Part 1

1. Arrays
2. Linked List
3. Stacks
4. Queues
5. Hash tables.

Big (O)

Algorithm এর performance বোঝায়।

Array, LinkList এ আছে [time & space complexity] Analysis.

O(1)

↳ constant time ↳ run করবে।

s.o.p (n[0]);

↳ n Array এর size এখানে matter করবে না।

যেকোনো constant time ই লাগবে

Always.

s.o.p (n[0]);
s.o.p (n[1]);] O(2)

but generally we assume the above code to be having a complexity of running constant time.

O(n):- (Input size dependent)

S-1

```
for (int x : number) {
    s.o.p (x);
}
```

→ O(n)

S-2

```
s.o.p ()          ——— (1)
for (int x : n){  ——— (n)
    s.o.p (x);
}                 ——— (n)
s.o.p ()          ——— (1)
```

মনে হচ্ছে $O(n+1+1) = O(n+2)$

↳ negligable

Case input size million হলেও তার যোগে ২ যোগ করলেও তেমন কিছু হচ্ছে না। চলনবিল সাগর থেকে ১ বালতি পানি তোলার মতো। (এতে সাগরের পানি কমবে না)।

S-3

mA (P_1 , P_2) {

for (int n : x) { ——— o(n) → O(n)

S.O.P (n);

}

for (int n : y) { ——— o(n) → O(n)

S.O.P (n);

}

→ negligible

o(n+n) = O(2n)

= O(n)

O(n+m)

but linearly increase

করে তাই O(n)

ধরা হবে।

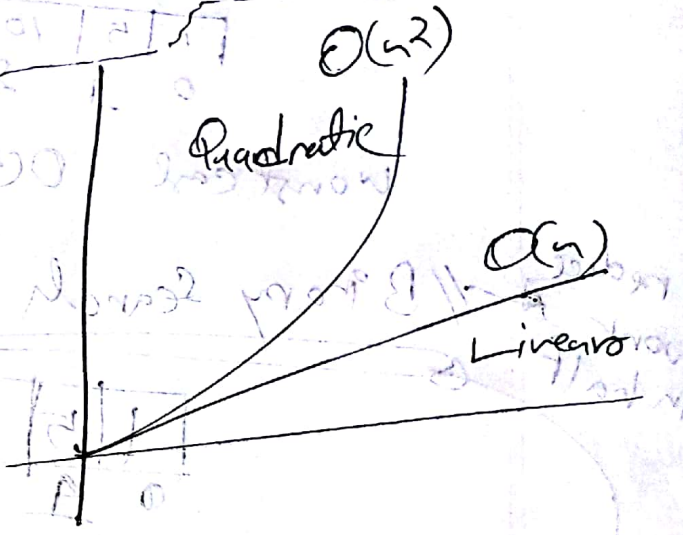
$O(n^2)$

nested for loop

```
for (int n=1; n<5; n++){
    for (int k=1; k<=3; k++){
        s.o.p(k+n);
    }
}
```

$O(n*n) = O(n^2)$

↳ nested loop থাকা হলে তখন difference হয় না but nested loop এলোমেলো বেশি হলে বেশি বড় মাপের difference দেখা যায়।



another for loop → $O(n)$

before → duplicate → $O(n^2)$

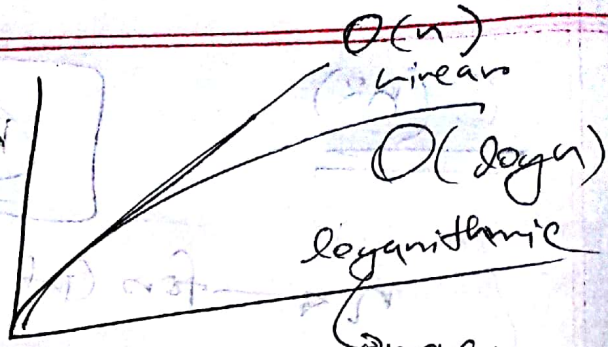
$\therefore O(n+n^2) \equiv O(n^2)$

we need approximation not exact value.

