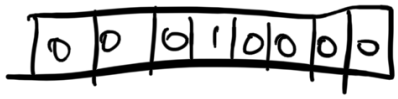
ARRAYS

integer

int → 4 bytes → 32 0/1

1 bit → 0/1

8 bits → byte



long long int → 8 byte → 64 bits (0/1)

Char → 1 byte → 8 bits (0/1)

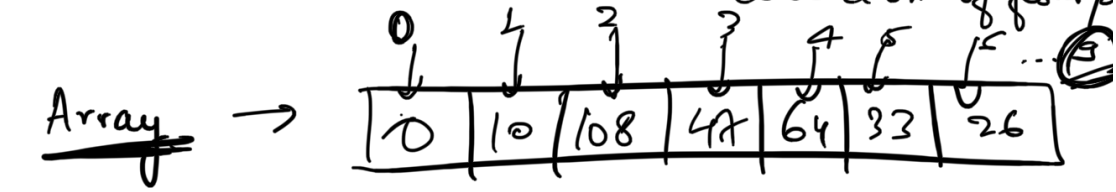
① Virat Kohli score → collection of integers

② 12293.26 Rs

−269.86 Rs

float / double

Collection of float/d



End Quick ← ✓ ① Insert a new entry.

Quick ← ✓ ② Get / Access

✓ ③ Delete

Quick ← ✓ ④ Update

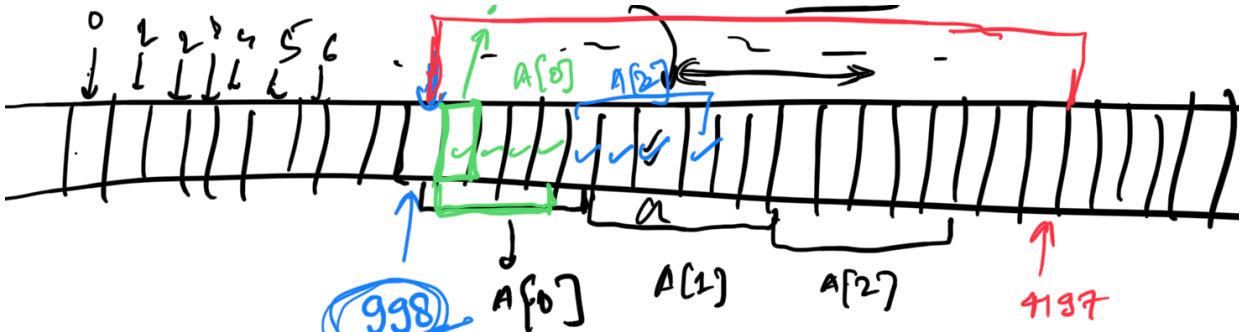
A[9]

1 byte

← 1/0

↓

processor



A[0]

100 * 4 byte
8 bits
3200 bits.

998 — 998 + 3200 − 1
4197

A[0]
998 — 998+31
A
998+32 — 998+
998+64 — 998

A[i]
998 → A[0]

A[1] 998 + 32
A[2] → 998 + 64
⋮
A[i]

A[i] → base address + (32 * i) (byte)
998

Addr

A[i] → base address + 4 * i

A — 100 integers

998 — 998+400 — 100 * 4 bytes — 400

A[400]

