This repository Search Pull requests Issues Gist

appacademy / curriculum Private Watch 34 Star 126 Fork 415

Code Issues 2 Pull requests 13 Projects 0 Wiki Pulse Graphs

Branch: master curriculum / ruby / readings / intro-algorithms-and-data-structures.md Find file Copy path

loschorts update ruby readme 3454cdc on May 16, 2016

1 contributor

126 lines (94 sloc) 4.13 KB Raw Blame History

Intro to Algorithms and Data Structures

Goals

- Know what an algorithm is.
- Know what a data structure is.
- Know what a tree is.
- Know what DFS and BFS are

This class won't focus heavily on abstract algorithms: we will not become experts in every algorithm under the sun. We are going to cover the basic algorithms that *will* prove useful in your career.

Coding up algorithms is a good way to practice the skills we've been learning. It gives you practice thinking like a programmer. Given a description of an algorithm, you should be able to translate it into Ruby code. That's our primary goal in this chapter.

Trees

Ruby provides an Array class which is a "linear" collection of elements. But there are other ways to represent collections and organize data.

Trees store data in a *hierarchy* of layers. An element, or node at each layer can have links to lower level nodes. One simple example is a file system:

- /
  - Users
    - ruggeri
      - Desktop
      - Documents
      - Downloads
    - patel
      - Desktop
      - Downloads
  - System
    - Library

The top-level node is called the root. Each node can hold a value: here the root holds `/`. The children of a node are the nodes one level deeper. The children of the `Users` node hold `ruggeri` and `patel`. The lowest level nodes (the ones with no children) are called leaves.

In general, nodes can have any number of children. In the special case of binary trees, nodes can have at most two children. These children are called the left and right children.

An array and a tree are two kinds of data structures. A data structure is a way of storing and organizing data in a computer so that it can be used efficiently. Depending on how you will use the data, different data structures may be appropriate.

Depth First Search (DFS)

Given a tree, we may wish to enumerate all the values held by nodes in the tree. For instance, we may wish to go through the files/folders of the tree and print each one.

One common way to traverse (i.e., visit all the nodes) a tree is depth first search. The nodes are numbered in the order that we visit them:

```
        1
       / \
      2   5
     /   / \
    3   6   9
   /   / \
  4   7   8
```

Each time, we try to visit the left child, if it exists and hasn't been visited yet. If it has, we try to visit the right child, if it exists and hasn't been visited yet. If all the children have been visited, then we move up one level and repeat.

Watch this animation to see the order that you want to visit nodes in the tree.

Breadth first search (BFS)

Breadth first search is an alternative to depth-first search.

```
        1
       / \
      2   3
     /   / \
    4   5   6
   /   / \
  7   8   9
```

Here we visit a node, then each of its children, then each of their children, etc. Watch this animation to see the order that you want to visit nodes in the tree.

An advantage of breadth-first search is that it considers shallower nodes before deeper ones.

Algorithm

DFS and BFS are algorithms. What's the difference between an algorithm and a method? An algorithm is an idea, an unambiguous but unrealized process that solves a problem and which potentially could be written in any language. A method is the implementation, a conversion of an algorithm into Ruby code which can then be run.

An algorithm can be coded up in any language.

References

- Wikipedia: Data structure

- Wikipedia: Algorithm