Basic Sorting Algorithms Implemented in Python

Jan 26th, 2014

This post includes Python based implementation of some of the classic basic sorting algorithms. Although Python already includes the excellent Timsort algorithm implementation, this was done more as an academic exercise to not forget the basic principles of sorting.

Setup and Driver Program

Each sorting algorithm is implemented as a Python function, which will sort the list in-place. I used the following piece of code to test all the algorithms.

```
import random

random_items = [random.randint(-50, 100) for c in range(32)]

print 'Before: ', random_items
insertion_sort(random_items)
print 'After : ', random_items
```

Bubble Sort

http://en.wikipedia.org/wiki/Bubble_sort

Bubble sort is one of the most basic sorting algorithm that is the simplest to understand. It's basic idea is to bubble up the largest(or smallest), then the 2nd largest and the the 3rd and so on to the end of the list. Each bubble up takes a full sweep through the list.

Bubble Sort in Python

```
def bubble_sort(items):
    """ Implementation of bubble sort """
    for i in range(len(items)):
        for j in range(len(items)-1-i):
            if items[j] > items[j+1]:
                items[j], items[j+1] = items[j+1], items[j]     # Swap!
```

Insertion Sort

http://en.wikipedia.org/wiki/Insertion_sort

Insertion sort works by taking elements from the unsorted list and inserting them at the right place in a new sorted list. The sorted list is empty in the beginning. Since the total number of elements in the new and old list stays the same, we can use the same list to represent the sorted and the unsorted sections.

Insertion Sort in Python

```
def insertion_sort(items):
    """ Implementation of insertion sort """
    for i in range(1, len(items)):
        j = i
        while j > 0 and items[j] < items[j-1]:
            items[j], items[j-1] = items[j-1], items[j]
            j -= 1
```

Merge Sort

http://en.wikipedia.org/wiki/Merge_sort

Merge sort works by subdividing the the list into two sub-lists, sorting them using Merge sort and then merging them back up. As the recursive call is made to subdivide each list into a sublist, they will eventually reach the size of 1, which is technically a sorted list.

Merge Sort in Python

```
def merge_sort(items):
    """ Implementation of mergesort """
    if len(items) > 1:

        mid = len(items) / 2        # Determine the midpoint and split
        left = items[0:mid]
        right = items[mid:]

        merge_sort(left)            # Sort left list in-place
        merge_sort(right)           # Sort right list in-place

        l, r = 0, 0
        for i in range(len(items)):     # Merging the left and right list

            lval = left[l] if l < len(left) else None
            rval = right[r] if r < len(right) else None

            if (lval and rval and lval < rval) or rval is None:
                items[i] = lval
                l += 1
            elif (lval and rval and lval >= rval) or lval is None:
                items[i] = rval
                r += 1
            else:
                raise Exception('Could not merge, sub arrays sizes do not match the main array')
```

Quick Sort

http://en.wikipedia.org/wiki/Quicksort

Quick sort works by first selecting a pivot element from the list. It then creates two lists, one containing elements less than the pivot and the other containing elements higher than the pivot. It then sorts the two lists and join them with the pivot in between. Just like the Merge sort, when the lists are subdivided to lists of size 1, they are considered as already sorted.

Quick Sort in Python

```
def quick_sort(items):
    """ Implementation of quick sort """
    if len(items) > 1:
        pivot_index = len(items) / 2
        smaller_items = []
        larger_items = []

        for i, val in enumerate(items):
            if i != pivot_index:
                if val < items[pivot_index]:
                    smaller_items.append(val)
                else:
                    larger_items.append(val)

        quick_sort(smaller_items)
        quick_sort(larger_items)
        items[:] = smaller_items + [items[pivot_index]] + larger_items
```

Heap Sort

http://en.wikipedia.org/wiki/Heapsort

This implementation uses the built in heap data structures in Python. To truly understand haepsort, one must implement the `heapify()` function themselves. This is certainly one obvious area of improvement in this implementation.