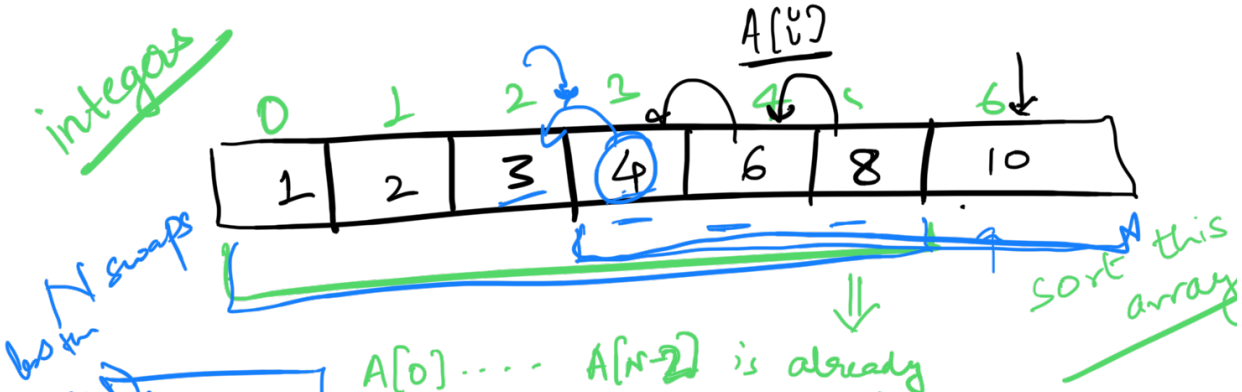
Out of index bound
segmentation fault

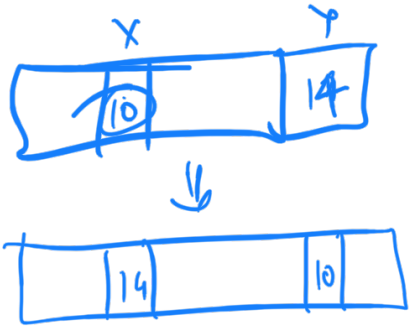
998 + 4 * 400

Static array

int A[100]

A[i] = 10





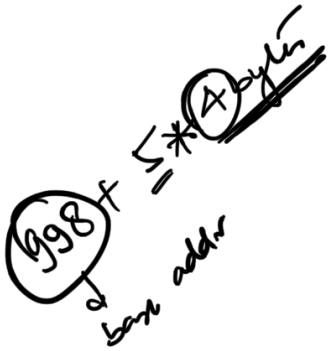
A

Swap(x, y) :

temp = A[x]

A[x] = A[y]

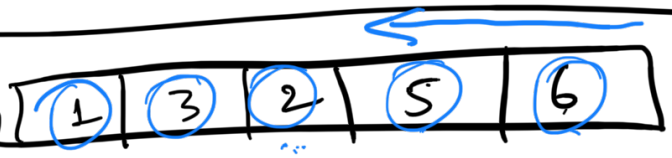
A[y] = temp



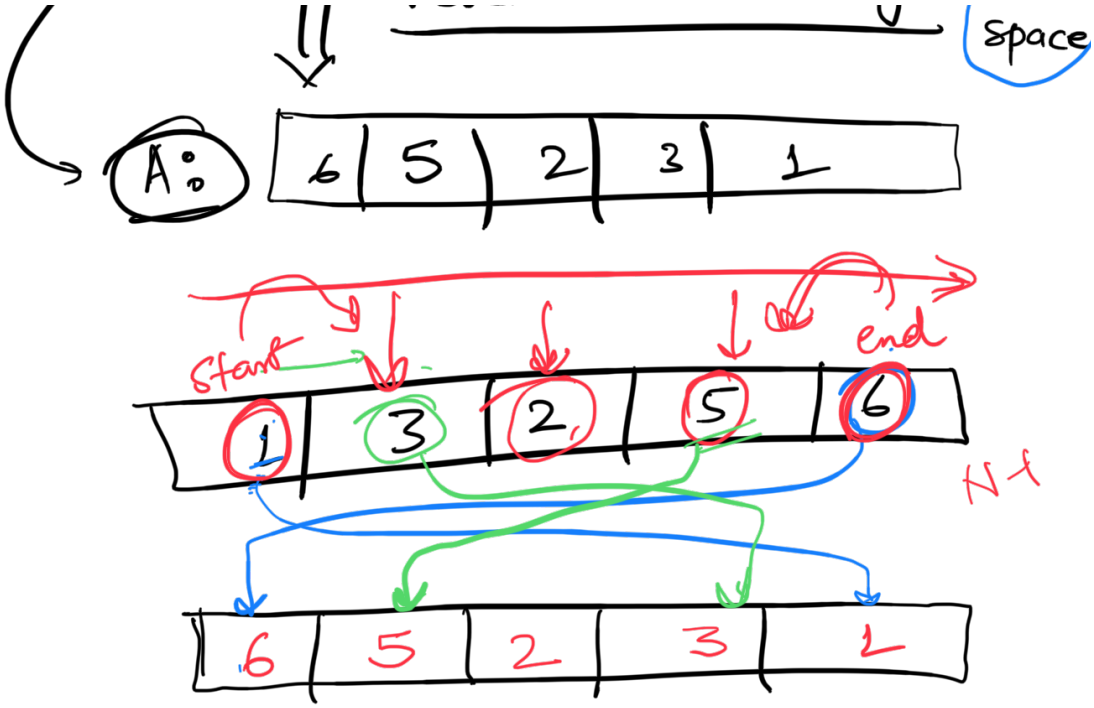
O(N) Time complexity.

