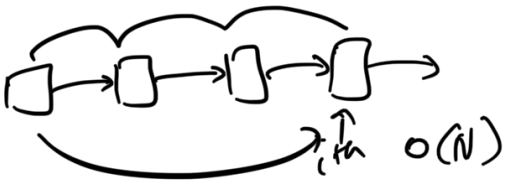
Arrays & dynamic Arrays

↳ Collection of same type of data

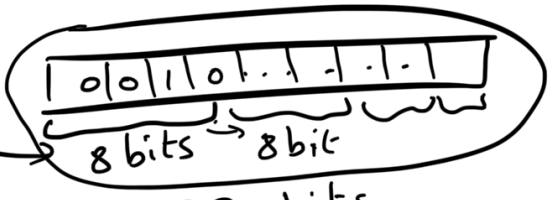
eg. heights of student / names etc.

Why array → Random access / O(1) lookup time.

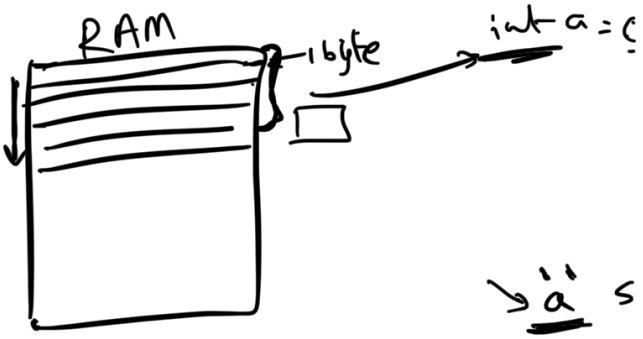
eg in Linked →



int → 4 bytes



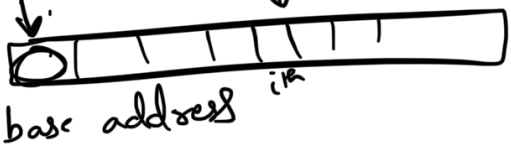
32 bits



int a [n]

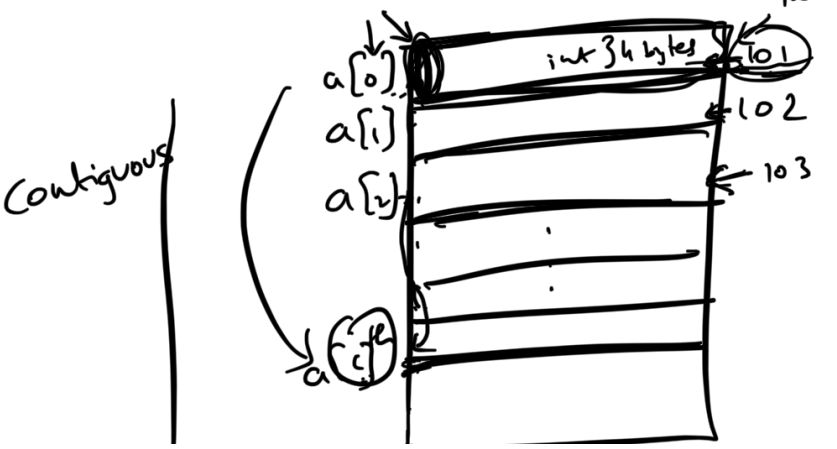
↳ 'a' stores address of the first element

a[i] *a = a[0] (pointers)



hexadecimal

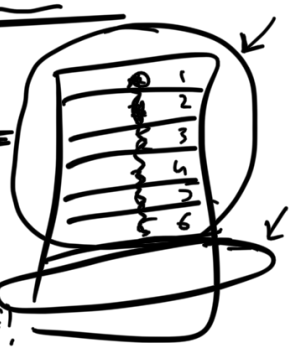
↓ int



O(1)

$a[i] = a + i * (\text{size of object})$

int a[6]



(we can't change the size)

Assign ⇒ $a[i] = 12$ ⇒ $O(1)$

Q. Given array of N integers, first (n-1) integers are sorted. How much time to sort entire arrays.

Google phone

45%

eg A = [1 | 2 | 7 | 14 | 20 | 21 | 27 | 11]

N-1 sorted

Binary search → position → $O(\log N)$

To shift → $O(N)$

Sub routine insertion sort

T.C. $O(N)$

[1 2 7 14 20 21 27 11]

1 2 7 11 20 21 27 14

Q.2 Given array, reverse the array.

