Radix Sort :→ Bucket Sort   1's, 10's, 100's and so on.

* Works for multi-digit numbers
* Also for constant length strings   pavan, rahul, etc
* Also a non-comparison algo

325	236	008	092	041
041	092	325	236	008	→ ①
008	325	236	041	092	→ ②
008	041	092	236	325	→ ③

countSort ↵

Step → 1 Find the max → 325
digits = 3, 3 passes ✓
Step → 2 Create 0 – 9, 10 buckets

$O(n + max)$

0	1	2	3	4	5	6	7	8	9
→	041	092			325	236		008	
008		325	236	041					092
008, 041, 092		236	325						

[Wipro, IBM, TCS, Tackle Box, Capgemini]   H/W

* How are the number of iterations controlled in the radix sort algorithm?   max → 325 ③ passes

1, 2, 3 → stop

```
for (int exp = 1; max/exp > 0; exp *= 10)
{
    countSort (arr, exp);
}
```

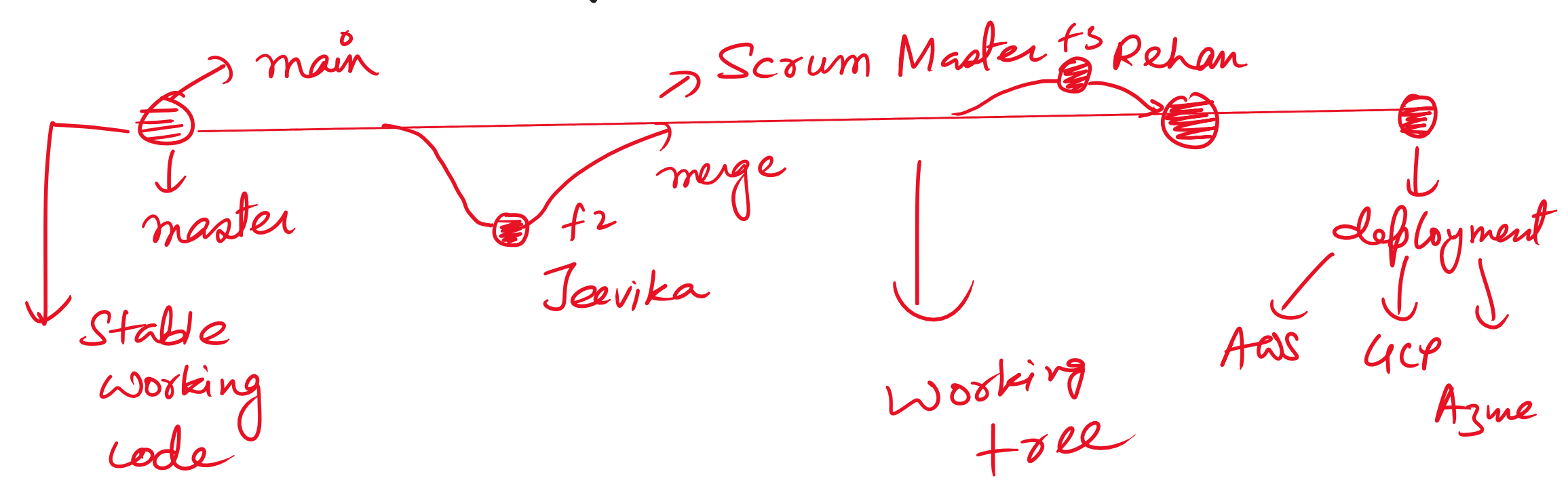
max = 325 / 1 = 325
exp = 1×10 = 10 ×10 = 100
325 / 10 = 32
325 / 100 = 3
325 / 1000 = 0

Developer → [laptop]
App → website
index.html
script.js
styles.css
login.java
auto.py
not safe
Red
[Untracked Area / Local Area] → Tracking Area / Staging Area

Version Control Tool
Git / Mercurial
Empty Repository
git init   Green
git add <file>
git add .
git commit -m "message"
(Check sum) xx

Cloud
Github

DDL
DQL
DML
TCL
↳ commit
rollback
Savepoint
access
revoke



git remote add origin <github_url>

Searching Algorithms :→

* We search for a particular key in an array, if found, the index at which the element is present, is returned or else we return an invalid index → generally (–1).
* Linear Search → Simplest Searching Algorithm

int[ ] arr = { 2, 8, 9, 6, 4, 3, 0 };   (indices 0 1 2 3 4 5 6)
int key = 3.   O/P → index = 5

We traverse the array from 0^{th} index till the last index (length – 1).
Time Complexity → n – elements so $O(n)$. [Worst case]

DSA **Percentage Break-up of Topics in Competitive Coding Rounds**

CS / IS

Binary Search	Recursion	Graphs + Trees	Arrays, Strings, LL, Greedy, Bits
30%	20%	25%	25%

Recursion ↑ Dynamic Programming, ↓ Back Tracking
Stacks / Queues

Core Subjects → CN, OS, OSI Model, DBMS
SDLC
* CI – CD ↑ Dev Ops + Cloud Principles → IaaS, PaaS, SaaS
Development HTML, CSS, JS

Binary Search :→ Pre-requisite (Sorted Array)   s = m+1

$m = \frac{s+e}{2} = \frac{3+5}{2} = \frac{8}{2} = 4$
Key = 6

2	6	9	12	15	19
0	1	2	3	4	5
s, l	e	m	s'		e, h

arr[0] = 2 ≠ 6   key = 15
$mid = \frac{s+e}{2} = \frac{0+5}{2} = 2$
arr[2]
9 == 15
9 < 15

m = 2 ① if (arr[mid] == key) {   arr[4] 15 == 15
return mid; →
}
arr[2]
9 == 6
9 < 6
9 > 6
else if (arr[mid] < key)
s = m + 1;   (go to the rightside)
s = 0, e = 1
$\frac{0+1}{2} = \frac{1}{2} = 0$
else (arr[mid] > key) {
e = m – 1;
}
arr[1] 6 == 6 $\frac{1+1}{2} = \frac{2}{2} = 1$

N $\frac{N}{2^0}$ ✓   ← N →
$\frac{N}{2}$ $\frac{N}{2^1}$ ✓   ← $\frac{N}{2}$ →   ← $\frac{N}{2}$ →
$\frac{N}{4}$ $\frac{N}{2^2}$ ✓ $\frac{N}{4}$ $\frac{N}{4}$
$\frac{N}{8}$ $\frac{N}{2^3}$ ✓ $\frac{N}{8}$ $\frac{N}{8}$
$\frac{N}{2^K}$

s
[x]
e
O
s > e
s —— e

K = 0, 1, 2, 3 → constant
$\log_2 N = K$
Big O [log N]

$[mid = \frac{s+e}{2}]$ {Max + 1 = Min
Min – 1 = Max}
INT_MAX → $2^{31} - 1$
INT_MIN → -2^{31}

s, e are integer values.

$mid = \frac{s^{MAX} + e^{MAX}}{2}$ (+)ve (–)ve → OOB

Q What is the Optimal formula for mid?
$m = \left[s + \frac{(e - s)}{2}\right] = \frac{2s + e - s}{2} = \frac{s+e}{2}$

n = 2, 5, 9, 8, 16, 19, 17, 22, 32, 64
(2^1, ×, ×, 2^3, 2^4, ×, ×, ×, 2^5, 2^6)

How do you check whether a given integer 'n' is a power of 2 or not?

TCS → 2024 → BIET

n & (n – 1) == 0
O(1)

n = 8 – 1000
7 – 0111
0000

4 – 0100
3 – 0011
0000

5 – 0101
4 – 0100
0100
n & (n–1)