আরেকটা nested loop হলে $O(n^3)$ হত।

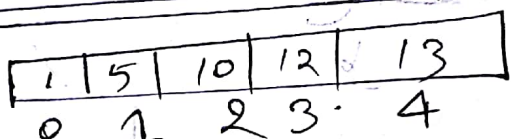
$O(n^3) < O(n^2)$ [fast]

O(log n)



more scalable than O(n)
O(n²).

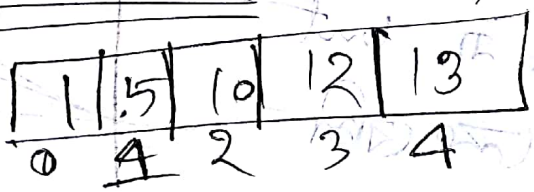
Linear Search



worst case O(4) ≡ O(n)

reduce work in half

// Binary Search: (But array sorted হতে হবে)



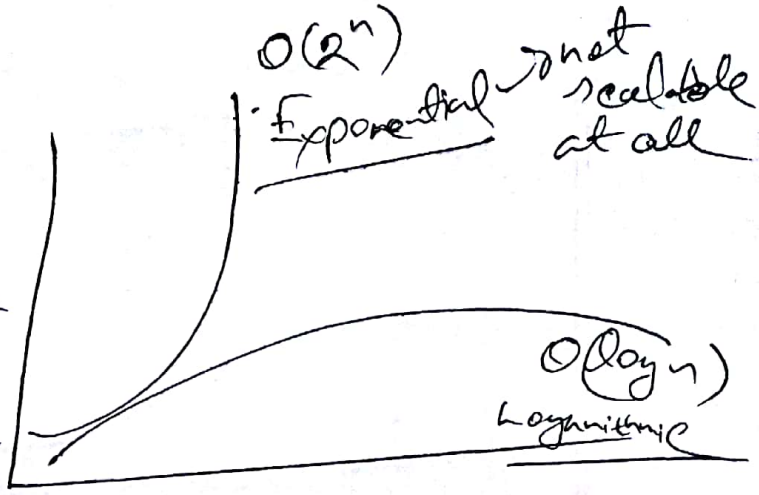
O(log n)

number < mid go left
number > mid go right
then again mid (left/right)/2

array কত বড় নিয়ে খুবে 1

1M items → only 19 comparison

$O(2^n)$



Exponential is opposite logarithmic growth

Summary (O):

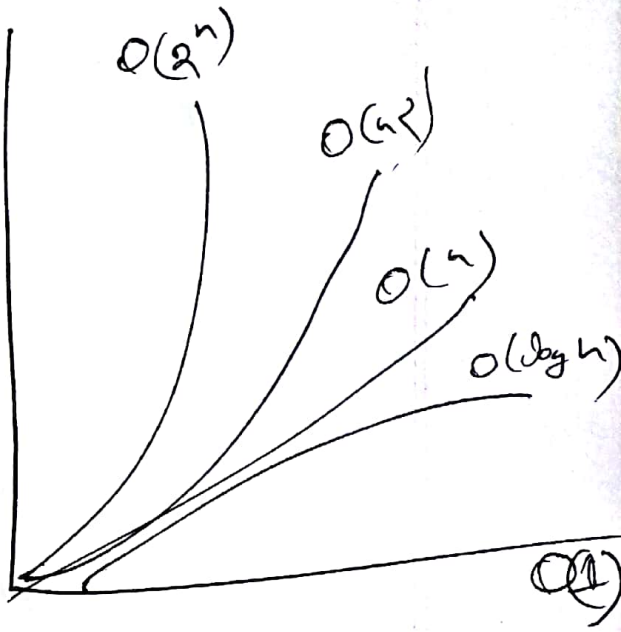
O(1) → constant

O(log n) → logarithmic

O(n) → linear

O(n2) → Quadratic

O(2^n) → Exponential



Space Complexity :-

memory

```
MA ( String[] names) {
// O(1) → space
for loop → সর্বোচ্চ 1 পরিমাণ space
case input যত হোক নিচের লাইনগুলো,
```

but

```
MA (String[] names) {
// O(n) → space.
String[] copy = new String[names];
```