Bubble Trouble

A statistical analysis of the Bubble Sort.
The Guys Who Bring It To You

- Jon Kroening
- Nick Jones
- Erik Weyers
- Andy Schieber
- Sean Porter
History and Background

Jon Kroening
The Bubble Sort

- Popular algorithm used for sorting data
- Iverson was the first to use name ‘bubble sort’ in 1962, even though used earlier
- Unfortunately it is commonly used where the number of elements is too large
The Bubble Sort

- Starts at one end of the array and make repeated scans through the list comparing successive pairs of elements.

- If the first element is larger than the second, called an inversion, then the values are swapped.

5 3 6 2 8 9 1

3 5 6 2 8 9 1
The Bubble Sort

Each scan will push the maximum element to the top

3 5 6 2 8 9 1
3 5 6 2 8 9 1
3 5 6 2 8 9 1
3 5 2 6 8 9 1
3 5 2 6 8 9 1
3 5 2 6 8 9 1
3 5 2 6 8 9 1
3 5 2 6 8 9 1
3 5 2 6 8 9 1
The Bubble Sort

- This is the “bubbling” effect that gives the bubble sort its name
- This process is continued until the list is sorted
- The more inversions in the list, the longer it takes to sort
Bubble Sort Algorithm

template <typename R>
void bubbleSort(vector<R>& v)
{
    int pass, length;
    R temp;
    for( length = 0; length < v.size()-1; length++)
    {
        for(pass = 0; pass < v.size()-1; pass ++)
        {
            if(v[pass] > v[pass +1])
            {
                temp = v[pass];
                v[pass] = v[pass + 1];
                v[pass + 1] = temp;
            }
        }
    }
}
Best and Worst Case Scenarios

Nick Jones
Best Case of the Bubble Sort

Let $X$: number of interchanges (discrete).

Consider list of $n$ elements already sorted

$$x_1 < x_2 < x_3 < \ldots < x_n$$
Best Case (cont.)

- Only have to run through the list once since it is already in order, thus giving you \( n-1 \) comparisons and \( X = 0 \).
- Therefore, the time complexity of the best case scenario is \( O(n) \).
Worst Case of the Bubble Sort

Let $X$: number of interchanges (discrete).

Consider list of $n$ elements in descending order
\[ x_1 > x_2 > x_3 > \ldots > x_n. \]
Worst Case (cont.)

- # of comparisons = # of interchanges for each pass.
- $X = (n-1) + (n-2) + (n-3) + \ldots + 2 + 1$
- This is an arithmetic series.
Worst Case (cont.)

\[(n-1) + (n-2) + \ldots + 2 + 1 = n(n-1)/2\]

So \(X = n(n-1)/2\).

Therefore, the time complexity of the worst case scenario is \(O(n^2)\).
Time Complexity Model of the Worst Case

![Graph showing time complexity model]
Average Case

Nick Jones, Andy Schieber & Erik Weyers
Random Variables

- A random variable is called discrete if its range is finite or countably infinite.
- Bernoulli Random Variable – type of discrete random variable with a probability mass function
  \[ p(x) = P\{X = x\} \]
Bernoulli Random Variable

\( X \) is a Bernoulli Random Variable with probability \( p \) if

\[
p_x(x) = p^x(1-p)^{1-x}, \ x = \{0,1\}.
\]

Other notes:

\( E[X] = p \)
Expected Value

The expected value of a discrete random variable is
Sums of Random Variables

If are random variables then

i.e. expectation is linear
Analytic Estimates for Bubble Sort

- $I = \text{number of inversions}$
- $X = \text{number of interchanges}$
- Every inversion needs to be interchanged in order to sort the list, therefore $X$ must be at least $I$,
- $E[X] = \text{The average number of inversions}$
The probability that \(i, j\) are inverted is \(.5\).

\(I\) is a Bernoulli random variable, therefore

there are

terms in the above sum
Since \( p(I) = 0.5 \) and there are terms in the above sum

then
So

The average case is equivalent to the worst case
Simulation Results

Jon Kroening, Sean Porter, and Erik Weyers
int MAX = 99;
int Random[MAX];
int temp, flag;

for(int i = 0; i<100; i++)
{
    flag = 1;                        // Random Number
    temp = 1+(int) (100.0*rand())/(RAND_MAX+1.0));  //obtained
    for(int check = 0; check <= i; check++){
        if(Random[check] == temp)   //Checks if number is
            flag = 0;        // already obtained
    }
    if(flag == 0)                 //if number is already obtained
        i--;
    else{                        //then it will reloop to find new one
        Random[i] = temp;       //Otherwise enters number
        cout << Random[i] << endl; //in array and displays
    }
}
Simulation Results

BUBBLE SORT ANALYSIS

INVERSIONS

DATASETS

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33
Worst Case:
Closing Comments

The Bubble sort is not efficient, there are other sorting algorithms that are much faster
A Final Note

Thank you Dr. Deckleman for making this all possible
References