A [1 2 3 4]

random array

[4 3 2 1]

Brute → Create a new array → S.C ⇒ O(N) extra space

b[4 3 2 1] ← N

T.C ⇒ O(N)

2 pointers swapping elements into correct place.

1 2 0 0 7 5

5 7 2 1

$i < N/2$

```
for( i=0 ; i < N ; i++)
{
    temp = a[i]
    a[i] = a[n-i-1]
    a[n-i-1] = temp
}
```

→ original array.

XOR

T.C ⇒ O(N)

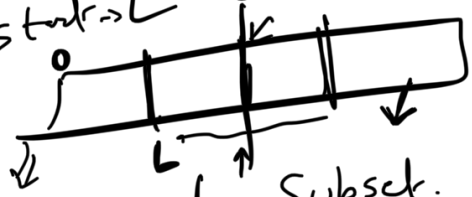
S.C ⇒ O(1)

2 0 diff

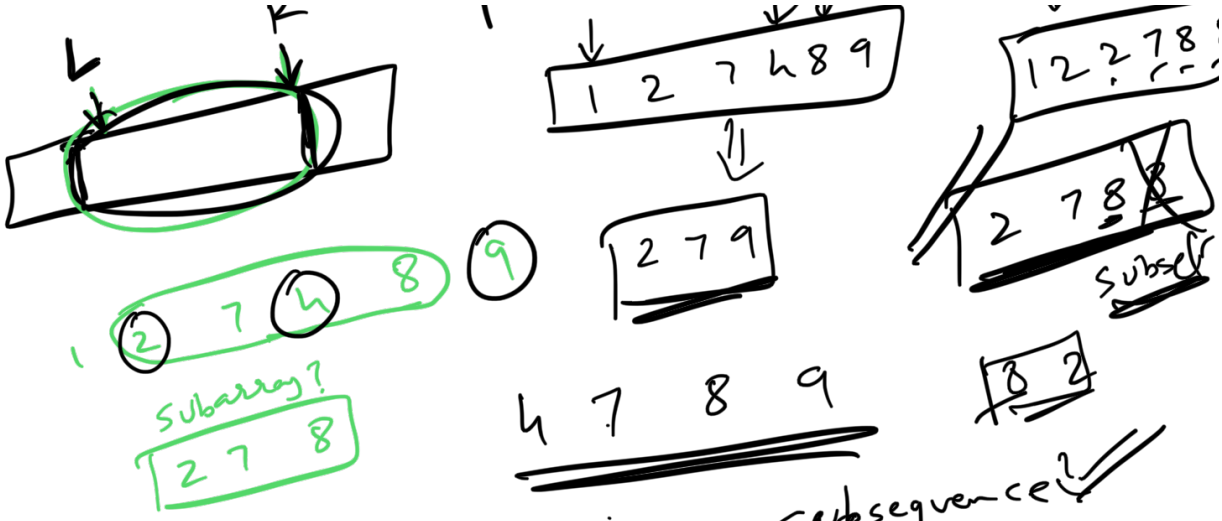
$O(N/2)$ $N/2$ $O(N/2)$

$\left(\frac{L+R}{2}\right)$

Reverse a sub-array, start → L, end → R

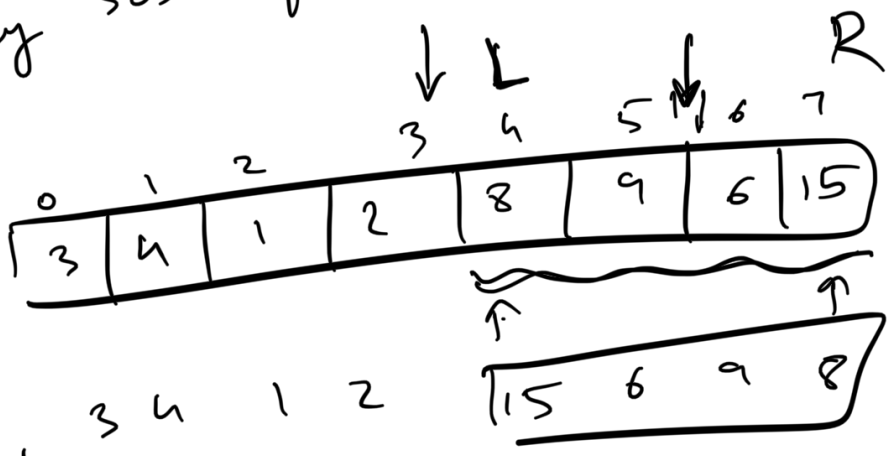


Sub-array	Subsequence	Subset.
Contiguous slice from the array (same order)	This can be derived after deleting some elements (same order)	Pick any elements (distinct)



Every sub-array is a subsequence? ✓

Every sub-seque is a subset ✗



$L = 4$

$R = 7$

Q.3 Given array, rotate array by K steps (right rota

$K = 3$

$A = [3\ 2\ 9\ 6\ 5\ 8\ 1]$

$A = [5\ 8\ 1\ 3\ 2\ 9\ 6]$

Try this $O(1)$ extra space

element → K times.

Brute: → shift 1 elem

3 2 9 6 5 8 1 — 1 rotation → O(N)

1 3 2 9 6 5 8

T.C ⇒ O(N*K)

O(N-K) | K=3 | O(K)

3 2 9 6 | 5 8 1

5 8 1 3 2 9 6

reverse 2 blocks ⇒ O(N)

6 9 2 3 1 8 5

reverse entire O(1

5 8 1 3 2 9 6

S.C ⇒ O(1)

T.C ⇒ O(N)

%

K

K

n-k

0 — N-K (N-K+1 K=3 N)

