Rosetta Mind Documentation

Release

introom

December 07, 2014

Contents

This document writes something about Python, something about common algorithms and data structures.

CHAPTER 1

TOC

1.1 Life

1.2 Programming

1.2.1 Basics

Searching/Sorting

Sorting

Heap Sort HeapSort is easy to comprehend given the understanding of the data structure binary heap.

```
void SiftDown(int *arr, int start, int end) {
    // for a zero-based array, suppose current pos is i
    // iParent = floor((i-1) / 2)  iLeftChild = 2*i + 1  iRightChild = 2*i + 2
    int root{start};
    while (root*2+1 < end) { // while the root has at least one child
        int lchild{root*2+1}; // the pos of the left child
        int maxPos{root}; // maxPos to be set to the largest element, so as to maintain heap-property
        if (arr[maxPos] < arr[lchild])
            maxPos = lchild;
        if (lchild+1 < end && arr[maxPos] < arr[lchild+1])
            maxPos = lchild+1; // the pos of the right child
        if (maxPos != root) {
            swap(arr[maxPos], arr[root]);
            // NB this iteratively sifting part could be best illustrated with a binary tree
            root = maxPos; // repeat to continue sifting down the child
        } else
            return;
    }
}

void Heapify(int *arr, int n) { // we are building a max-heap
    // the last element in a 0-based array is at index count-1; find the parent of that element
    int start{(n-2) >> 1};
    while (start > -1) {
        SiftDown(arr, start, n);
        start--;
    }
```

```
}

void HeapSort(int *arr, int n) {
    Heapify(arr, n);
    int epos{n-1};
    while (epos > 0) {
        swap(arr[0], arr[epos]);
        SiftDown(arr, 0, epos); // do not pass n here
        --epos; // NB the heap size is decremented by 1.
    }
}
```