Power finding

$2^n = 2^{n/2} \cdot 2^{n/2}$

$a = 2, b = 64$

$2^{64} = 2^{32} \cdot 2^{32}$

2^{64}

2^{32} 2^{32}

2^{16} 2^{16}

$256 * 256 =$

2^8 2^8

$16 * 16 = 256$

2^4 2^4

$4 * 4 = 16$

2^2 2^2

$2 * 2 = 4$

2^1 2^1

$a = 2, b = 65$

$2^{65} = 2 * 2^{64}$

$a *$

Recursion

$2^{67} = 2 * 2^{66}$

2^{66}

2^{33} 2^{33}

2×2^{32}

BigInteger

Binary Search

$8 < 10$

0	1	2	3	4
2	4	8	10	12

target = 10

Output = 3

Recursion

low = 0, high = n - 1
= 4

T(n)

function name

mid = 2

binarySearch (arr, low, high, target) {

while (low <= high) {

Base case condition

if (arr (mid) == target)

1 C {

return mid;

}

2 else if (arr (mid) < target)

{ Right

Recursion binarySearch(arr, mid + 1, low

high, target)

T(n/2)

}

else

Left T(n/2)

3 binarySearch(arr, low,
mid - 1, target)

0	1	2	3	4	5
2	4	6	8	10	12

low = 0

high = 5

target = 10

mid = 2

Example 1

binarySearch(arr, 3, 5, 10)

mid = 4

result = 4

0	1	2	3	4	5
2	4	6	8	10	12

low = 0

high = 5

target = 2

mid = 2

Example 2

binarysearch(arr, 0, 1, 2)

2,

mid = 1

binarysearch(arr, 0, 0, 2)

→ mid = 0

→ return mid

output = 0

Binary search

$$T(n) = T(n/2) + c \quad \text{--- } 1$$

$$T(n/2) = T(n/2^2) + c$$

$$T(n) = T\left(\frac{n}{2^2}\right) + 2c$$

$$T(n) = T(n/2^3) + 3c$$

} K times

$$T(1) = 1$$

$$T(n) = T\left(\frac{n}{2^k}\right) + 3k$$

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$\log_2 n = k \cancel{\log_2 2}$$

$$k = \log_2 n$$

$$T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + (\log_2 n) * c$$

$$= T\left(\frac{\cancel{n}\ 1}{\cancel{n^{\log_2 2}}\ 1}\right) + c \cdot \log_2 n$$

$$a^{\log_2 b} = b^{\log_2 a}$$

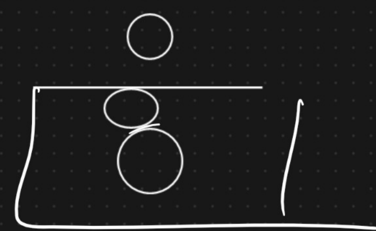
$$\log_a a = 1$$

$$= 1 + c \cdot \log_2 n$$

$$= O(\log_2 n)$$

Searching

$n = 1$



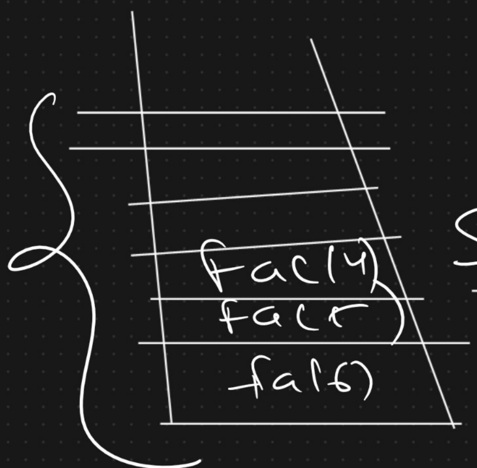
target = 9

```
if ( arr(0) == target )
{
    return 0;
}
return -1;
```

c

Space complexity

function call



Stack = LIFO

Last In first Out

n times

log n times

n — 10

$Num = 12345$

sum of digit =

$$\underbrace{1+2+3+4+5}_{} = 15$$

972

4 $\quad 9+7+2$

18

SumDigits(num){

Base case condition

{ $num = 0$

↳ 0

$num = 12345$

else {

$(num \% 10) + \underline{sumDigits(\underline{num/10})};$

r

$12345 \; \underline{\%} \; \overset{\text{remainder}}{10} = 5$

fun(1234)

↓

4 + fun(123)

fun(12345) — 15

10

5 + fun(1234)

6

4 + fun(123)

3

3 + fun(12)

Assignment

1

2 + fun(1)

0

1 + fun(0)

Problem Statement

num = 4   k = 5

4, 8, 12, 16, 20

num = 12   k = 4

12, 24, 36, 48

result = 12

num = 12   k = 1

k == 1

return num

o

printMult (n, k){

1 if(k==1){

print n;

}

2 printMult (n, k-1)

3 print(n * k)

}

60

printMult(12, 5)

48

printMult (12, 4)

36

printMult(12, 3)

24

printMult(12, 2)

12

printMult(12, 1)