DSML Intermediate : Time Complexity and Sorting III

Recap

$f(N) = N^3 + 10N^2 + 8$

$= N^3 + N^2$

$\Rightarrow O(N^3)$

Quizzes

Big-O:

0. Calculate no of iterations / operations in the program
1. Drop off constants
2. Take only the highest power in terms of the input (N)

In [ ]:

In [ ]:

In [ ]:

1. Drop off constants
2. Take only the highest power in terms of the input (N)

Quiz - 2

Pseudo Code

```
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (numbers[i] < numbers[j]) {
            ++eqPerms
        }
        else {
            ++neqPerms
        }
    }
}
```

Pseudo Code

=> independent of programming lang.

for i in range(N)

start = 0
end = N-1
inc = 1

for (i = 0 ; i < N ; i++)

start cond^n inc

In [ ]:

Quiz-3

$N^2 + N$

$N^2 + N$

N^2

In [ ]:

In [ ]:

In [ ]:

1. Drop off constants
2. Take only the highest power in terms of the input (N)

Quiz - 2

```
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (numbers[i] < numbers[j]) {
            ++eqPerms
        }
        else {
            ++neqPerms
        }
    }
}
```

i += 1
j += 1
list
eqPerms += 1
neqPerms += 1
} int

```
for i in range(N):
    for j in range(N):
        if numbers[i] < numbers[j]:
            eqPerms += 1
        else:
            neqPerms += 1
```

Python

In [ ]:

1. Drop off constants
2. Take only the highest power in terms of the input (N)

[a,b] => b-a+1

Quiz - 2

```
for (i = 0; i < N; i++)  {
   for (j = 0; j < N; j++) {
      if (numbers[i] < numbers[j]) {
         ++eqPerms
      }
      else {
         ++neqPerms
      }
   }
}
```

[0, N-1]

=> N-1-0+1

=> N

```
for i in range(N):
    for j in range(N):
        if numbers[i] < numbers[j]:
            eqPerms += 1
        else:
            neqPerms += 1
```

i	j	Cnt Iterations
0	[0, N)	N-0 = N
1	[0, N)	N
⋮	⋮	⋮
N-1	[0, N)	N

N(N) = N^2

In [ ]:

1. Drop off constants
2. Take only the highest power in terms of the input (N)

Quiz - 2

```
for (i = 0; i < N; i++)  {
   for (j = 0; j < N; j++) {
      if (numbers[i] < numbers[j]) {
         ++eqPerms
      }
      else {
         ++neqPerms
      }
   }
}
```

0 Ith ✓

1 Constants removed ✓

2

$2N^2$

```
for i in range(N):
    for j in range(N):
        if numbers[i] < numbers[j]:
            eqPerms += 1
        else:
            neqPerms += 1
```

$1(N^2)$

$3N^2$

$\Rightarrow O(N^2)$

In [ ]:

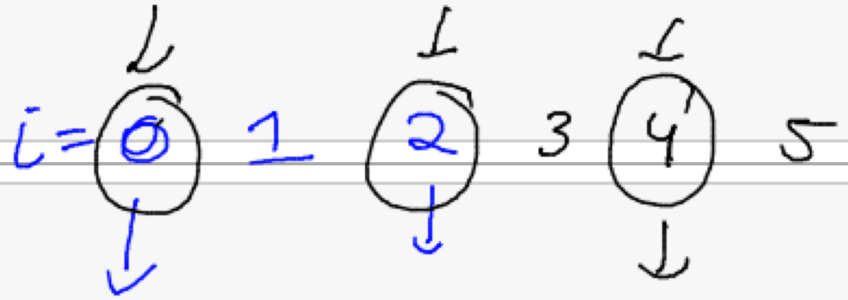
```
            eqPerms += 1
        else:
            neqPerms += 1
```

Quiz - 4

```
for (i = 0; i < N; ++i) {
   if ((i % 2) == 0) {
      outVal[i] = inVals[i] * i
   }
}
```

i = 0 → N

N = 6




```
for i in range(N):
    if i % 2 == 0:
        outVal[i] = inVal[i] * i
```

N

1
2
3

i has to be even.

