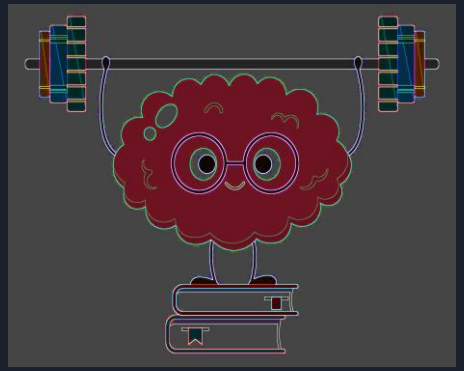
INTRO TO COMPETITIVE PROGRAMMING

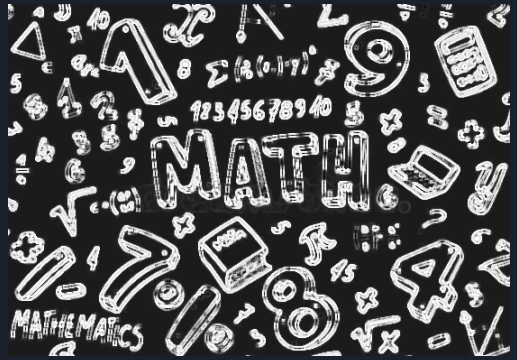
Bharat Singla

Yi Heng

Competitive Programming

Programming with a flavour of competitiveness!

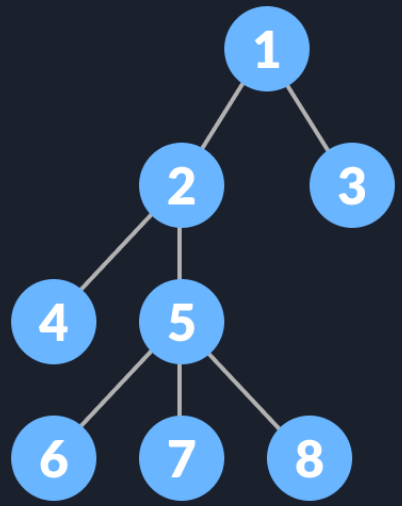




Competitive Programming

Competitive Programming

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100



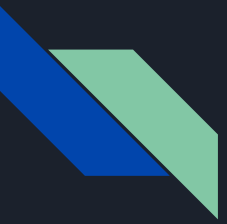
100 Doors!

- There are 100 doors, initially all closed
- The 1st person opens all doors
- 2nd person closes all doors in multiples of 2 i.e. 2^{nd}, 4^{th}, 6^{th}... 100^{th} door
- 3rd person toggles all doors in multiples of 3
- and so on...


0
1
2
3

Stone Game!

- ➔ Alice and Bob play a game by taking turns
- ➔ They start with a pile of *N* stones
- ➔ In each move, the player can remove 1 or 2 stones
- ➔ The person to remove the last stone wins the game