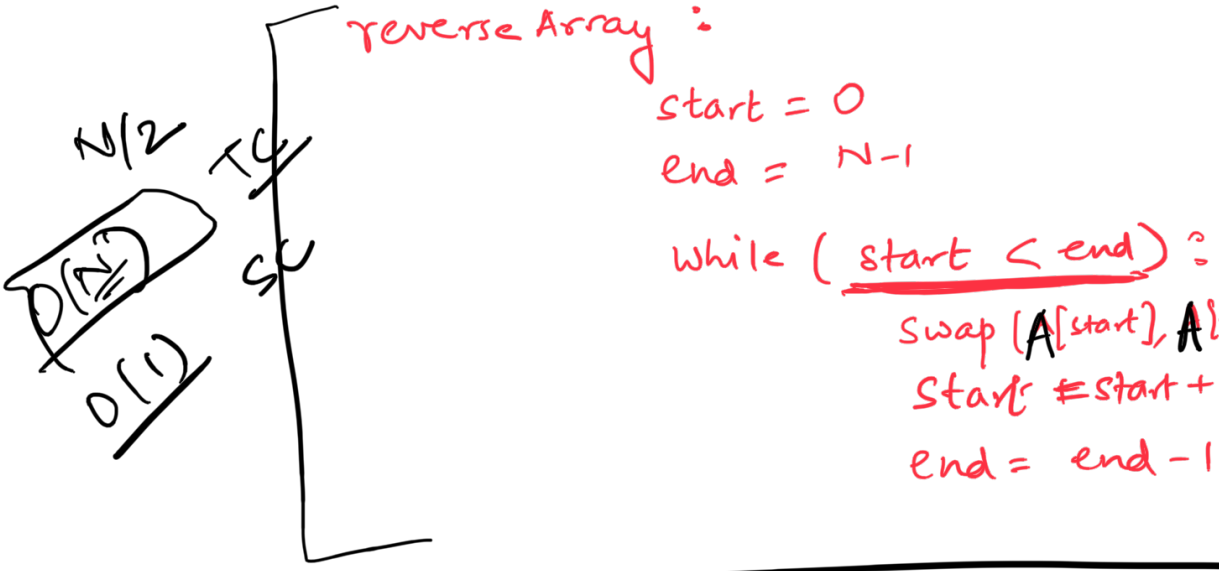
O(1) space

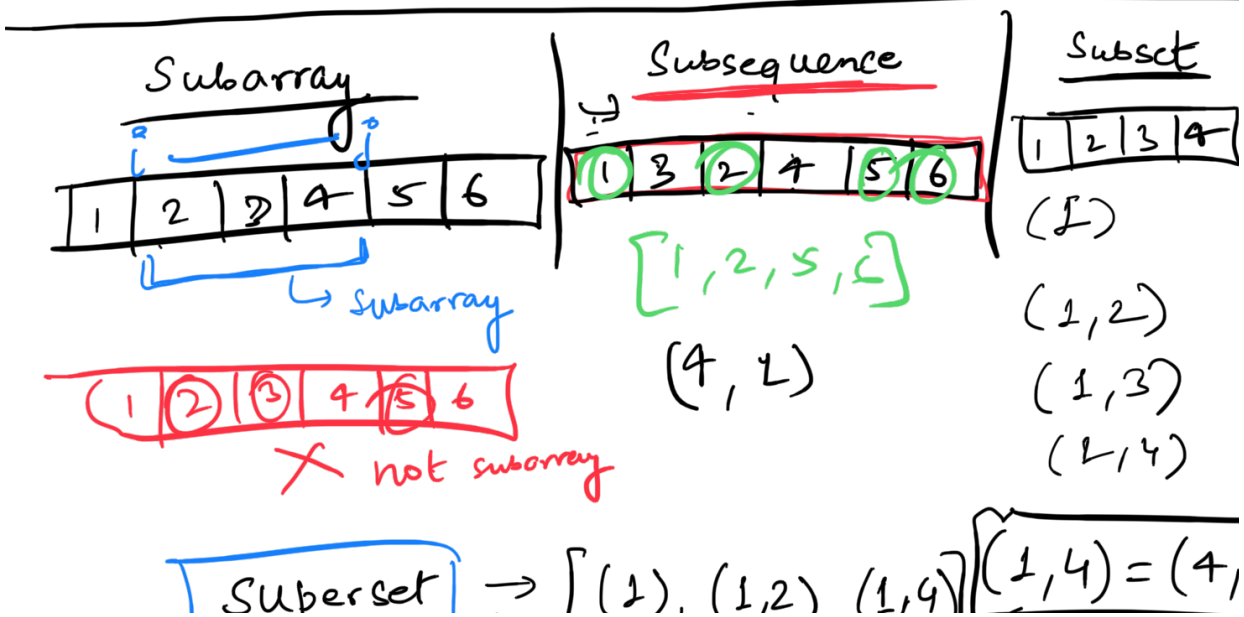
A of integers



// reverse the array in O(1)


Space
A:
6 5 2 3 1
start
end
1 3 2 5 6
N-1
6 5 2 3 1


reverse Array :
start = 0
end = N-1
while (start < end) :
swap (A[start], A[