$f(N) = \frac{N}{2} + c$

In [ ]:

In [ ]:

In [ ]:

In [ ]:


```
for (i = 0; i < N; ++i) {
   if ((i % 2) == 0) {
      outVal[i] = inVals[i] * i
   }
}
```

Iteration N

```
for i in range(N):
    if i % 2 == 0:
        outVal[i] = inVal[i] * i
```

O(N)

In [ ]:

```
i = 0                                   1
while i < N:                            → N+1
    if i % 2 == 0:                      → N
        outVal[i] = inVal[i] * i        → N/2
    i += 1                              → N
```

$1 + N + 1 + N + \frac{N}{2} + N$

$= \frac{3}{2}N + \frac{N}{2} + 2$

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

File Edi nel)

In [ ]:

In [ ]:

Sorting

Bubble Sort

In [1]:
```
def bubble_sort(a):
    j = 0
    N = len(a) # get the length
    while j < N - 1:
        if a[j] > a[j + 1]:
            # swap larger with smaller
            a[j], a[j + 1] = a[j + 1], a[j]
        j += 1
    return a
```

In [2]:
```
bubble_sort([10, 6, 14, 20, 2, 19])
```

Out[2]: [6, 10, 14, 2, 19, 20]

Largest element was settled/bubbled.

In [ ]:

In [ ]:

```
        j += 1
    print(a)
    print('-'*30)

    j = 0
    while j < N - 1:
        if a[j] > a[j + 1]:
            # swap larger with smaller
            a[j], a[j + 1] = a[j + 1], a[j]
        j += 1

    print(a)
    print('-'*30)
    return a
```

6, 10, 14, 2, 19, 20

In [6]:
```
bubble_sort([10, 6, 14, 20, 2, 19])
```

```
[6, 10, 14, 2, 19, 20]
------------------------------
[6, 10, 2, 14, 19, 20]
------------------------------
```

6, 10, 2, 14, 19, 20

Out[6]:
```
[6, 10, 2, 14, 19, 20]
```

In [ ]:

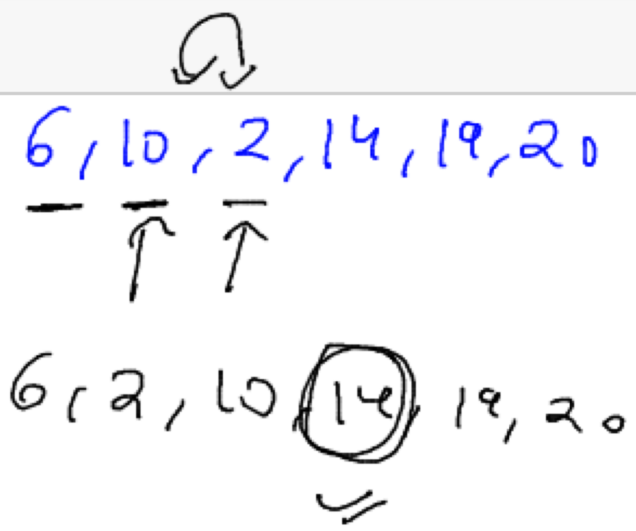
In [ ]:

```
    print(a)
    print('-'*30)

    j = 0
    while j < N - 1:
        if a[j] > a[j + 1]:
            # swap larger with smaller
            a[j], a[j + 1] = a[j + 1], a[j]
        j += 1

    print(a)
    print('-'*30)
    return a
```

In [8]:
```
bubble_sort([10, 6, 14, 20, 2, 19])
```

```
[6, 10, 14, 2, 19, 20]
------------------------------
[6, 10, 2, 14, 19, 20]
------------------------------
[6, 2, 10, 14, 19, 20]
------------------------------
```

Out[8]: [6, 2, 10, 14, 19, 20]

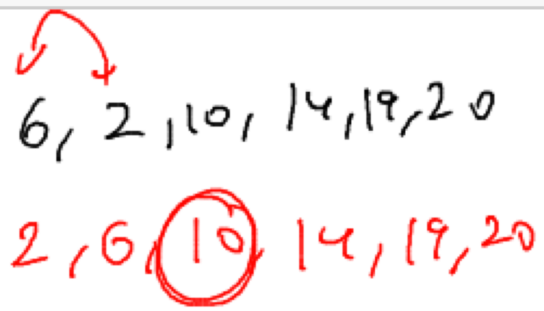


In [ ]:

```
    print(a)
    print('-'*30)

    j = 0
    while j < N - 1:
        if a[j] > a[j + 1]:
            # swap larger with smaller
            a[j], a[j + 1] = a[j + 1], a[j]
        j += 1

    print(a)
    print('-'*30)
    return a
```