3 2 9 6 5 8 1

N-K

6 9 2 3 1 8 5

5 8 1 3 2 9 6

N

exe

T.C →

Limitations of array! → runtime modifica
↳ size is fixed/st

When do we use array → size of known/fixed

eg. when do you perform exercise

exercised [366] ⇒

Realistic estimate of [upper limit] maximum size.

Run in ODI

ViratRuns [ ]

500, 700, [1000]

47	108	3	.....

Max → Sachin → 463
started at 16

eg. details of users watching youtube ch

names [ ] → size → 100 views, 1K views, 1 Mil.

10

for viral video → billions →

If same space is used, space is wasted.

How to scale

Dynamic arrays → space no demand

vectors in c++ → push_back..
ArrayList in → java ← add()..
List → python.. etc

How do these work?

Look-up → O(1) → random access.

↳ flexible in size.

Asked google, FB

O(1) acces → use array (contiguous)

Obs: Create a new array of increased size by 1 each, & copy existing into the new array.

How to calculate T.C.?

T.C. of inserting N elements

	operations	
[ ]		
a.push(1) →	1 →	[1]
a.push(2) →	1 + 1 = 2	[1 2]
a.push(3) →	3	[1 2 3]
⋮		
a.push(i) →	i	[1 2 3 ... i]
⋮		[1 2 3 ... n]

a.push (n) → $\{n-1\} + 1$

copying → assign

Total time for all insert

$= 1 + 2 + 3 + \dots n$

$= \frac{n(n+1)}{2} = \frac{n^2+n}{2}$

$= O(n^2)$.

Inserting → $O(N^2)$ ← bad time.

