strings → Sequence of characters ✓
Array/List of characters ✓
Group/Set of characters ✗

{'a', 'b', 'c'}
{'b', 'a', 'c'} equal

abc ≠ bac

ASCII

[0 255]

a → 97
b → 98
⋮
z → 122

A → 65
B → 66
⋮
Z → 90

'0' → 48
'1' → 49
⋮
'9' → 57
'10' → 58 ✗
49 48

s="125"

s="-1" 2 Bytes
'-' '1'

2^8
256 unique characters
→ 8 bits → 1 Byte ✓

_ _ _ _ } → 16 numbers
3 2 1 0
0/1 0/1 0/1 0/1 → 2^4

s="abc" → 3 Bytes

s="97"
→ '7' → 55
→ '9' → 57

[0 — 255] → 256 unique values
→ 8 bits

char c='x' → 1 Bytes
int a=10 → 4 Bytes

---

char c='a'
256 different values
8 bit of memory → [0 255]
2^8 = 256

int x = 23875;
2^{32} unique values
32 bits of memory

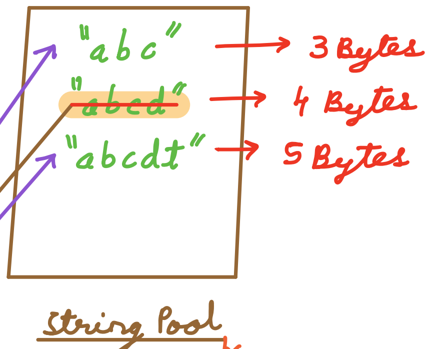
7 6 5 4 3 2 1 0
'a' → 97 → 0 1 1 0 0 0 0 1 ✓

String s="Shivam" → 6 characters → 6 Bytes

Immutable Strings (eg Java, C#, Python etc)

s = "abc"
s = s + 'd' // abcd
s = s + 't' // abcdt

s1 = "abc"
s2 = "abcdt"
s3 = "abc"

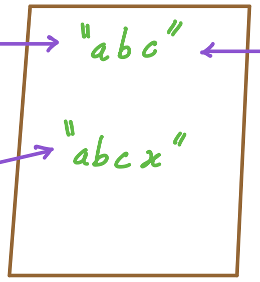


String Pool
↳ collection of all strings in memory.

Garbage Collector ✓

s = 'abc'
s1 = s

s1 += 'x' // abcx
print(s) → abc ✓

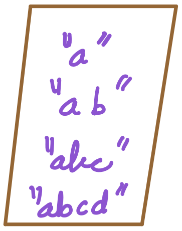


s5 = "abc"

String Pool

Disadvantage

	#iterations
String s = "a"	1
s = s + 'b'	2
s = s + 'c'	3
s = s + 'd'	4
	10



String Pool

Append N characters in a empty string

#iterations → $1 + 2 + 3 \ldots N = \frac{N*(N+1)}{2}$ N → length of string.

TC = $O(N^2)$

Generating a string char by char of length N → TC = $O(N^2)$

String Builder → dynamic array of characters.

---

s = s + 'x' → TC = O(length of s)

s = s + "abcd"

$$\left.\begin{array}{l} N+1 \\ N+2 \\ \vdots \\ N+K \end{array}\right\} \rightarrow N*K + \frac{K(K+1)}{2} = O(N*K + K^2) \rightarrow O(N*K) \quad (K<N)$$

Q → Given a string s. Toggle the case of every character.

"aBcdE" → "AbCDe"

a → 97 A → 65 97 − 65 = 32

b → 98 B → 66 98 − 66 = 32

⋮ ⋮

z → 122 Z → 90 122 − 90 = 32

```
for i → 0 to (N-1)
    if ( s[i] >= 'a' && s[i] <= 'z')
        s[i] -= 32
    else if ( s[i] >= 'A' && s[i] <= 'Z')
        s[i] += 32
              ↳ 'a' − 'A'
        abs('A' − 'a') ✓
```

if ('a' > 'A') ⇒ lower to uppercase

subtract ✓

---

a → 97
b → 98
⋮
z → 122

$64 < \underbrace{122}$ < 128

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

$97 - 64 = 33$
$98 - 64 = 34$
⋮
$122 - 64 = \underbrace{58}$

$32 < 58 < 64$

A → 65
B → 66
⋮
Z → 90

$64 < \underbrace{90}$ < 128

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

1 ___ 26

	7	6	5	4	3	2	1	0
a → 97 →	0	1	1	0	0	0	0	1
A → 65 →	0	1	0	0	0	0	0	1
z → 122 →	0	1	1	1	1	0	1	0
Z → 90 →	0	1	0	1	1	0	1	0

difference → $32 = 2^5$

toggle 5th bit → ^32   ^(1<<5)

s[i] ^= 32 ✓

TC = O(N) ✓
SC = O(1) → O(N)
to convert string to char array or StringBuilder.

Q → Given a string with lower case characters. Sort in dictionary order.

a − z
26

s = "utkarsh" → "ahkrstu"
s = "vinit" → "iintv"

Sorting Algo → TC = O(N log(N))

s = "d a b d b e d a" → aabbddde

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

a____ b____ c____ d____ . . . z____

freq(a) freq(b) freq(c)

F =

2	2	0	3	1	- - -	0	0	0
0	1	2	3	4		23	24	25
a	b	c	d	e		x	y	z

a — 0
b — 1
⋮
z — 25

'a' - 97 = 0
'b' - 97 = 1
⋮
'z' - 97 = 25

'a' - 'a' = 0
'b' - 'a' = 1
⋮
'z' - 'a' = 25

```
for i → 0 to (N-1)
    F[s[i] - 'a'] += 1
```

→ TC = O(N)

SC = O(26) = O(1)

```
for i → 0 to 25
    char c = (char)(i + 'a')
    for j → 1 to F[i]
        print(c)
```

→ TC = ~~O(N²)~~ ✗

~~26 * N~~ ✗

F[a] + F[b] + F[c] -- F[z] = N

TC = O(N)

i	j	# iterations
0	1 — F[0]	F['a']
1	1 — F[1]	F['b']
⋮		⋮
25	1 — F[25]	F['z']
		length of string = N

Total

TC = O(N)

SC = O(1)

Count Sort

10^9 unique values → ✗ → memory too high.