Sorting Algorithms

Intro and Concepts

- What does “sort” mean?
- “Non-decreasing” and “Increasing”
- “Non-increasing” and “Decreasing”
- Greedy algorithms

Bubble Sort

Concept

- Around n^2 operations
- Comparing each two adjacent elements
- Can be optimized to $O(n)$ on special cases
- Best case: sorted array (you don't say!)
- Worst case: reverse-sorted array
- On the i^{th} iteration, the last i elements are in their correct places
- Can check if array is sorted on each iteration

Illustration

Complexity

- On the i^{th} time, last i elements are in their correct position
- On the i^{th} time, we do $\mathrm{n} - i - 1$ comparisons
- $n - 1 + n - 2 + n - 3 + \ldots + 3 + 2 + 1 = \frac{n(n-1)}{2}$
- Fun fact: $1 + 2 + 3 + \ldots + n - 2 + n - 1 + n = \frac{n(n+1)}{2}$
- This translates to $O(n^2)$

Implementation

Insertion Sort

Concept

- Also around n^2 operations
- Dividing the array into two sections: sorted and unsorted
- Best case: sorted array (you don't say!)
- Worst case: reverse-sorted array
- Good example: **12**, 11, 13, 5, 6
- Can't optimize to $O(n)$

Illustration

Complexity

- In the reverse sorted array, the current element always become the first element
- Which means on the i^{th} iteration, we make i swaps
- This translates, like Bubble Sort, to $O(n^2)$

Implementation

Selection Sort

Concept

- Around n^2 operations as well
- Finding the minimal element on each iteration
- On the i^{th} iteration, the first i elements are in their correct places
- Can't be optimized to $O(n)$
- It considers all cases to be the same
- Good example: 64, 25, 12, 22, 11

Illustration

Complexity

- On the i^{th} iteration, we check $n - i - 1$ elements to find the current minimal element
- This is exactly the same as bubble sort, with complexity $O(n^2)$

Implementation

So Which is the Best?

Bubble Sort

Insertion Sort

Selection Sort

Are They All the Same?

So You're Telling Me Sort is Always $O(n^2)$?

C++ Sort

- C++ sort uses a number of optimization to reach the complexity of $O(nlogn)$
- What is log?
- What library? We're too old for that! `<bits/stdc++.h>` - `<algorithm>`
- Sort array: `sort(a + index1, a + index2)` [index1, index2[
- Most common way: `sort(a, a + n)`
- Sort vectors: `sort(v.begin() + index1, v.begin() + index2)` [index1, index2[
- Common: `sort(v.begin(), v.end())`

Count Sort

- An interesting sort algorithm that can get a complexity of $O(n)$!
- If elements are in a small range, it gets the best results
- Has a complexity of $O(n + k)$ where k is the range

Illustration + Implementation

What If Elements are Negative?

That was the Fun Part

Codeforces Problems

CF

- http://codeforces.com/problemset/problem/992/A
- http://codeforces.com/problemset/problem/994/B
- http://codeforces.com/problemset/problem/572/B
- http://codeforces.com/problemset/problem/632/C
- http://codeforces.com/problemset/problem/545/D
- http://codeforces.com/problemset/problem/922/D

Hope You Learned Something New!