Heap Sort in Python

```
import heapq

def heap_sort(items):
    """ Implementation of heap sort """
    heapq.heapify(items)
    items[:] = [heapq.heappop(items) for i in range(len(items))]
```

Posted by Danish Mujeeb Jan 26th, 2014 Programming, Python

Related Posts

- **Parsing & Evaluating Reverse Polish Notation in Python**
- **Why I've Been Cheating On Java With Python**
- **How To Generate Javadoc Style Documentation For Python**
- **How To Convert MRI Scans To PNG Images**
- **The Best Way To Use a 32-Bit DLL Library In a 64-Bit Platform**

« The Best Way To Use a 32-Bit DLL Library In a 64-Bit Platform How To Convert MRI Scans To PNG Images »

Comments

Meghna • 2 months ago

Thank you for sharing these!! I always refer to these!

Bhanu Chander • 9 months ago

Thnaks for the detailed explanation :)

Ariel Salvo Caliban • a year ago

Hey thanks for sharing these!

Seth @ FBT · 2 years ago

Hey Danish, Thanks for sharing these easy tutorials on sorting algorithms for Python. We too have some interesting and similar tutorials at http://www.fireboxtraining.com...

1 · Reply · Share ›

papaKenya · a day ago

In insertion sort, I have included the following statement at the top of the function:

items = [3, 1, 8, 10, 5, 7, 2, 9, 4, 6]

I have also included the following statement at the end of the function - to run it:

if __name__ == '__main__':
insertion_sort(items):

When I run the program I get "None" as the response from Power-Shell prompt. Who can tell me what I am doing wrong or how to run the code. I am new to python.

· Reply · Share ›

Danish Mujeeb Mod → papaKenya · 18 hours ago

Hi, to make it easy for you, I've added the code to github at https://github.com/danishm/pyt.... You will need to install Python in Windows and run it from the prompt as "python sorting.py"

· Reply · Share ›

Max Marchuk · 3 days ago

Great post, thank you for the helpful code :)
Might I suggest something?

Your "Setup and Driver Program" uses print statements that are deprecated in Python 3, so I would suggest using print with parentheses. e.g. "print('Before: ', random_items)" instead of "print 'Before: ', random_items"

· Reply · Share ›

Matti Rintala · 6 months ago

The merge sort does not work if the data contains 0s. The test "lval and rval..." fails not only if lval or rval is None, but also if one of the is zero. ;-)

· Reply · Share ›

Marcell · 8 months ago

Hi!

In merge sort, the midpoint determination is wrong when the length of the items is odd.

for example: len(items) == 5

mid = len(items) / 2 # Determine the midpoint and split
left = items[0:mid]
right = items[mid:]

items[0:mid] will throw exception for 2.5 index.

· Reply · Share ›

Brian Fang → Marcell · 6 months ago

force it to an int, or use floor or ceil

• Reply • Share ›

Recent Posts

- How To See Free Movies & TV Shows From Google Play Store
- Parsing & Evaluating Reverse Polish Notation in Python
- How To Clean Install OS X Yosemite The Easy Way
- Amazon's Fire Phone Can Fix Face Unlock
- How To Never Loose Your Files Again

Related Posts

- Parsing & Evaluating Reverse Polish Notation in Python
- Why I've Been Cheating On Java With Python
- How To Generate Javadoc Style Documentation For Python
- How To Convert MRI Scans To PNG Images
- The Best Way To Use a 32-Bit DLL Library In a 64-Bit Platform

GitHub Repos

- python-sorting-algos

  Basic sorting algorithms implemented in Python

- dicom-to-png

  A simple python module to make it easy to batch convert DICOM files to PNG images.

- oneapp

- python-appengine-mdetect

  An example of how to use the mdetect library with webapp2 on Google App Engine to detect mobile device capabilities

- python-appengine-template

  A basic template Python project which has the most basic setup needed for a Google AppEngine application

- taskflow

  An online game to get the correct ordering of a list of steps to do a task

- easymagick

  A module to simplifying scripting some batch processing of my photographs using ImageMagick.

- intelliparse

  A Python library to help parse product properties from simple English text

- intelliutils

  A set of useful Python utility functions and classes

- getratings

  Node.js based service to extract user ratings from online shopping sites

@danishm on GitHub