Comparison & non-comparison sort

Sorting algos can be classified into two main categories:
1) Comparison sorts
2) Non-comparison sorts

The most know comparison sorts:
1) bubble sort
2) insertion sort
3) selection sort
4) quicksort
5) mergesort

* The main characteristic of comparison sort is that they MUST compare pairs of elements to determine the sorted order.

Non-comparison sorts DO NOT make comparisons.
Examples are:
1) Counting sort
2) Radix sort
3) Bucket sort

Why are we studying different sorting types?

* Because comparison sorts have a limit on their WORST CASE RUNNING TIME.

- They cannot be faster than N LOG N. This is not the case for non-comparison sorts.

Comparison Sort	Worst case running time	Best case running time
Bubble	$\Theta(N^2)$	$\Theta(N)$
Insertion	$\Theta(N^2)$	$\Theta(N)$
Selection	$\Theta(N^2)$	$\Theta(N^2)$
Quicksort	$\Theta(N^2)$	$\Theta(N \lg N)$
Mergesort	$\Theta(N \lg N)$	$\Theta(N \lg N)$

↪ most asymptotically optimal for large values of N.

Sorting algos that DO NOT make comparisons can beat the lower bound of N LOG N comparison sorts.

Bubble sort

```
function BubbleSort(Arr, N)      <- size of array
    Swapped = true
    while(swapped) do
        Swapped = false
        for 0 ≤ i < N-1 do
            if (Arr[i] > Arr[i+1]) then
                swap(Arr[i], Arr[i+1])
                Swapped = true
            end if
        end for
        N = N-1
    end while
    return Arr
end function
```

0	1	2	3
4	3	2	1
3	2	1	4
2	1	3	

pass	Swapped	i	(Arr[i] > Arr[i+1])	N-1	Swap
1	true	0	4 > 3 = true	3	true
		1	4 > 2 = true		true
		2	4 > 1 = true		true
		3	— —	2	—
2	false	0	3 > 2 = true	2	true
		1	3 > 1 = true		true
		2	— — —	1	
3	false	0	2 > 1 = true	1	true
		1	— —		— . . .
4	false	—	—	—	—

Bubble sort time complexity

Best Case: we have a sorted array

[1, 2, 3]

```
function BubbleSort(A,N)
    swapped=true
    while (swapped) do
        swapped=false
        for 0 <= i < N-1 do
            if (A[i] >A[i+1]) then
                swap(A[i],A[i+1])
                swapped=true
            end if
        end for
        N=N-1
    end while
    return A
end function
```

$T(N)$

C_0

C_1

C_2

$C_3 N$

$C_4(N-1)$

won't execute in best case

C_5

C_6

$T(N) =$

$C_0 + C_1 + C_2 + C_3 N + C_4(N-1) + C_5 + C_6$

$T(N) = C_7 + C_3 N + C_4(N-1)$

$= C_7 + C_3 N + C_4 N - C_4$

$= \underline{C_8 + C_9 N}$

$T(N)$ is $O(N), O(N^2), \ldots$

is $\Omega(N), \Omega(\log N), \ldots$

is $\Theta(N)$

Worst Case

```
function BubbleSort(A,N)
    swapped=true
    while (swapped) do
        swapped=false
        for 0 <= i < N-1 do
            if (A[i] >A[i+1]) then
                swap(A[i],A[i+1])
                swapped=true
            end if
        end for
        N=N-1
    end while
    return A
end function
```

N times

$T(N)$

C_1

$C_2(N+1)$

$C_3 N$

$C_4 N(N-1)$

$C_5 \frac{N(N-1)}{2}$

$\sum_{k=1}^{N} (N-k) = \frac{N(N-1)}{2} \cdot C_5$

$C_6 N$

C_7

$$T(N) = \cancel{C_1} + C_2(N+1) + C_3N + C_4N(N-1) + C_5\left(\frac{N(N-1)}{2}\right) + C_6N + \cancel{C_7}$$

$$= \cancel{C_8} + C_2N + \cancel{C_2} + C_3N + C_4N^2 - \cancel{C_4} + C_5 \cdot \frac{N^2-N}{2} + C_6N$$

$$= C_9 + \left(C_4 + \frac{C_5}{2}\right)N^2 + \left(C_2 - \frac{C_5}{2} + C_6\right)N$$

$$T(N) = c_9 + \left(c_4 + \frac{c_5}{2}\right)N^2 + \left(c_2 - \frac{c_5}{2} + c_6\right)N$$

