Algorithm Lab

# Week 1: Sort

Description:

In computer science, sorting algorithm is an algorithm that puts elements of a list in a certain order (From Wikipedia). Thus, we can define sorting algorithms as followed:

*Instance: a list of elements* 𝐴 = (𝑎1, 𝑎2, … , 𝑎𝑛) *and a comparator*

𝐶(𝑣0, 𝑣1) → {𝑇𝑟𝑢𝑒, 𝐹𝑎𝑙𝑠𝑒}

*Result: a list* 𝐵 = (𝑏1, 𝑏2, … , 𝑏𝑛) *that* {𝑎1, 𝑎2, … , 𝑎𝑛} = {𝑏1, 𝑏2, … , 𝑏𝑛} *and for all* 1 < 𝑖 ≤ 𝑛*,* 𝐶(𝑏𝑖−1, 𝑏𝑖) *is* 𝑇𝑟𝑢𝑒*.*

If we want to sorting a list of integers by increasing order, our comparator, we should use ≤

as our comparator. Please design an algorithm to sorting integers in increasing order.

# Algorithm Design

Insertion sort will divide data to 2 partitions, sorted and unsorted.

1. Let first element as sorted part and the others as unsorted part.
2. Insert an unsorted element 𝑣𝑖 to sorted part and keep them in certain order.

2.1 Find minimum 𝑗 that 𝐶(𝑣𝑗, 𝑣𝑖) = 𝐹𝑎𝑙𝑠𝑒.

* 1. Shifting all 𝑣𝑘 that 𝑘 ≥ 𝑗.
  2. Insert 𝑣𝑖 to current position.

1. Repeat step 2 until unsorted part contains no elements.

# Implementation

Language: C

int\* sort (int\* A, int n)

{

// index i divide A[] to 2 partitions

// A[0…i-1] are sorted, A[i…n-1] are unsorted for (int i = 1; i < n; ++i)

{

int j = i, val = A[j];

while (j > 0 && !(A[j-1] <= val))

{

A[j] = A[j-1];

--j;

}

A[j] = val;

}

return A;

}

# Analysis

Space complexity

Assume that instance is an 𝑛 elements array.

* Instance: 𝑛 values and 1 index (A, n)
* Divider: 1 index (i)
* Ordering maintain: 1 index (j) and 1 value(val)

Totally needs 𝑛 + 1 values and 3 indices, so the space complexity is 𝑂(𝑛)

Time complexity

For N elements, there are N-1 steps

Best case: O(n) – When elements are already sorted. There will be N-1 comparisons

Average case: O(n2) ­– On average, half of the elements are sorted, and on average,

the element to be sorted has to be moved to the middle of the sorted part. N(N-1)\*1/2\*1/2

Worst case: O(n2) – Elements are sorted in reverse order. In each step, the element to be sorted needs to be moved completely to the left.

1 + 2 + … + N-2 + N-1 = (N-1)(N-1)/2