In [10]:
```
bubble_sort([10, 6, 14, 20, 2, 19])
```

```
[6, 10, 14, 2, 19, 20]
------------------------------
[6, 10, 2, 14, 19, 20]
------------------------------
[6, 2, 10, 14, 19, 20]
------------------------------
[2, 6, 10, 14, 19, 20]
------------------------------
```



Out[10]:
```
[2, 6, 10, 14, 19, 20]
```

In [ ]:

```
    print(a)
    print('-'*30)

    j = 0
    while j < N - 1:
        if a[j] > a[j + 1]:
            # swap larger with smaller
            a[j], a[j + 1] = a[j + 1], a[j]
        j += 1

    print(a)
    print('-'*30)
    return a
```

n = 5

In [11]:
```
bubble_sort([9, 8, 7, 6, 5])
```

```
1 [8, 7, 6, 5, 9]
  ------------------------------
2 [7, 6, 5, 8, 9]
  ------------------------------
3 [6, 5, 7, 8, 9]
  ------------------------------
4 [5, 6, 7, 8, 9]
  ------------------------------
```

n - 1 iterations

Out[11]:
```
[5, 6, 7, 8, 9]
```

In [ ]:

In [15]:
```
bubble_sort([9, 8, 7, 6, 5])
```

```
[8, 7, 6, 5, 9]
------------------------------
[7, 6, 5, 8, 9]
------------------------------
[6, 5, 7, 8, 9]
------------------------------
[5, 6, 7, 8, 9]
------------------------------
```

Out[15]: [5, 6, 7, 8, 9]

start end inc=

for (j=0; j<N-1; j++)

for j in range(N-1)

In [ ]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        j = 0
        while j < N - 1:
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
            j += 1
        print(a)
        print('-'*30)
    return a
```

In [ ]:

File Edi nel)

14 / 14

In [16]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
            j += 1
        print(a)
        print('-'*30)
    return a
```

In [17]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[17]:
```
[5, 6, 7, 8, 9, 10, 11]
```

15 / 15

In [1

```
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
            j += 1
        print(a)
        print('-'*30)
    return a
```

$(a-b)^2 = a^2 + b^2 - 2ab$

$(N-1)^2$

$= N^2 + X - 2N$

$= O(N^2)$

Worst Case

N = 7

In [17]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

6 it

```
i=0 [10, 9, 8, 7, 6, 5, 11]
----------
i=1 [9, 8, 7, 6, 5, 10, 11]
----------
i=2 [8, 7, 6, 5, 9, 10, 11]
----------
i=3 [7, 6, 5, 8, 9, 10, 11]
----------
i=4 [6, 5, 7, 8, 9, 10, 11]
----------
i=5 [5, 6, 7, 8, 9, 10, 11]
----------
```

Out[17]: [5, 6, 7, 8, 9, 10, 11]

i	j	# iterations
0	[0, N-1)	N-1
1	[0, N-1)	N-1
2	⋮	⋮
⋮	⋮	⋮
N-2	[0, N-1)	N-1
# N-1		(N-1)(N-1)

In [ ]:

