Hashing

$$325$$

$$3 \times 10^{2} + 2 \times 10^{1} + 5 \times 10^{0}$$

 add 7 at the front.

$$325$$

$$3 \times 10^{2} + 2 \times 10^{1} + 5 \times 10^{0}$$

append 9 at the end

$$7325 = 7 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$

- To avoid overflow use Mod

- Base and Mod should be co-prime

Use double hashing to avoid collison

base1 = 1827017
mod1 = 100888001

base2 = 5195977
mod2 = 133767331

$$\begin{aligned}\text{hash}(s) &= s[0] + s[1] \cdot p + s[2] \cdot p^2 + \ldots + s[n-1] \cdot p^{n-1} \mod m \\ &= \sum_{i=0}^{n-1} s[i] \cdot p^i \mod m,\end{aligned}$$

```cpp
long long compute_hash(string const& s) {
    const int p = 31;
    const int m = 1e9 + 9;
    long long hash_value = 0;
    long long p_pow = 1;
    for (char c : s) {
        hash_value = (hash_value + (c - 'a' + 1) * p_pow) % m;
        p_pow = (p_pow * p) % m;
    }
    return hash_value;
}
```

Find hash of a substring in O(1) time.

$$H(j, i) = (H(j, 0) - H(i-1, 0)) / base^{i}$$

- Precalculate all prefix hashes in $O(n)$ time.

Find hash of a substring in O(1) time.

$$H(j, i) = (H(j, 0) - H(i-1, 0)) / base^{i}$$

4	3	2	1	0
3	2	9	5	4

$$3 \times 10^{4} + 2 \times 10^{3} + 9 \times 10^{2} + 5 \times 10^{1} + 4 \times 10^{0}$$

Concatenate two strings

$$H = H_1 * base^{l_2} + H_2$$

$S_1 = 25$

$H_1 = 2 \times 10^1 + 5 \times 10^0$

$l_1 = 2$

$S_2 = 37$

$H_2 = 3 \times 10^1 + 7 \times 10^0$

$l_2 = 2$

$$H = H_1 * base^{l_2} + H_2$$

$S_1 = 25$

$H_1 = 2 \times 10^1 + 5 \times 10^0$

$l_1 = 2$

$S_2 = 32$

$H_2 = 3 \times 10^1 + 2 \times 10^0$

$l_2 = 2$

```cpp
struct Hash {
    ll val1 = 0, val2 = 0;
    ll base1 = 1827017, base2 = 5195977;
    ll mod1 = 100888001, mod2 = 133767331;
    void push(int num) {
        val1 *= base1; val1 += num; val1 %= mod1;
        val2 *= base2; val2 += num; val2 %= mod2;
    }
    bool operator < (const Hash& p) const {
        if(val1 == p.val1) return val2 < p.val2;
        return val1 < p.val1;
    }
    bool operator == (const Hash &p) const {
        return val1 == p.val1 && val2 == p.val2;
    }
};
Hash x; // u can push many  integers in x like x.push(tmp);
map<Hash,int>cnt; //can be used
```