N = 1: Win

N = 2: Win

N = 3: Lose

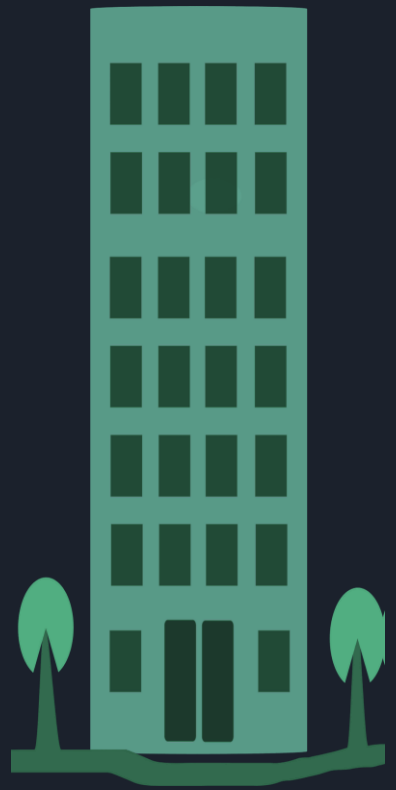
N = 4: Win

N = 5: Win

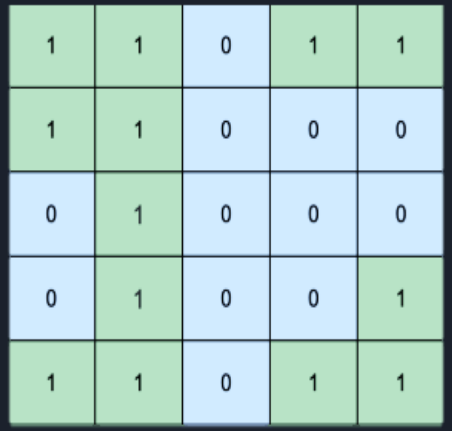
N = 6: Lose

Egg Drop!

- There is a 100 floors tall skyscraper
- You must determine the highest floor from which an egg can be dropped without breaking in the least tries
- If the egg breaks from a floor, it will also break from all floors higher



Some Other Problems!



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

2	6	7	5	4	9	3	1	8

Time Complexity

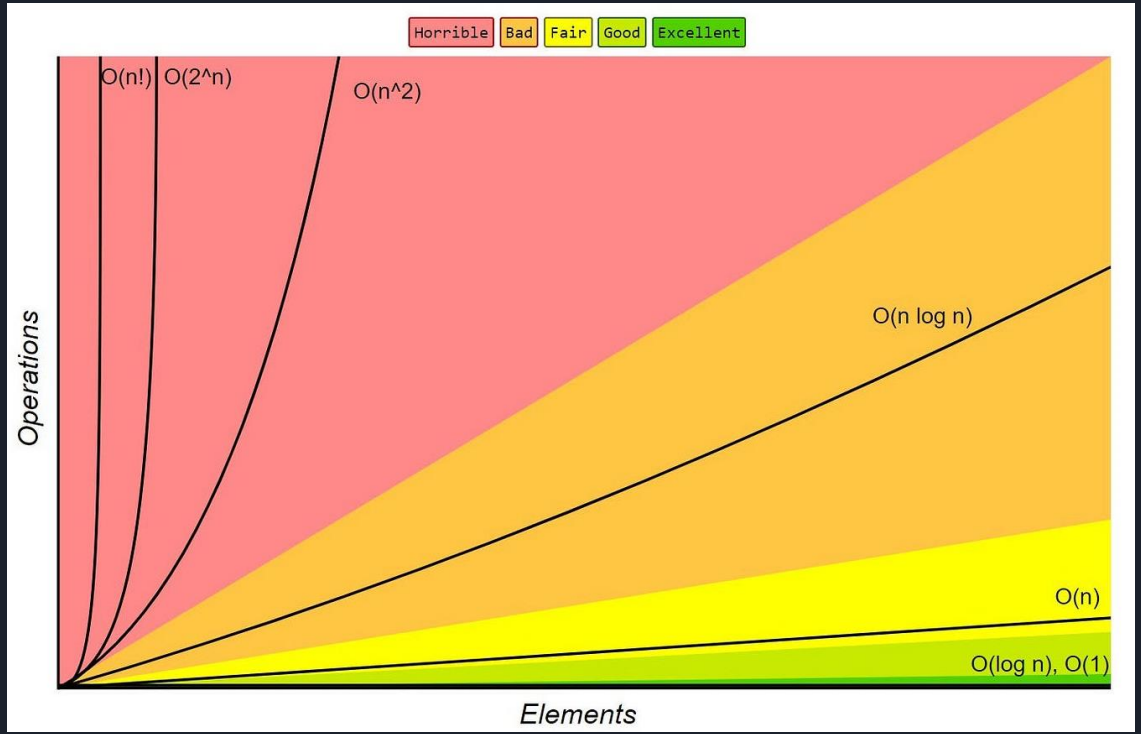
Technique to measure algorithmic efficiency