start = start +
end = end - 1
N/2
TC
SC
O(N/2)
O(1)

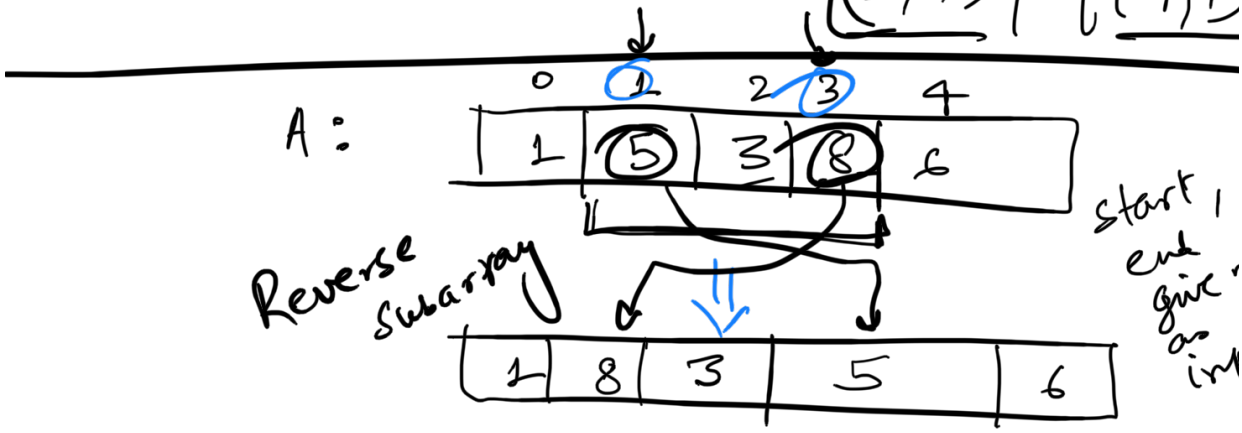

Subarray
1 2 3 4 5 6
subarray
1 2 3 4 5 6
not subarray
Subsequence
1 3 2 4 5 6
[1,2,5,6]
(4,2)
Subset
1 2 3 4
(1)
(1,2)
(1,3)
(2,4)
Subset → [(1), (1,2) (1,4)
(1,4) = (4,

