Sorting Magic

Yhaliff Said Barraza

May 23, 2019

1 Sorting

Have you ever needed to order/sort something *your life, time, fiends, numbers?* well that boring as fuck, fortunately for you computers can do that for you with the magic of **sorting algorithms!!!!**

1.1 Bubble sort

1.1.1 Problem definition

You what to sort stuff Numbers in this case

1.1.2 Input and output

Example :arr = 1|5|2|3|8|4 this is your **Input** after will sort it, you will receive **This** arr = 1|2|3|4|5|8 which is your **Output**

Now let's start simple width the most basic form a sorting know to **MAN KINDDD** BUBBLE SORT
The way bubble sort work is the following manner

1.1.3 Explanation

1. Start at the beginning of the array

2. check if the values at the current index is bigger that the one at the next index

3. if true than Swap the values

4. move to the next index

5. repeat until the current index is the last index

6. keep repeating all previous steps, until the number of representations is equal to the amount of element in the array OR the array is sorted.

If that hard to understand then lets look at some pseudo code

1.1.4   pseudo code

Note: "N" mean input which is an array in this case and x means index

---
Algorithm 1 Bubble sort

1: **for** each number in N **do**
2:     **for** each number in N **do**
3:         **if** $N[x] > N[x+1]$ **then**
4:             Swap(N[x],N[x + 1])
5:         **end if**
6:     **end for**
7:     comment if no Swap occurred then we finish sorting
8: **end for**
9: *Return N* // Sorted
---

simple words   To put in simple words what bubble sort does, it move the largest elements (numbers) to the end of the array, in the process moving the smaller element to the beginning of the same array **Now Time to put this in code**   Then do a benchmark

1.1.5   Code

Here is the C++ implementation

```cpp
void BubbleSort(std::vector<int> &Vec)
{
  // to not have to sort an necessarily
  bool isSorted = true;

    for (auto Elemento : Vec)
    {
      for (int j = 0; j < Vec.size() - 1; ++j)
      {
        if (Vec[j] > Vec[j + 1]) {
```

```
            Swap(Vec[j], Vec[j + 1]);
            isSorted = false;
        }
    }

    if (isSorted) { break; }

    isSorted = true;
    }
}
```
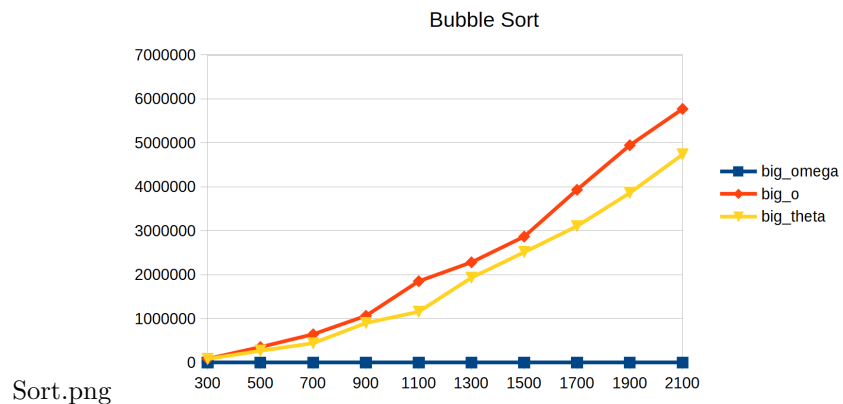
1.1.6 Benchmark

TIME FOR BENCHMAKING
Here is the resulting of bench-marking Bubble sort, should also mention that the y-axis represents *Microseconds* and the x-axis represent the amount of elements in the array

Sort.png

1.2 Insertion sort

Now it's time to talk about another algorithm, one that likes going **DEEP** the man that go his own way (backwards) muh boi **INSERTION SORT!!!!**

1.2.1 Problem definition

It's still the same problem that i mention in 1.1.1

1.2.2 Input and output

the same that i said in 1.1.2 sill applies here

1.2.3   Explanation

The way that Insertion sort work is very particular because it start's at (almost) the end of an array , and works it's way to the end it would be best explained with a step by step guide of insertion sort
something

1. Start at the pen-ultimate index of the array(will call this N[1])

2. Check if the item in the current index is bigger than it's neighbor if true then do step 3 else go to step 4

3. swap the values and move forward then repeat step 2 until your reach the last index in the array

4. Now start the entire process at N[1] - (the amount of times you reached this step) and repeat step 2 and 3 until you've go through the all the array

note   now after explain the process of insertion sort, we can use this knowledge convert the previous steps into code.

1.2.4   Code

```cpp
void InsertionSort(std::vector<int> &Vec)
{
  for (int i = Vec.size() - 1; i > 0; --i) {

    if (Vec[i] < Vec[i - 1])
    {
      // this is so we don't go out of bounds
      int CurrentPos = i;
      // making sure we don't go out of bonds
      while (CurrentPos <= (Vec.size() - 1) && Vec[CurrentPos] < Vec[CurrentPos
      {
        Swap(Vec[CurrentPos - 1], Vec[CurrentPos]);
        CurrentPos++;
      }
    }
  }
}
```

and teh amunt of fast   now it's time to answer another question you probably have in your head , ? how fast is this thing ? will the simple answer is **BENCH MARKING** and it depends

1.2.5 BENCH MARKING

Here's the result's of bench making and what i said in 1.1.6 is still true here.

Sort.png