Lecture 9

Date: 20-Jan-2021

Date in time Complexity.

In reverse efficiency of algo.

A:
B:

which is better?

Answer depends on function of the number n.

↳ Helpful chart → (10).

Algorithm runtime → the time that it takes to execute as a function of input size.

↳ 0-100 m.

The data will not be linear for length input of n until in better.

n=1	21
n=2	22
n=3	23
n=4	24
n=5	25
n=6	26
n=7	27
n=8	28
n=9	29
n=10	210
n=11	211
n=12	212
n=13	213
n=14	214
n=15	215
n=16	216
n=17	217
n=18	218
n=19	219
n=20	220

2^{n-1}

$2^{n-1} \le \frac{2^{\log_2 n}}{n}$

↳ $\log_2 n$

$\log_2 n$

Date

$n \le 2(n-1) < 2(n) < 2(n+1) < 2(n+2) < 2(n+3) < 2(n+4)$

↳ $O(n)$

↳ $O(n)$

While loop fun?

Linear Search.

0	1	2	3	4	5	6
5	1	2	3	4	5	6
0	1	2	3	4	5	6

↳ $O(n)$

for-> n

$\sum_{i=1}^{n} O(n) = O(n)$

for-> n

$\sum_{i=1}^{n} O(n) = O(n)$

↳ $O(n)$.

```
for (int i=1; i<n; i++){
    for ( ) {
        }
    }
}
```

// $O(n)$

```
for (int i=1; i<n; i++){
    for (int j=1; j<n; j++){
        }
    }
}
```

// $O(n^2)$

n times ... n times = $O(n^2)$.

```
for ( ) {
    for ( ) {
        for {
            }
        }
    }
}
```

↳ $O(n^3)$.

Bubble Sort:-

```
for (int i=arr.length-1; i>=0; i--){
    for (int j=0; j<=i-1; j++){
        if (arr[j] > arr[j+1]){
            temp = j;
        }
    }
}
```

temp ( )

int temp = a ↳ c

a = b; ↳ c

b = temp; ↳ c

⇒ c+c+c = 3c

→ $O(1)$

↳ $O(1)$

$n-1 + n-2 + n-3 + \dots + 1 = (n)(n-1)/2$

$= \frac{n^2}{2} - \frac{n}{2}$

$O(n^2)$

$(y)^x \to (m)^y$

$\sqrt[y]{(m)^x}$

$\sqrt[2]{n}$, i.e.

$n^{\frac{1}{2}}$

$\frac{1}{2}$

$\frac{1}{2}$

$\frac{1}{2}$

$f_{y}(n)^{\frac{1}{2}}$

$\left(\log_2 1=0\right)$

↳ (10)

```
while (n%2==0) {
    n = n/2
    m = m*2
    n = n-m;
}
```

1=0, ↳ 0

1=1, ↳ 1

1=2 → 2=1

2=3 → 3=2

3=4 → 4=3

4=5 → 5=4

$3/2 \to 1$

both the

$\left(\log_2 m\right)$

```
for (int i=1; i<n; i++)
    for (int j=m; j>0)
        ≡
    }
    for (int k=1; k<n; k++)
        ≡
}
```

n^2

→ $n+n^2$.

↳ $O(n^2)$.

n

$n^2 + n$.

$n = 10^{10} \to \boxed{10^{10}} + \boxed{10^{10}}$

Q: merge 2 sorted array.

arr1 = [1, 3, 5, 8] n

arr2 = [2, 4, 6] m

MAX = 1, 2, 3, 4, 5, 6, 7, 8

Let c:- [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

m+n space

$\boxed{O(n+m)} \to O(n+m)$

Idea → arr1 = 1, 3, 5, 8 ; c = 1, 3

arr2 = 2, 4, 6

↳ 1, 2, 3, 4, 5, 6, 7, 8

Let c:- [1] [2] [3] [4] [5] [6] [7] [8]

arr1 [1] [3] [5] [8]

arr2 [2] [4] [6]

```
public static int[] merge (int[] arr1, int[] arr2){
    int[] arr = new int[arr1.length + arr2.length];
    int i=0, j=0, k=0;
    while (i<arr1.length && j<arr2.length){
        if (arr1[i] < arr2[j]){
            arr[k] = arr1[i];
            i.plus(); k++; i++;
        } else {
            arr[k] = arr2[j];
            j++; k++;
        }
    }
    while (i<arr1.length){
        arr[k] = arr1[i];
        k++;
        i++;
    }
    while (j<arr2.length){
        arr[k] = arr2[j]
        j++;
        k++;
    }
}
```

m-1 + n.

m

n

0 $\boxed{(m+n)}$ >>

$O(mn)$.

$m = 10^{10}$

$n = 10^{10}$.

