Decimal Number System (Base = 10)

Digits allowed - 0 1 2 3 4 5 6 7 8 9

No. of Digits = 10

first →	Second pos. →
0	10
1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19

Decimal Number System.

20
21
22
23
24
25
26
27
28
29

Binary System (Base - 2)

No. of Digit = 2 , 0 , 1

Dec	Bin.
0 →	0
1 →	1
2 →	10
3 →	11
4 →	100
5 →	101
6 →	110
7 →	111
8 →	1000
9 →	1001
10 →	1010
11 →	1011
12 →	1100
13 →	1101
14 →	1110
15 →	1111

Octal Number (Base : 8)

0, 1, 2, 3, 4, 5, 6, 7 } → No. of Digits = (8)

	First					
0 -	0	8 - 10	16 - 20		70	100
1 -	1	9 → 11	17 - 21		71	
2 -	2	10 → 12	18 - 22		72	
3 -	3	11 → 13	19 - 23		73	
4 -	4	12 → 14	20 - 24		74	
5 -	5	13 → 15	21 - 25		75	
6 -	6	14 → 16	22 - 26		76	
7 -	7	15 - 17	23 - 27		77	

Hexadecimal → Base - 16

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F } 16 digit

Binary to Decimal →

number = $(1\ 1\ 1)_2$

$2^2 \times 1 \quad 2^1 \times 1 \quad 2^0 \times 1$

$4 + 2 + 1 = 7$

$(1\ 1\ 1)$ Base = 2

$base^2 \times 1 \quad base^1 \times 1 \quad base^0 \times 1$

$1\ 9\ 8$

$1 \times base^2 \quad 9 \times base^1 \quad 8 \times base^0$

1 1 0 1 0 1

res = 0

digit × base power = Answer

$1 \times 2^0 = 1$

$0 \times 2^1 = 0$

$1 \times 2^2 = 4$

$0 \times 2^3 = 0$

$1 \times 2^4 = 16$

$1 \times 2^5 \rightarrow 32$ (Decimal) → 53

Any base :

$1 \times base^0$

$0 \times base^1$

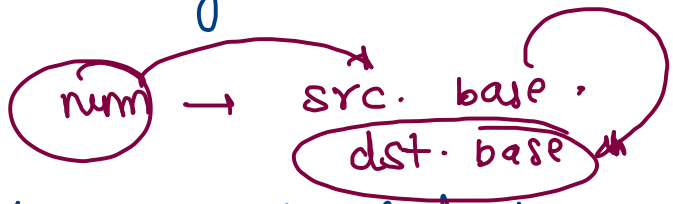
$1 \times base^2$

$0 \times base^3$

$1 \times base^4$

$1 \times base^5$

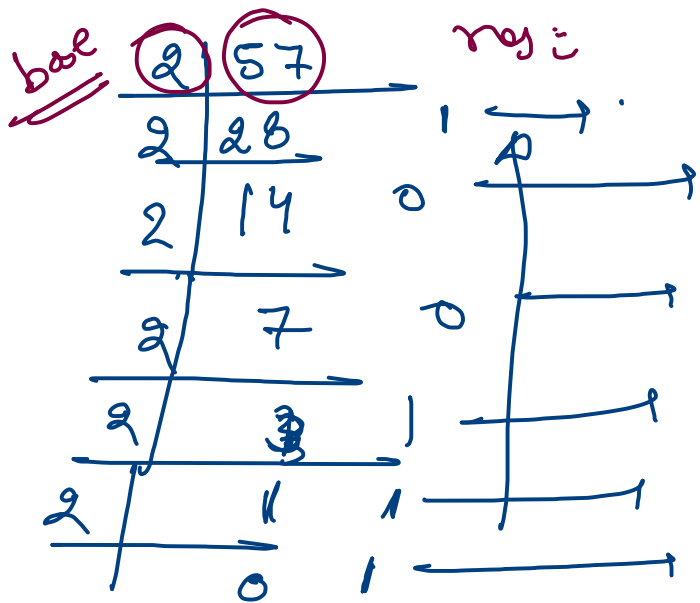
Anybase to Anybase



⇒ Anybase to decimal

⇒ Decimal to Anybase

57



⇒ num & power.

Subset / Subsequence

Subset → Array

Subsequence - String

No of arr. $= 2^n$

→ order maintain

→ but it is not necessary

that it will be continuous

String - a b c , array = {10, 20, 30}

Binary No. (a b c) → 0 → char - Reject, 1 → char - Accept → Subseq → Subsets

1. 0 0 0 → — — — → =) — → { — — — }
2. 0 0 1 → — — c → =) c → { — — 30 }
3. 0 1 0 → — b — → =) b → { — 20 — }
4. 0 1 1 → — b c → =) bc → { —, 20, 30 }
5. 1 0 0 → a — — → =) a → { 10, —, — }
6. 1 0 1 → a — c → =) ac → { 10, —, 30 }
7. 1 1 0 → a b — → =) ab → { 10, 20, — }
8. 1 1 1 → a b c → =) abc → { 10, 20, 30 }

Subarray / Substring.