A: [illegible]

B: [(1) (1,2)]

A is superset

[(1,4)] [(4,1)]



reverseArray(A, i, j):

start = i

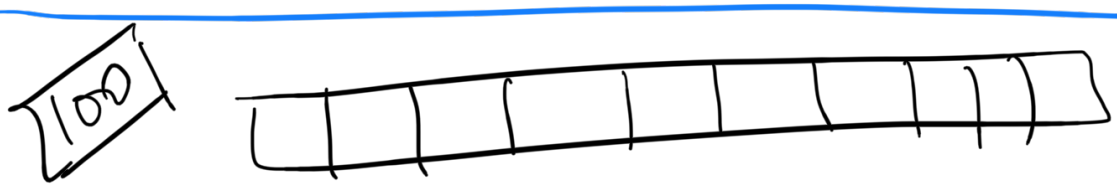
end = j

while (start < end)

swap(A[start], A

start ++

end --

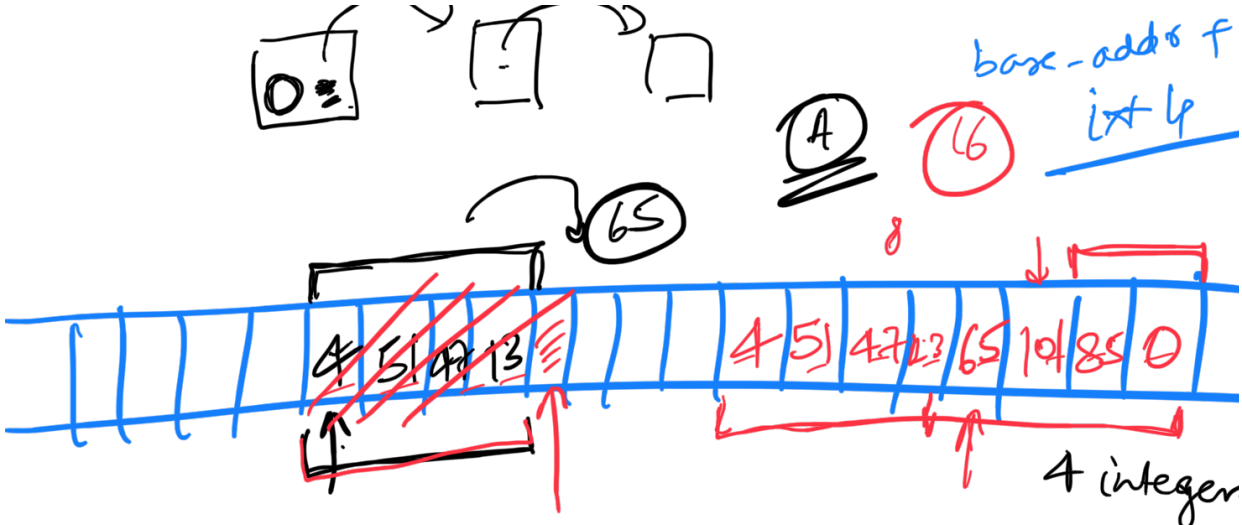


static array → size of the array.