File Edi

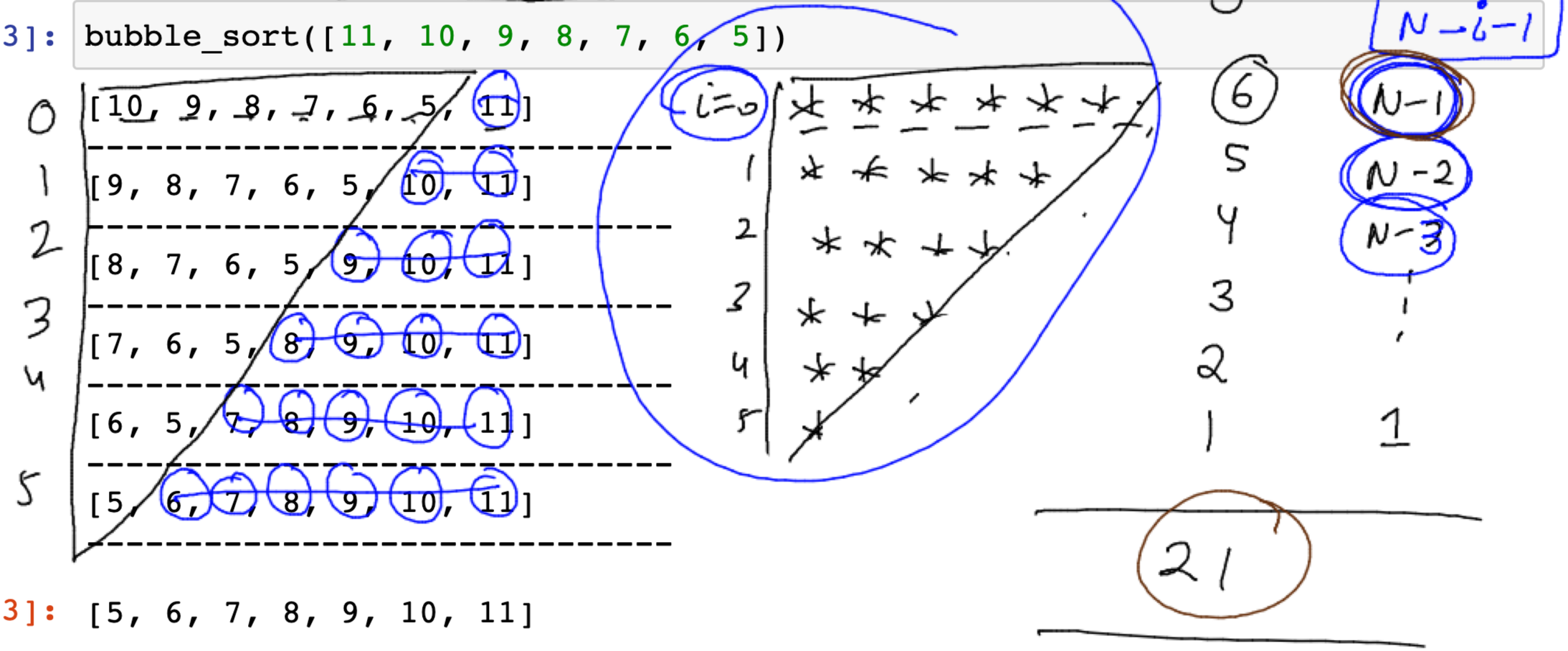
In [42]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
        print(a)
        print('-'*30)
    return a
```

$1+2+\cdots+N = \frac{N(N+1)}{2}$

$\frac{6*7}{2} = \frac{42}{2} = 21$

$N = 7$

In [43]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```



Out[43]: [5, 6, 7, 8, 9, 10, 11]

In [44]:
```
bubble sort([5, 4, 1, 2, 7, 8, 0, 10])
```

In [ ]:
```
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
            j += 1
        print(a)
        print('-'*30)
    return a
```

In [18]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[18]: [5, 6, 7, 8, 9, 10, 11]

In [ ]:

Out[2

In [22]:
```
bubble_sort([5, 4, 1, 2, 7, 8, 0, 10])
```

```
[4, 1, 2, 5, 7, 0, 8, 10]
--------------------------------
[1, 2, 4, 5, 0, 7, 8, 10]
--------------------------------
[1, 2, 4, 0, 5, 7, 8, 10]
--------------------------------
[1, 2, 0, 4, 5, 7, 8, 10]
--------------------------------
[1, 0, 2, 4, 5, 7, 8, 10]
--------------------------------
[0, 1, 2, 4, 5, 7, 8, 10]
--------------------------------
[0, 1, 2, 4, 5, 7, 8, 10]
--------------------------------
```

last ? pass →

Out[22]:
```
[0, 1, 2, 4, 5, 7, 8, 10]
```

In [ ]:

In [ ]:

In [ ]:

```
[1, 2, 0, 4, 5, 7, 8, 10]
-----------------------------
[1, 0, 2, 4, 5, 7, 8, 10]
-----------------------------
[0, 1, 2, 4, 5, 7, 8, 10]
-----------------------------
[0, 1, 2, 4, 5, 7, 8, 10]
-----------------------------
```

Out[22]: `[0, 1, 2, 4, 5, 7, 8, 10]`

In [26]:
```
bubble_sort([1, 2, 3, 5, 4, 7, 6])
```