- ➔ Compare various approaches to solve problems
- ➔ Estimate run-time of a code

Big O

HackerRank

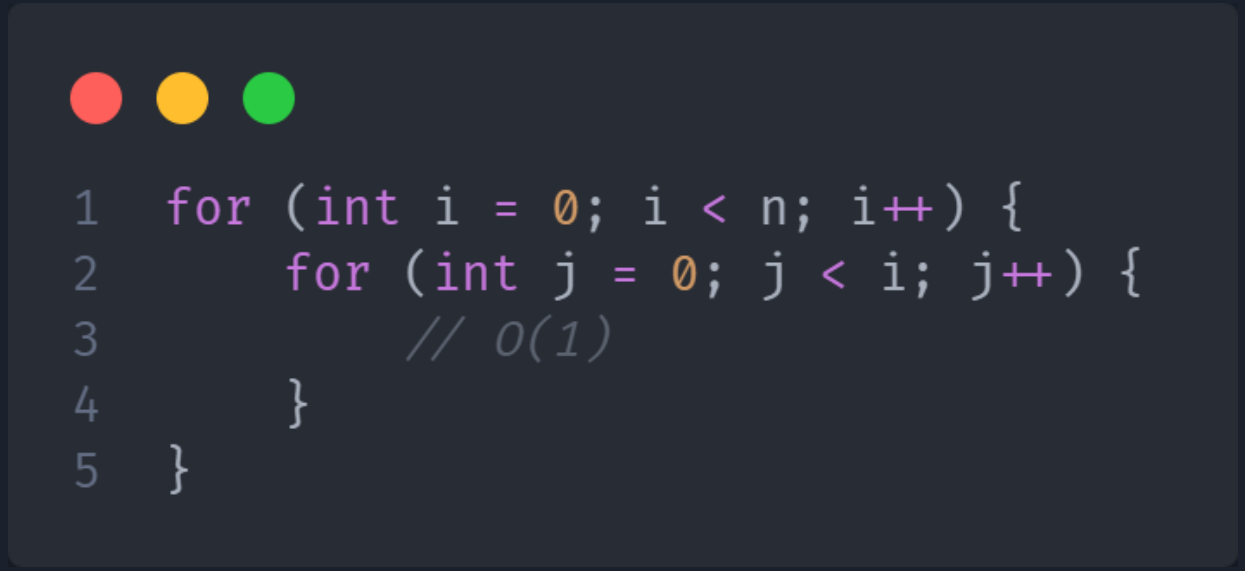
Time Complexity



Time Complexity

Input size	Expected time complexity
$n \leq 10$	$\mathcal{O}(n!)$
$n \leq 20$	$\mathcal{O}(2^n)$
$n \leq 500$	$\mathcal{O}(n^3)$
$n \leq 5000$	$\mathcal{O}(n^2)$
$n \leq 10^6$	$\mathcal{O}(nlogn)$ or $\mathcal{O}(n)$
n is large	$\mathcal{O}(logn)$ or $\mathcal{O}(1)$

Time Complexity



```
1  for (int i = 0; i < n; i++) {
2      for (int j = 0; j < i; j++) {
3          // O(1)
4      }
5  }
```

Time Complexity

```
1  for (int i = 0; i < n; i++) {
2      for (int j = 0; j < i; j++) {
3          for (int k = 0; k < j; k++) {
4              // O(1)
5          }
6      }
7  }
```

Time Complexity

```
1  int sum = 0
2  for (int i = 1; sum < n; i++) {
3      sum += i;
4  }
```

Time Complexity

```
1  for (int i = 0; i < n; i++) {
2      for (int j = 0; j < n; j += i) {
3          // O(1)
4      }
5  }
```

Time Complexity

```
1  // sorted a
2  int l = 0, r = n - 1;
3  while (l < r) {
4      int sum = a[l] + a[r];
5      if (sum == target) {
6          break;
7      } else if (sum < target) {
8          l++;
9      } else {
10         r--;
11     }
12 }
```

That’s all folks!