① Quick random access

A[i]

② Size is not fixed

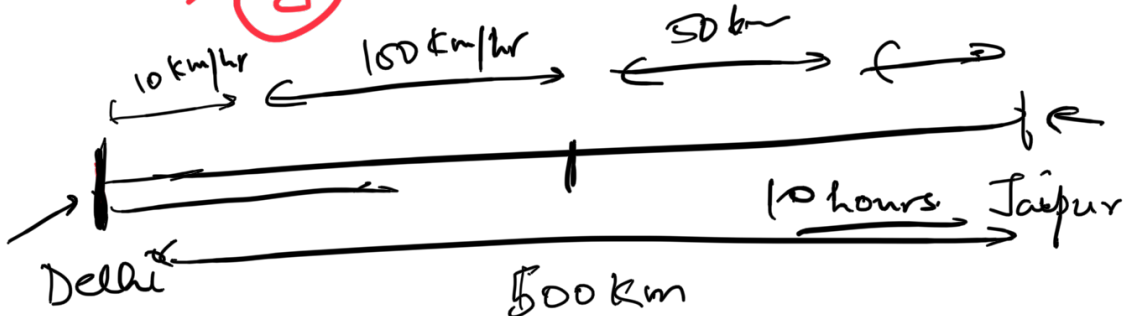


in

1 ← 1
2 ← 1
3 ← 1
4 ← 1
5 ← 4 + 1
6 ← 1
7 ← 1
8 ← 1

9

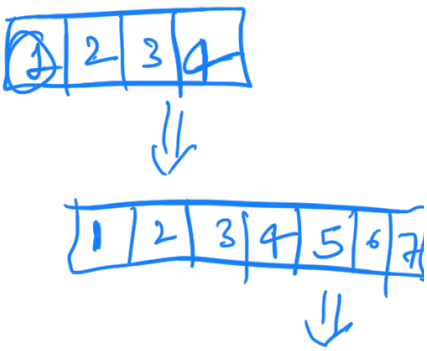
A → 4
51
47
13
65
101
85
0
101

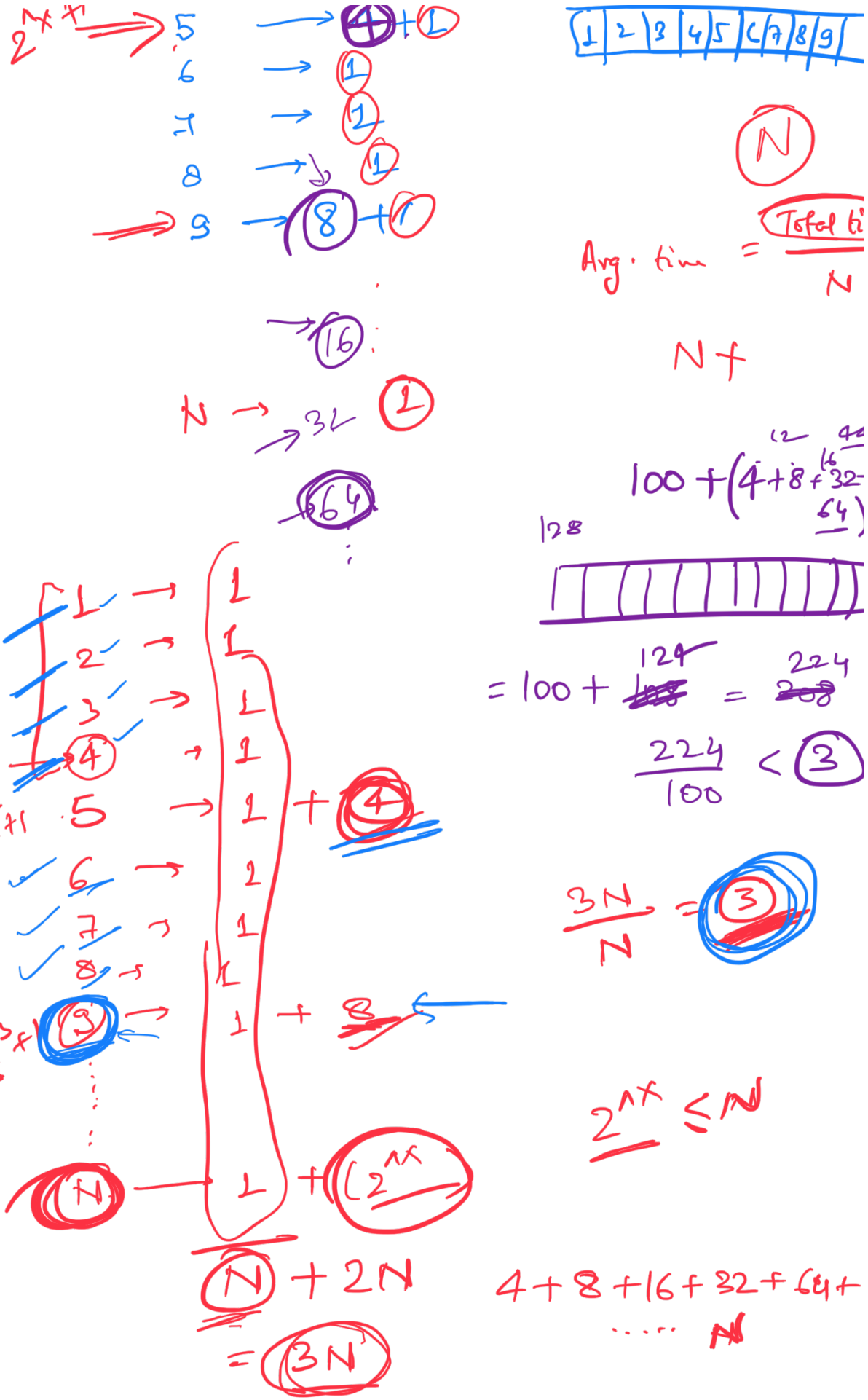


50 km/hr

N elements in dynamic array

1 → 1
2 → 2
3 → 2
4 → 2




2^x = x
5
6
7
8
9
+1
1
1
1
8
+1
16
N → 32 1
64
1 2 3 4 5 6 7 8 9
N
Avg. time = Total ti / N
N +
100 + (4 + 8 + 16 + 32 + 64)
128
= 100 + 124 = 224
224/100 < 3
3N/N = 3
2^x ≤ N
1 2 3 4 5 6 7 8 9
1 1 1 1 1 1 1 1 1