Subarray - array

Substring - String.

→ order Maintain

→ Continus

String = "a b c d" : n = 4

Total count = $1+2+3+\dots+n = \frac{n(n+1)}{2}$

n+1 ... 1

n-2 ... 1

Substring:

1. a
2. a b
3. a b c
4. a b c d
5. b
6. b c
7. b c d
8. c
9. c d
10. d

Subarray →

{10, 20, 30}

{10}

{10, 20}

{10, 20, 30}

{20}

{20, 30}

{30}

Subset of an array.

{10, 20, 30}. length = 3 answer. 2^3

0 →	0	0	0
1 →	0	0	1
2 →	0	1	0
3 →	0	1	1
4 →	1	0	0
5 →	1	0	1
6 →	1	1	0
7 →	1	1	1

10, 20, 30

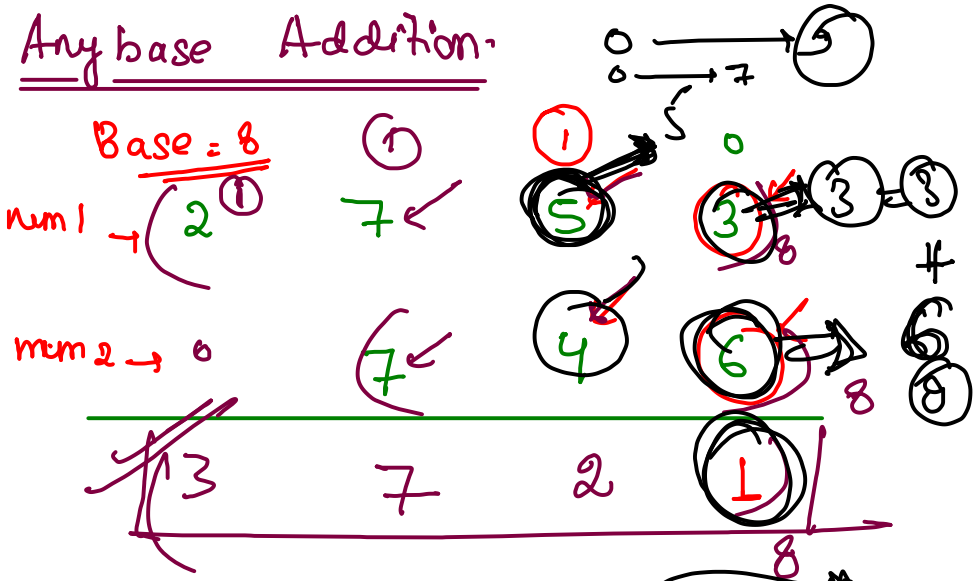
leng = 3, 2^3

-	-	-	0
-	-	30	1
-	20	-	2
-	20	30	3
10	-	-	4
10	-	30	5
10	20	-	6
10	20	30	7

binary = 10 (1)

digit = binary % 10;

Any base Addition.



carry = 0

val1 = num1 % 10 num1 /= 10;

val2 = num2 % 10; num2 /= 10;

sum = val1 + val2 + carry.

val = sum % base

carry = sum / base.

carry = ~~0~~ ✗ 0

val1 = ~~3~~ ~~8~~ ~~7~~ 2

val2 = ~~0~~ ~~4~~ ~~7~~ 0

sum = val1 + val2 + carry ~~10~~ ~~10~~ 3

val = sum % b = ~~2~~ ~~7~~ 3

carry = sum / b; ~~1~~ ~~1~~ 0

Octal Decimal

Decimal Addition

Decimal Number System.

1	1	1	0	0
	9	9	7	8 ✓
		2	5	1 ✓ carry
			12 % 10 = 2	12 / 10 = 1
1		2	2	9

```java
public static int anybaseAddition(int num1, int num2, int b) {
    int carry = 0;
    int power = 1; // 10 ^ 0;
    int res = 0;
    while(num1 != 0 || num2 != 0 || carry != 0) {
        int val1 = num1 % 10;
        num1 /= 10;
        int val2 = num2 % 10;
        num2 /= 10;
        int sum = val1 + val2 + carry;
        int val = sum % b;
        carry = sum / b;

        res += val * power;
        power *= 10;
    }
    return res;
}
```

num1 = 7 7 7

num2 = 1

1	0	0	0

(2) 2%2 = 0

Carry = 0 1 1 1 0

val1 = 7 7 7 0

val2 = 1 0 0 0

sum = 8 8 8 1

val = 0 0 0 1

res += val * power;