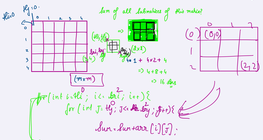
$m - \cancel{1} + n \cancel{+1} + 0$

$= (m+n)$

Q: Sum

$\boxed{Sum = 0}$

Sum of all Submatrix of this matrix



(m,n)

$(1,1)$

$(2,2)$

$(3,3)$

$(4,4)$

$(5,5)$

(m,m)

$1+2+3+4$

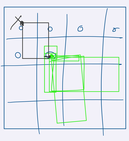
$\to 1+2+4$

$\to 1+2+4$

$\to$ 16 times

$(0,0)$ $(0,1)$ $(1,2)$

$\boxed{(1,2)}$



Can you reduce it.

```
for (int i=1; i<=n; i++){
    for (int j=1; j<=n; j++){
        }
    }
}
```

m = arr.length

$\boxed{Sum = 0}$

```
for (int k=0; k<m;  ; k++){
    for (int l=0; l<n;  ; l++){
        for (int i=k; i<m;  ; i++){
            for (int j=l; j<n; j++){
                sum = sum + arr[i][j];
            }
        }
    }
}
```

m

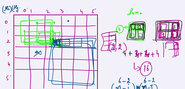
m

m

m

m

↳ $O(n^6)$.



(k,l) 0 1 2 3 4 5

A B C

(k,l)

(k,l)

(k,l)

(k,l)

(k,l)

(k,l)

Sum.

$\boxed{0}$ $\boxed{1}$ $\boxed{2}$ $\boxed{3}$ $\boxed{4}$ $\boxed{5}$

↳ (11)

$(n-1)^{(n-1)}$

(4×3)

$\boxed{Sum = (Sum+1) + (Sum+1)}$

$\boxed{Sum = (Sum + 1 + Sum + 1)}$

↳ $O(n^2)$

$\boxed{n^2 << n^6}$

Resume at 8:53.

Sol:- Sum of Evaluations $(2-2^n)$.

↳ $\boxed{2-N}$ ⇒ N.

0	1	2	3	4	5	6
✓	✓	✓	✓	✓	✓	
✓	✓	✓	✓	✓	✓	
✓	✓	✓	✓	✓	✓	
✓	✓	✓	✓	✓	✓	
✓	✓	✓	✓	✓	✓	
✓	✓	✓	✓	✓	✓	
✓	✓	✓	✓	✓	✓	

2 → ✓

3 → ✓

4 → ✗

5 → ✓

6

$\boxed{\sqrt{n}}$

↳ $O(\log \log n)$

↳

$(n \times n) \Rightarrow \boxed{n^2}$

```
nsqrt = 2;
while ( nsqrt * nsqrt < n){
    }
    for (int j=0; j<n; j++){
        int k = (i+1)*(j+1)
        int la = (n-1)-i-(n-j);
        Sum = Sum + (k*la) arr[i][j]
    }
}
```

↳ $T(n) \to O(n^2)$

$\boxed{14 \ 16 + 2} \Rightarrow (32)$

$\boxed{n^2 << n^6}$

Resume at 8:53.

```
nsqrt = 2;
while ( nsqrt * nsqrt < n){
    }
    for (int i=0; i<n; i++){
        for (int j=0; j<n; j++){
            int k = (i+1)*(j+1)
            int la = (n-1)-i-(n-j);
            Sum = Sum + (k*la) arr[i][j]
        }
    }
}
```

↳ $T(n) \to O(n^2)$

$\boxed{n^2 << n^6}$

(4×3)

	1	2	3
0			
1			
2			
3			

```
nsqrt = 2;
while ( nsqrt * nsqrt < n){
    }
    for (int i=0; i<n; i++){
        for (int j=0; j<n; j++){
            int k = (i+1)*(j+1)
            int la = (n-1)-i-(n-j);
            Sum = Sum + (k*la) arr[i][j]
        }
    }
}
```

↳ $T(n) \to O(n^2)$

$\boxed{n^2 << n^6}$

$\boxed{14 \ 16 + 2} \Rightarrow (32)$

```
nsqrt = 2;
while ( nsqrt * nsqrt < n){
    }
    for (int multiplier = 2; multiplier * nsqrt < n; multiplier++){
        primes [multiplier * nsqrt] = false;
    }
}
nsqrt++;
}
```

```
max_subarray_sum :-
    arr = { 2, 1, 5, -3, -4, 6 };
    for (int i=1; i<arr.length; i++){
        for (int j=1; j<arr.length-(i-1); j++){
            int[] temp = new int[arr.length-(i-1)];
            int sum = 0;
            for (int k=j; k<=j+(i-1); k++){
                temp[k-j] = arr[k];
                sum = sum + temp[k-j];
            }
            if (sum > max){
                max = sum;
            }
        }
    }
```

$\boxed{n^3}$

```
public static int maxSubarray(int[] arr){
    int max = Integer.MIN_VALUE;
    int sum = 0;
    for (int i=1; i<arr.length; i++){
        for (int j=1; j<=arr.length-(i-1); j++){
            int[] temp = new int[arr.length-(i-1)];
            int sum = 0;
            for (int k=j; k<=j+(i-1); k++){
                temp[k-j] = arr[k];
                sum = sum + temp[k-j];
            }
            if (sum > max){
                max = sum;
            }
        }
    }
}
```

$\boxed{n^3}$

Kadane's Algorithm.

```
public static int maxSubarray(int[] arr){
    int max = 0;
    int sum = 0;
    for (int i=1; i<arr.length; i++){
        sum = sum + arr[i];
        if (sum > max){
            max = sum;
        }
        if (sum < 0){
            sum = 0;
        }
    }
}
```

$\boxed{n^2}$

$\boxed{sum = sum + arr[i]}$

$\boxed{max = sum}$ ①

$\boxed{-1}$ → -1, -3

Exec. Complexity

Algorithm

Write class → Interface

↳ $\boxed{\text{Resume}}$