```
[1, 2, 3, 4, 5, 6, 7]
-----------------------------
[1, 2, 3, 4, 5, 6, 7]
-----------------------------
[1, 2, 3, 4, 5, 6, 7]
-----------------------------
[1, 2, 3, 4, 5, 6, 7]
-----------------------------
[1, 2, 3, 4, 5, 6, 7]
-----------------------------
[1, 2, 3, 4, 5, 6, 7]
-----------------------------
```

Out[26]: `[1, 2, 3, 4, 5, 6, 7]`

In [ ]:

```
[0, 1, 2, 4, 5, 7, 8, 10]
------------------------------
[0, 1, 2, 4, 5, 7, 8, 10]
------------------------------
```

Out[22]:
```
[0, 1, 2, 4, 5, 7, 8, 10]
```

In [27]:
```
bubble_sort([1, 2, 3, 5, 4, 7, 6])
```

```
[1, 2, 3, 4, 5, 6, 7]
------------------------------
[1, 2, 3, 4, 5, 6, 7]
------------------------------
[1, 2, 3, 4, 5, 6, 7]
------------------------------
[1, 2, 3, 4, 5, 6, 7]
------------------------------
[1, 2, 3, 4, 5, 6, 7]
------------------------------
[1, 2, 3, 4, 5, 6, 7]
------------------------------
```

Out[27]:
```
[1, 2, 3, 4, 5, 6, 7]
```

In [ ]:

In [ ]:

In [33]:
```
def optimized_bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
            j += 1
        print(a)
        print('-'*30)
    return a
```

unhappy student ?

In [34]:
```
optimized_bubble_sort([1, 2, 3, 4, 5, 6])
```

all happy
=> no swaps !!

```
[1, 2, 3, 4, 5, 6]
------------------------------
[1, 2, 3, 4, 5, 6]
------------------------------
[1, 2, 3, 4, 5, 6]
------------------------------
[1, 2, 3, 4, 5, 6]
------------------------------
[1, 2, 3, 4, 5, 6]
------------------------------
[1, 2, 3, 4, 5, 6]
------------------------------
```

Out[34]:
```
[1, 2, 3, 4, 5, 6]
```

Worst case time complexity:

In [3

```
    N = len(a) # get the length
    for i in range(N - 1):
        num_swaps = 0
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
                num_swaps += 1
            j += 1
        if num_swaps == 0:
            break
        print(a)
        print('-'*30)
    return a
```

HW:
Try making
code simpler

In [39]:
```
optimized_bubble_sort([4, 5, 4, 2, 5, 6])
```

Can you use a boolean variable?

```
[4, 4, 2, 5, 5, 6]
------------------------------
[4, 2, 4, 5, 5, 6]
------------------------------
[2, 4, 4, 5, 5, 6]
------------------------------
```

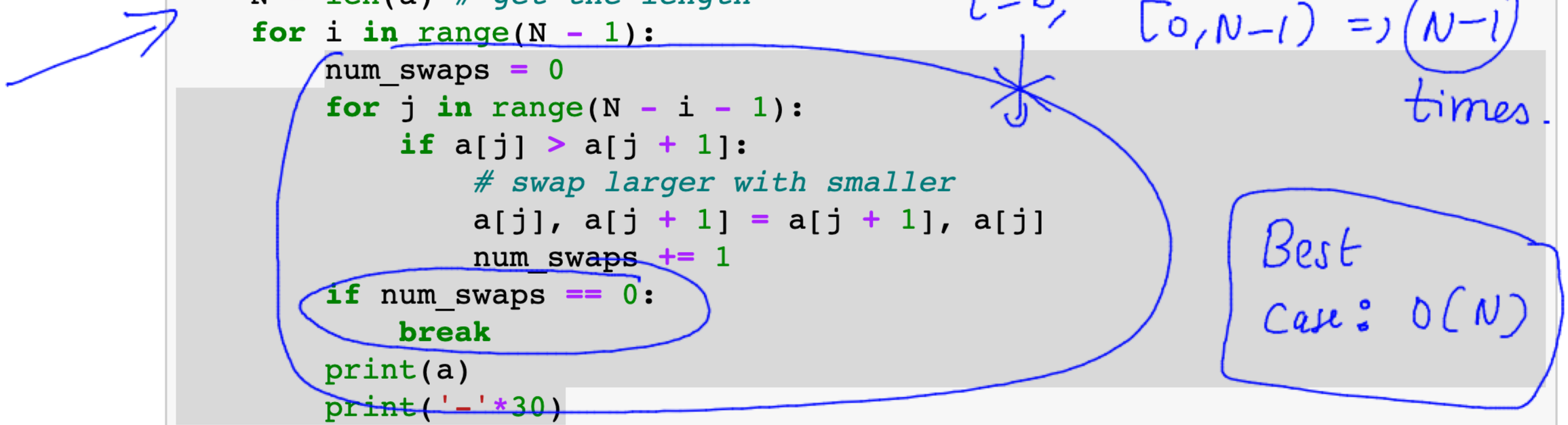
Out[39]:
```
[2, 4, 4, 5, 5, 6]
```

Worst case time complexity:

In [ ]:

Optimized Bubble Sort

In [46]:
```
def optimized_bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        num_swaps = 0
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
                num_swaps += 1
        if num_swaps == 0:
            break
        print(a)
        print('-'*30)
    return a
```



In [47]:
```
optimized_bubble_sort([4, 5, 4, 2, 5, 6])
```