Access → $O(1)$

Req → $O(N)$ time. for inserting N elements

Instead of increasing size by 1, what if we double the size.

		operations.	
	[ ]		
a.size	a.push (1)	→ 1	[1]
	a.push (2)	→ 1 + 1	[1 2]
a[5]	a.push (3)	→ 2 + 1	[1 2 3 _]
	a.push (4)	→ 1	[1 2 3 4]
int size	a.push (5)	→ 4 + 1	[1 2 3 4 5 0 0]

a[7]

throw out

a.push (6) → 1

(7) → 1

(8) → 1

(9) → 8 + 1

10 → 1

16 → 1

17 → 16 + 1

18 → 1

32 → 1

39

a.size

O(1)

[ ]
16

[ ]
32

Total Time → $(1 + 1 + 1 + \dots N)$ ← inserting element → N

$+$

copying → $(1 + 2 + 4 + 8 + \dots N)$

$2^0 \quad 2^1 \quad 2^2 \quad 2^3 \quad \dots \quad 2^i$

$i = \log_2 N$

GP

$$\frac{a(r^n - 1)}{r - 1}$$

$2^i \leq N$

$\log 2^i < \log$

$i < \log$

a → starting → 1

r → multiplier → 2

n → log N

$$\rightarrow \frac{1(2^{\log_2 N} - 1)}{2 - 1}$$

$2^{\log_2 N} \rightarrow N$

$$\rightarrow \frac{N - 1}{1}$$

$$\rightarrow N - 1$$

T.C ⇒ O(N+N) ⇒ O(N)

Inserting 1 element ⇒ O(1)

Amortised O(1) for each

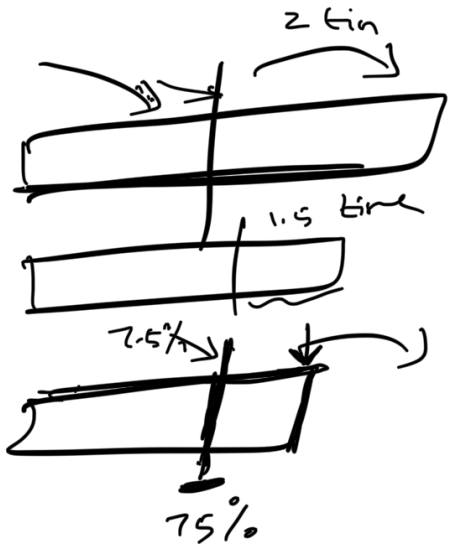
2 2.1 2.2

2.1

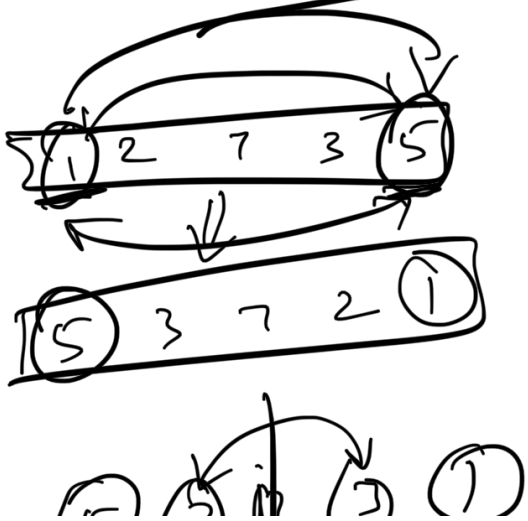
Java

load factor

1.3

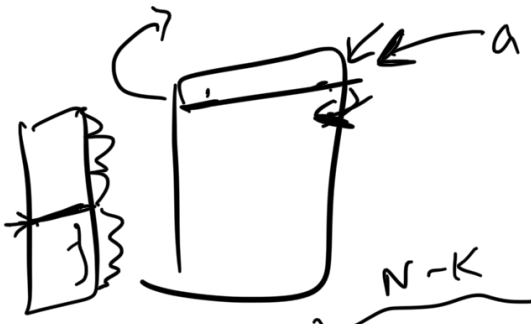


Doubts



O(1) extra space

suhas



$a[0] = a * 4\text{byte} \times 0$
$= a$

$a[1] = a + 4 \times 1$

N-K K=3

3 9 8 5 1 6 2

16 2 3 9 8 5

~~1 6 2 5 3 9 8~~

K

3 9 8 5 1 6 2

n/2

n/2

3 9 8 3 9 8 5

n-1

O(N)

N*K O(N²)

1 7 4 9 8 3 8 5 6 7

5 6 7 9 8 3 8 7 4

1 7 6 8 9 8 3

$N * \log N$

Binar $\rightarrow$