of al-Khwarizmi, the algorithm was associated with
- positional notation
- decimal point
- zero

A Little History

- Ada Lovelace working with Charles Babbage in the 1800s: Analytical Engine
    - Lovelace predicted that such a machine could be used to compose complex music, produce graphics, and would be useful for practical and scientific purposes
- 1900s: Turing was working on the Universal Turing Machine, a thought experiment to determine what is computable
    - Church-Turing Thesis: a calculation with natural numbers (integers) is “effectively computable” (that is, given enough time and pencils, a human could do it) only if the Universal Turing Machine could do it
    - The Universal Turing Machine is a mathematical proof for universal computation; it’s an algorithm

Ubiquitous

- Algorithms are everywhere
- They impact us more than we know
- We adapt ourselves to algorithmic systems, something Tarleton Gillespie (Microsoft Research, Communications prof at Cornell) calls, “tacit-negotiation”
  - We enunciate differently when talking to machines
  - We hashtag statements to empower trends
  - We include search engine friendly descriptions

Algorithms In Action

- Traveling Salesman > Routing UPS drivers
  - ORION, 1,000 page algorithm
- Google's PageRank algorithm
- Amazon > Transformational algorithm
- Facebook > Social algorithm

Algorithms in the Future

- Stephen Wolfram (computer scientist and polymath) argues that computers will eventually be able to simulate election outcomes or future price of stocks to *any desired degree* of accuracy

- From his book, *A New Science:*

  - “The crucial idea that has allowed me to build a unified framework for the new kind of science that I describe in this book is that just as the rules for any system can be viewed as corresponding to a program, so also its behavior can be viewed as corresponding to a computation.”

How are data structures and algorithms related?

Relationship

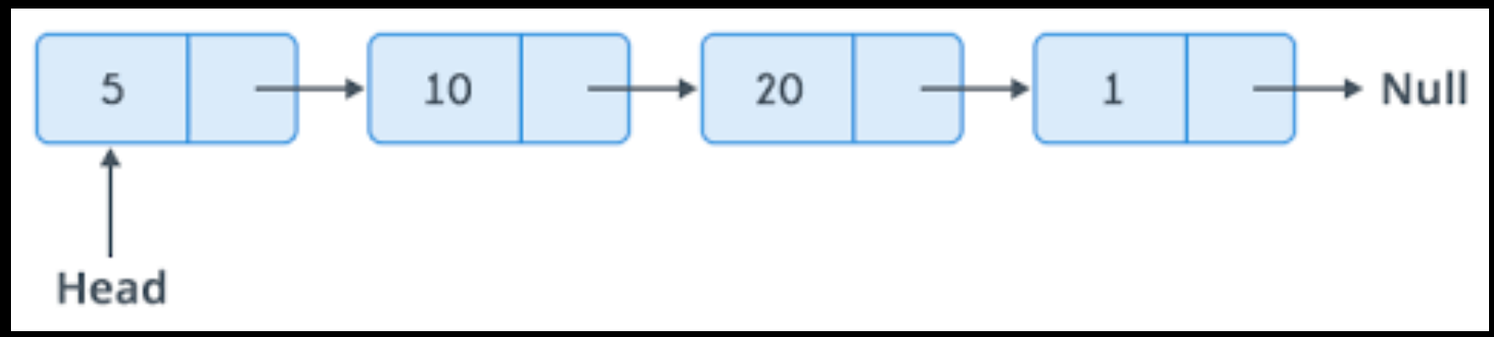
- We store data somehow and depending on what we want to do with that data
  - how often we insert new data
  - how often we retrieve data
  - how often we remove data
- Will determine what structure we want to use.
- Every structure has different algorithms that perform these tasks, and they are not all created equal so we have to choose wisely.
- Some algorithms use data structures within the algorithm itself.

Abstract Data Types

We don't want to be concerned with the details of how to add or remove elements…we just want to be able to do it.

Abstract data type	Description	Common underlying data structures
List	A ***list*** is an ADT for holding ordered data.	Array, linked list
Dynamic array	A ***dynamic array*** is an ADT for holding ordered data and allowing indexed access.	Array
Stack	A ***stack*** is an ADT in which items are only inserted on or removed from the top of a stack.	Linked list
Queue	A ***queue*** is an ADT in which items are inserted at the end of the queue and removed from the front of the queue.	Linked list
Deque	A ***deque*** (pronounced "deck" and short for double-ended queue) is an ADT in which items can be inserted and removed at both the front and back.	Linked list
Bag	A ***bag*** is an ADT for storing items in which the order does not matter and duplicate items are allowed.	Array, linked list
Set	A ***set*** is an ADT for a collection of distinct items.	Binary search tree, hash table
Priority queue	A ***priority queue*** is a queue where each item has a priority, and items with higher priority are closer to the front of the queue than items with lower priority.	Heap
Dictionary (Map)	A ***dictionary*** is an ADT that associates (or maps) keys with values.	Hash table, binary search tree

Linked List




```
Block current = head.next; //assumes you created a Block called head a
int nextX = head.x;
int nextY = head.y;
Block nextPos = head;

// Loops through the whole snake to move all of the blocks one space
while (current != null) {
  int tempX = current.x;
  int tempY = current.y;
  current.x = nextX;
  current.y = nextY;
  nextX = tempX;
  nextY = tempY;
  current = current.next;
}
```

Let’s think about adding…

- I know the current tail
- Make a new tail and place it at the current tail’s x/y
- Set the current tail’s next to point to the new tail
- Set the tail as the new tail

Let’s look at some common algorithms…starting with Linear Search

Linear Search

- Animation:

https://www.cs.usfca.edu/~galles/visualization/Search.html

- The code loops through each item in the list and compares it to the item for which we are searching
- Correct?
- Efficient?

Runtime

- An algorithm's runtime is the length of time it takes to execute
- If I have a list of 1,000,000 items, and it takes approximately one microsecond to look at an item in the list, compare it to my search item, and then iterate to the next item in the list, it would take one second to execute the algorithm.
- Do you think Google takes this approach?

Runtime

If you Google ‘algorithm’, you get these stats:

2021: About 507,000,000 results (0.69 seconds)

2020: About 400,000,000 results (0.54 seconds)

2019: 150,000,000 results (0.47 seconds)

Improvement

- How do we improve the linear search algorithm?
- How else could we search?
- How do you find a specific page in a large text?

Binary Search

- This algorithm assumes the list is sorted
- Ahhh…now we can search a little more efficiently
- How does it work?

Binary Search

- The search space is halved with each iteration.
- Is this more efficient than a linear search?
- If so, by how much?

Next Class

- If you did not already read Chapter 1 in ZyBooks, please do so.

- Read Chapter 2