```
[4, 4, 2, 5, 5, 6]
------------------------------
[4, 2, 4, 5, 5, 6]
------------------------------
[2, 4, 4, 5, 5, 6]
------------------------------
```

```
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[41]: `[5, 6, 7, 8, 9, 10, 11]`

Best case for this code?

In [42]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
        print(a)
        print('-'*30)
    return a
```

=> $O(N^2)$ worst & best case.

In [43]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
```

```
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[41]:
```
[5, 6, 7, 8, 9, 10, 11]
```

In [42]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
        print(a)
        print('-'*30)
    return a
```

In [43]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
```


```
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[41]: `[5, 6, 7, 8, 9, 10, 11]`

In [42]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
        print(a)
        print('-'*30)
    return a
```

In [43]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
```


```
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[41]: `[5, 6, 7, 8, 9, 10, 11]`

In [42]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
        print(a)
        print('-'*30)
    return a
```

In [43]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
```

```
[5, 6, 7, 8, 9, 10, 11]
------------------------------
```

Out[41]: [5, 6, 7, 8, 9, 10, 11]

In [42]:
```
def bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
        print(a)
        print('-'*30)
    return a
```

After 1 Complete pass

In [43]:
```
bubble_sort([11, 10, 9, 8, 7, 6, 5])
```

```
[10, 9, 8, 7, 6, 5, 11]
------------------------------
[9, 8, 7, 6, 5, 10, 11]
------------------------------
[8, 7, 6, 5, 9, 10, 11]
------------------------------
[7, 6, 5, 8, 9, 10, 11]
------------------------------
[6, 5, 7, 8, 9, 10, 11]
------------------------------
[5, 6, 7, 8, 9, 10, 11]
```

Optimized Bubble Sort

In [46]:
```
def optimized_bubble_sort(a):
    N = len(a) # get the length
    for i in range(N - 1):
        num_swaps = 0
        for j in range(N - i - 1):
            if a[j] > a[j + 1]:
                # swap larger with smaller
                a[j], a[j + 1] = a[j + 1], a[j]
                num_swaps += 1
        if num_swaps == 0:
            break
        print(a)
        print('-'*30)
    return a
```

In [47]:
```
optimized_bubble_sort([4, 5, 4, 2, 5, 6])
```

```
[4, 4, 2, 5, 5, 6]
------------------------------
[4, 2, 4, 5, 5, 6]
------------------------------
[2, 4, 4, 5, 5, 6]
------------------------------
```

Out[47]:
```
[2, 4, 4, 5, 5, 6]
```

You have left the meeting

We get frequent requests for your notes!

Notes written by you helps in understanding the topic better. You can upload the notes in two simple steps mentioned below

1 **Scan the QR code with your iPad**
Scanner should be present in the top menu on your iPad

2 **Upload Notes on the generated link**
All notes uploaded will be visible in the saved version of this session

OR

Drag and drop files or **click here to upload**

Files Uploaded from your computer appear here