Note Title 08-10-2018

COL100

Oct 8, 2018

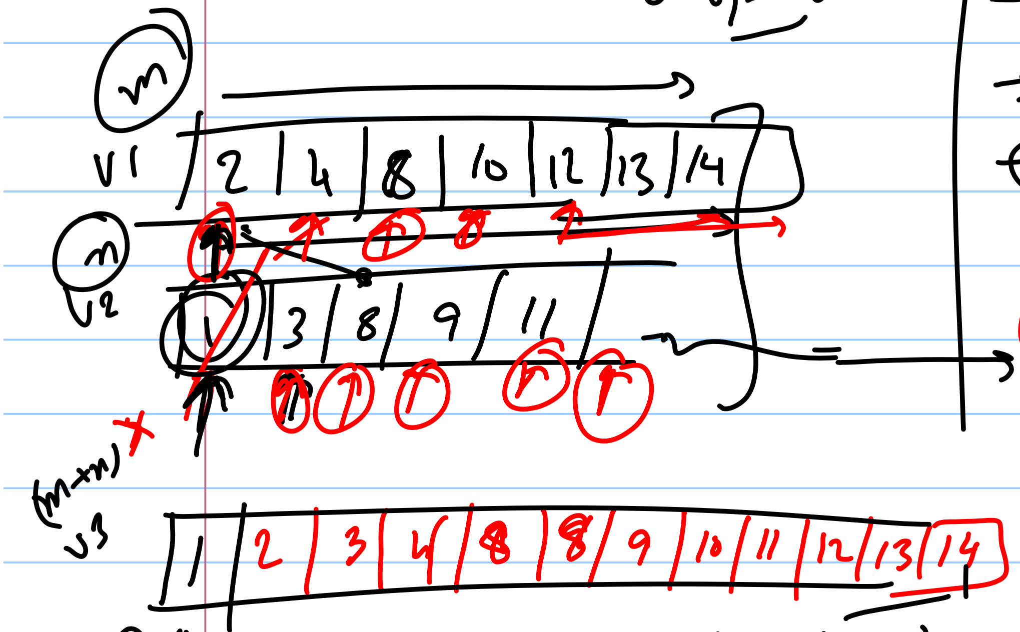
(Tentative) Plan for Rest

→ Complexity

→ Stacks/Queues/Maps

→ Recursion

Problem solving



①

```
int index1;
int index2;
int index3;
int size1 = v1.size();
int size2 = v2.size();
while(index1 < size1
   ||index2 < size2
```

Efficient

① $(n \times m) + (m+n) = O(mn)$

② $O(m+n)$ ✓ — Order Notation

an^2+bn+c $O(n^2)$

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

for (j=1; j<=i; j++){

~~print~~ cout<<"hello"; $O(n^2)$

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

$2\ O(n^2)$

}

for (int k=1; k=n; k++) {

cout<< " ";

}

$O(n^2+n)$

$= O(n^2)$

big-O

```
for(int x = N; x >= 0; x=x-4;){
        v.add(x);
}
```

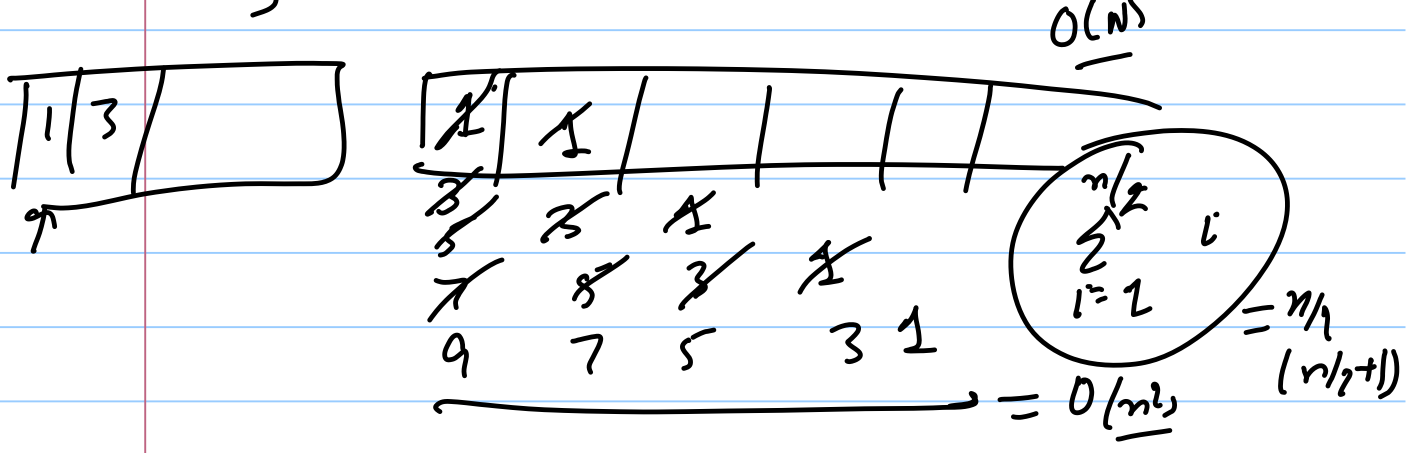
N=10

```
Vector<int> v
for(int x= 1; x<=N; x=x+2) {
    v.insert(0,x);
    cout << "hello";
```

N/2

O(N)

3



Input: N

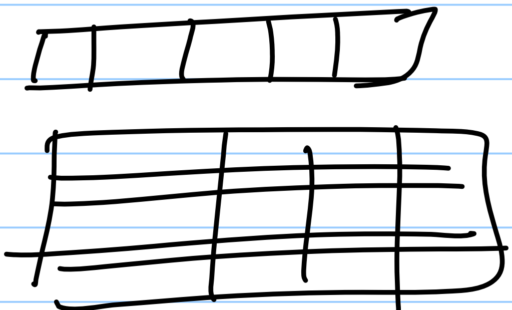
$N! \geq (N/2)^{N/2}$

	Complexity		If you double N, Time $N \to 2N$
1	Constant	$O(1)$	remain Same
2.	Logarithmic	$O(\log_2 N)$	Add constant (1)
3.	Linear	$O(N)$	Doubles
4.	log x linear	$O(n \log n)$	slightly more than double
5.	Quadratic	$O(n^2)$	four times
6.	log x quadratic	$O(n^2 \log n)$	slightly more than 4 times
7	Cubic	$O(n^3)$	8 times the complexity
8	Exponential	$O(2^n)$	grows extremely fast
9.	Factorial	$O(N!)$	grows extreme fast

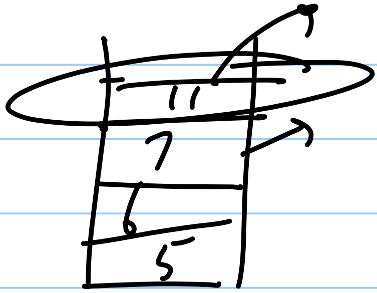
Stacks / Queues: -

ADT   Abstract Data Type

(1) Vectors
(2) Grids



(3)



← "First in Last out"

→ Queue