$T(N)$ is $O(N^2)$, $O(N^3)$, ...

is $\Omega(N^2)$, $\Omega(N)$, ...

is $\Theta(N^2)$

Insertion Sort

Best case running time

```
function InsertionSort(A,N)
    for 1 <= j <= N-1 do
        ins=A[j]
        i=j-1
        while (i>=0 and ins<A[i])
            A[i+1]=A[i]
             i=i-1
        end while
        A[i+1]=ins
        end for
end function
```

$T(N)$

$C_1 N$

C_2

C_3

$C_4(N-1)$

Not executed in bestcase

$C_5(N-1)$

0	1	2	3
1	2	3	4

$i \geq 0$ AND $A[j] < A[i]$

pass	j	i	ins	(i>=0 && ins < A[i])
1	1	0	2	false
2	2	1	3	false
3	3	2	4	false
4	4	-	-	-

$$T(N) = C_1 N + C_2 + C_3 + C_4(N-1) + C_5(N-1)$$

$$= C_5 + C_1 N + C_4 N - C_4 + C_5 N - C_5$$

$$= (C_1 + C_4 + C_5)N + C_6$$

$$= C_7 N + C_6$$

$T(N)$ is $O(N)$, $O(N^2)$, ...

is $\Omega(N)$, $\Omega(\log N)$, ...

is $\Theta(N)$ $T(N)$ behaves linearly.

Time Complexity
Worst case

```
function InsertionSort(A,N)
    for 1 <= j <= N-1 do
        ins=A[j]
        i=j-1
        while (i>=0 and ins<A[i])
            A[i+1]=A[i]
            i=i-1
        end while
        A[i+1]=ins
    end for
end function
```

$\left.\begin{array}{l}\texttt{ins=A[j]}\\ \texttt{i=j-1}\end{array}\right\}$ select an element you want to insert

$\left.\begin{array}{l}\texttt{A[i+1]=A[i]}\\ \texttt{i=i-1}\end{array}\right\}$ shifting elements until we find a suitable place to insert our element

↳ invariant is everything left of j is sorted

$T(N)$

$C_1(N-1)$

$C_2(N-2)$

$C_3(N-2)$

$C_4(N-2)(N\cdot$

comes from for loop

comes from the while loop executing i times

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ \end{array}$$
[4, 3, 2, 1]

pass	j	ins	i	i>=0 && ins<A[i]
1	1	3	0	✓ True 3 < 4
2	2	2	1	✓ True 2 < 4
3	3	1	2	✓ True 1 < 4

pass 1 [4, 4, 2, 1]

→ i = 0 − 1 = −1

[3, 4, 2, 1]

pass 2 [3, 4, 4, 1]

i = 0 && 2 < 4

[3, 3, 4, 1]

i = −1

→ [2, 3, 4, 1]

pass 3 [2, 3, 4, 4]

i = 1

i>=0 && 1 < 3

True

A[1+1] = A[1]

A[2] = A[1]

[2, 3, 3, 4]

Needs more work

0 1 2

Worst case time complexity

```
1  function InsertionSort(A,N)             T(N)
2      for 1 <= j <= N-1 do                C_0(N) ← mult
3          ins=A[j]                        C_1(N-1)
4          i=j-1                           C_2(N-1)
5          while (i>=0 and ins<A[i])       C_3(N+1)
6              A[i+1]=A[i]                 C_4(N)
7              i=i-1                       C_5(N)
8          end while
9          A[i+1]=ins                      C_6(N-1)
10     end for
11 end function
```

$$\sum_{i=1}^{k}(i) = \frac{n(n+1)}{2} - 1$$

$$5*5 = \frac{25+5}{2} - 1$$

$$\frac{n(n+1)}{2} - 1$$

Given an array in reverse order [3, 2, 1]

