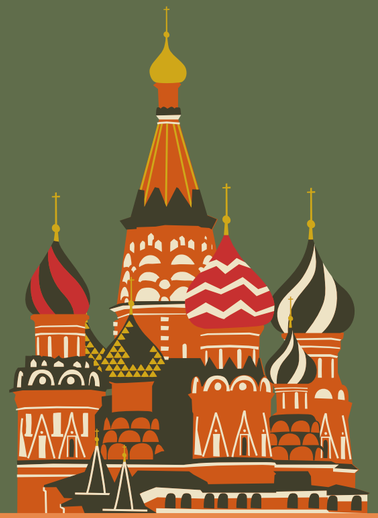
MULTIPLICATION A LA RUSS



DESIGN AND ANALYSIS OF ALGORITHM

64 x 32

64 x 32

64 x 32

Half

64 x 32

Half

64 x 32

Half

Double

$$2 \times 4 = 8$$

$2 \times 4 = 8$

$1 \times 8 = 8$

balanced

64 x 32

64 x 32

32 x 64

64 x 32

32 x 64

16 x 128

64 x 32

32 x 64

16 x 128

8 x 256

64 x 32

32 x 64

16 x 128

8 x 256

4 x 512

64 x 32

32 x 64

16 x 128

8 x 256

4 x 512

2 x 1024

64 x 32

32 x 64

16 x 128

8 x 256

4 x 512

2 x 1024

1 x 2048

64 x 32

32 x 64

16 x 128

8 x 256

4 x 512

2 x 1024

1 x 2048

$$64 \times 32$$

$$=$$

$$1 \times 2048$$

26 X 14

26 X 14

13 x 28

odd

13 X 28

12 +1

26 X 14

13 x 28

28

26 X 14

13 x 28 28

6 x 56

3 x 112 112

1 x 224 224

26 X 14

13 x 28	28
6 x 56	+
3 x 112	112
	+
1 x 224	224

26 X 14

13 x 28	28
6 x 56	+
3 x 112	112
	+
1 x 224	224

= 364

How to implement it ?

recursively

26 X 14

assign var a as the first num of the eq

13 x 28 28

6 x 56 +

3 x 112 112

1 x 224 +

224

= 364

assign var b as the second num of the eq

26 X 14

assign var a as the first num of the eq

13 x 28 28

6 x 56 +

3 x 112 112

+

1 x 224 224

= 364

assign var b as the second num of the eq

26 X 14

assign var a as the first num of the eq

13 x 28 28

6 x 56 +

3 x 112 112

+

1 x 224 224

= 364

Our base case is when a reaches 1

assign var b as the second num of the eq

26 X 14

assign var a as the first num of the eq

13 x 28	28
6 x 56	+
3 x 112	112
	+
1 x 224	224

= 364

Our base case is when a reaches 1

result is calculated by Halving (a) and Doubling (b)
if a odd -> add b to result

```
def russian_multiply(a, b):
    if a == 1:
        return b
    elif a % 2 == 0:
        return russian_multiply(a // 2, b * 2)
    else:
        return b + russian_multiply(a // 2, b * 2)
```

By decreasing the problem into smaller problems

this is called

Decrease and Conquer

(by half)

russian_multiplay(26 X 14)

```
def russian_multiply(a, b):
    if a == 1:
        return b
    elif a % 2 == 0:
        return russian_multiply(a // 2, b * 2)
    else:
        return b + russian_multiply(a // 2, b * 2)
```

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

28 + russian_multiplay(6 X 56)

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

28 + russian_multiplay(6 X 56)

russian_multiplay(3 X 112)

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

28 + russian_multiplay(6 X 56)

russian_multiplay(3 X 112)

112 + russian_multiplay(1 X 224)

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

28 + russian_multiplay(6 X 56)

russian_multiplay(3 X 112)

112 + 224

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

28 + russian_multiplay(6 X 56)

336

112 + 224

russian_multiplay(26 X 14)

russian_multiplay(13 X 28)

28 + 336

336

112 + 224

russian_multiplay(26 X 14)

364

28 + 336

336

112 + 224

russian_multiplay(26 X 14)

364

Time complexity

```
def russian_multiply(a, b):
    if a == 1:
        return b
    elif a % 2 == 0:
        return russian_multiply(a // 2, b * 2)
    else:
        return b + russian_multiply(a // 2, b * 2)
```

using recurrence relation

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$T(a/2) = T(a/4)+1$$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$= T(a/4)+1+1$$

$$T(a/2) = T(a/4)+1$$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$= T(a/4)+1+1$$

$$T(a/2) = T(a/4)+1$$

$$T(a/4) = T(a/8)+1$$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$= T(a/4)+1+1$$

$$= T(a/8)+1+1+1$$

$$T(a/2) = T(a/4)+1$$

$$T(a/4) = T(a/8)+1$$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$= T(a/4)+1+1$$

$$= T(a/8)+1+1+1$$

Genrlize

$$= T(a/2^k)+k$$

$$T(a/2) = T(a/4)+1$$

$$T(a/4) = T(a/8)+1$$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$= T(a/4)+1+1$$

$$= T(a/8)+1+1+1$$

Genrlize

$$= T(a/2^k)+k$$

$$T(a/2) = T(a/4)+1$$

$$T(a/4) = T(a/8)+1$$

assume $2^k = a$

$k = \log(a)$

$$T(a) = T(a/2)+1,\ T(1) = 0$$

$$= T(a/4)+1+1$$

$$= T(a/8)+1+1+1$$

Genrlize

$$= T(a/2^k)+k$$

substitute

$$= T(a/2^{\log(a)})+\log(a)$$

$$T(a/2) = T(a/4)+1$$

$$T(a/4) = T(a/8)+1$$

assume $2^k = a$

$k = \log(a)$

T(a)

= T(a/2^log(a))+log(a)

= T(1)+log(a)

but , T(1)=0

= log(a)

Time complexity

$$= T(a/2^{\log(a)})+\log(a)$$

$$= T(1)+\log(a)$$

but , $T(1)=0$

$$O(\log(a))$$

THANK YOU FOR WATCHING!

BY FAYROUZ MOHAMED