power = 1

val

$0 \times 10^0 \rightarrow 0 \times 10^0$ +

$0 \times 10^1 \rightarrow 0 \times 10^1$ +

$0 \times 10^2 \rightarrow 0 \times 10^2$ +

$1 \times 10^3 \rightarrow 1 \times 10^3$

res =

0
0
0
1

1000

1000

Any base Subtraction base: 8

n2: 0 2 7 5 3

n1: 7 4 6

res: 2 0 0 5

carry: ~~0~~ ~~-1~~ ~~0~~ 0 0

val2: ~~3~~ ~~5~~ ~~7~~ 2

val1: ~~6~~ ~~4~~ ~~7~~ 0

val: ~~-3~~ ~~5~~ ~~0~~ 0 2

subtraction is depend on n_2, in case of $n_2 - n_1$

carry = 0

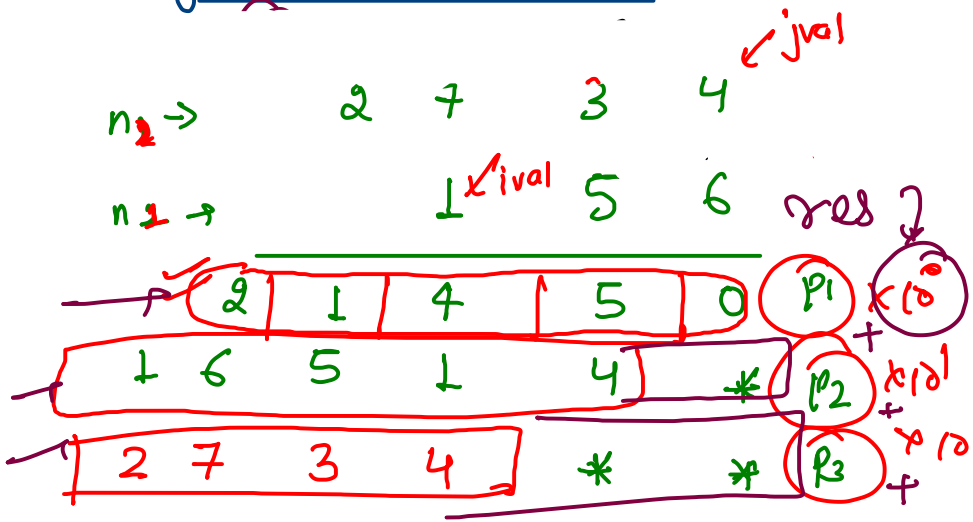
```
val2 = num2 % 10
num2 /= 10;
val1 = num1 % 10
num1 /= 10;
val  = (val2 + carry) - val1
if (subtract < 0) {
    subtract += base;
    carry = -1;
} else {
    carry = 0;
}
```

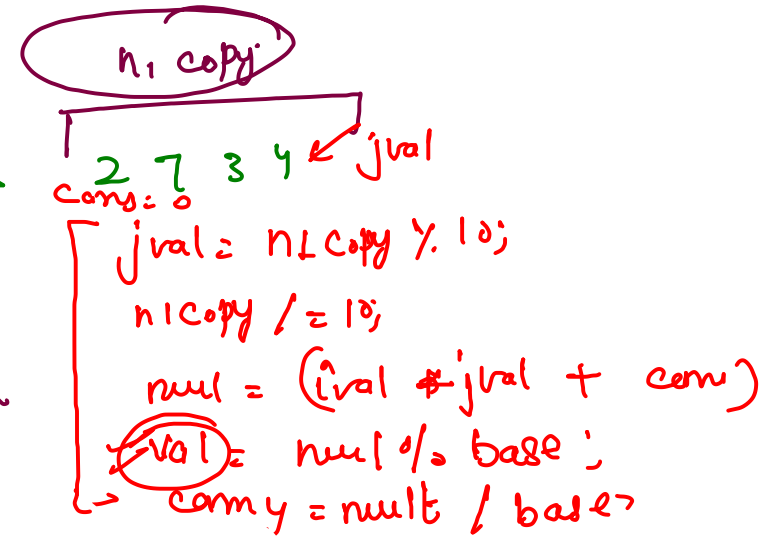
n2: 1 9 3 8

n1: 7 4 2

res: 1 1 9 6

Anybase Multiplication





carry = 0

ival = 1

jval = 4

mult = 4

val = 4

res += p1 = 2 1 4 5 0 × 10⁰ =

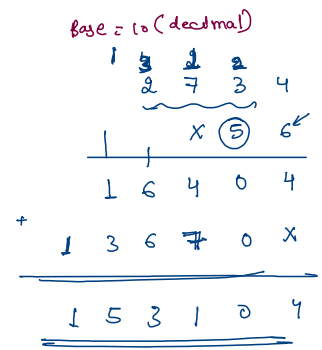
p2 = 1 6 5 1 4 0 × 10¹

p3 = 2 7 3 4 0 0 × 10²

res += p1 * 10⁰

5 0 2 2 1 0

carry base condition




ival