+ 4
+ 8
+ (2^x)
N + 2N
= 3N
4 + 8 + 16 + 32 + 64 + ... N

$= N + \frac{N}{2} + \frac{N}{4} + \dots 64 + 32 + 16 + 8 + \dots$

$= N\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots\right)$

$= 2$

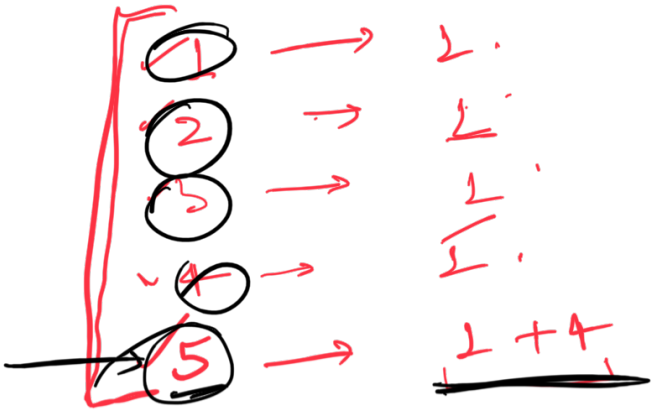
$\leq 2N$

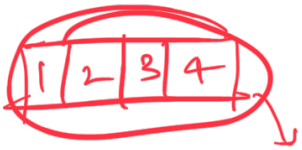
$A[i] = 1$

Avg cost of 1 insertion $=$ Total cost / Total inser

A.append(x) ← avg. cos

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





$\frac{1}{1} = 1$

$\frac{2}{2} = 1$

$\frac{3}{3} = 1$

$\frac{4}{4} = 1$

$\frac{9}{5} \simeq 2$