pass	j	ins	i	(i >= 0 && ins < A[i])	A[i+1] = A[i]	i = i - 1	Array
1	1	2	0	0>=0 && 2 <3 (True)	A[1]=3	-1	[3, 3, 1]
line 9:		since i = -1 we update A[i+1] ==> A[0] = ins ==> A[0]== 2					[2, 3, 1]
2	2	1	1	1>=0 && 1<3 (True)	A[2]=3	0	[2, 3, 3]
			0	0>=0 && 1 < 2 (True)	A[1]=2	-1	[2, 2, 3]
			-1	-1>=0 (False)			
line 9:		since i = -1 we update A[i+1] ==> A[0] = 1					[1, 2, 3]
3	3	j is not <= N -1 anymore so we exit the for loop and end the function execution					

Execution of while loop:

1. 4, 3, 2, 1 0
2. 0, 4, 3, 2, 1
3. 0, 1, 4, 3, 2
4. 0, 1, 2, 4, 3

5 + 4 + 3 + 2 = [14]

ins — 1 false case

0 = 4 + 1 = 5

1 = 3 + 1 = 4

2 = 2 + 1 = 3

3 = 1 + 1 = 2.

```
function SelectionSort(A,N)
   for 0 ≤ i < N-1 do
      min=pos_min(A,i,N-1)
      swap(A[i],A[min])
   end for
end function
```

$C_1(N)$

$C_2\left(\frac{n(n+1)}{2}-1\right)$

$C_3(n-1)$

j=0 0, 1, 2, 4 5 = 5+4+3+2

j = 1

```
// Find the minimum element in unsorted array
int min_idx = i;
for (int j = i+1; j < n; j++)
    if (numArray[j] < numArray[min_idx])
        min_idx = j;
   Return min_idx
```

C_1

$C_2 N$

$C_3(n-1)$

$C_4(n-1)$

C_5

n = 5, 5+4+3+2 = 14

n = 6, 6+5+4+3+2 = 20

n = 7, 7+6+5+4+3+2 = 27

$C_1 + C_2N + C_3(N-1) + C_4(n-1) + C_5$

$C_1 + C_2N + C_3N - C_3 + C_4n - C_4 + C_5$

$C_1 + (C_2 + C_3 + C_4)N + C_5 - C_4 - C_3$

$(C_2 + C_3 + C_4)N + C_1 + C_5 - C_4 - C_3$

$C_6N + C_7$

$\frac{n(n+1)}{2} - 1$

$7^2 = 49 + 7 = \frac{56}{2} - 1$

$6^2 = 36 + 6 = \frac{42}{2} - 1$

n = 5

1 0, 1, 2, 3, 4   0 = 4 + 1 = 5

2: 1, 2, 3, 4   1 = 3 + 1 = 4

3. 2, 3, 4   2 = 2 + 1 = 3

4 3, 4   1 + 1 = 2

5 + 4 + 3 + 2 = 14

n = 10

$\underbrace{\sum_{i=2}^{n}(i)}_{\text{for loop}}$

2+3+4+5+6+7+8+9+10

$(C(n-1))(C(N)+C) =$

$(Cn - C)(Cn + C)$

$= Cn^2 + Cn - Cn - C = Cn^2 + C$

$\hookrightarrow \frac{C(n(n+1))}{2} - 1$

No idea

Expand $(c(n-1))(c(n)+c)$: $n^2c^2-c^2$

Steps

$(c(n-1))(c(n)+c)$

$=(c(n-1))((n)c+c)$

Apply the distributive law: $a(b+c)=ab+ac$

$a=(c(n-1)),\ b=c(n),\ c=c$

$=(c(n-1))c(n)+(c(n-1))c$

$=(n)c(c(n-1))+c(c(n-1))$

Simplify $(n)c(c(n-1))+c(c(n-1))$: $n^2c^2-c^2$ Show Steps

$=n^2c^2-c^2$

Show Steps

Factor $n^2c^2-c^2$: $c^2(n+1)(n-1)$

Steps

$n^2c^2-c^2$

Factor out common term c^2

$=c^2(n^2-1)$

Factor n^2-1: $(n+1)(n-1)$ Show Steps

$=c^2(n+1)(n-1)$