字符串中的哈希

主讲人：邓哲也

字符串中的哈希

假设有 n 个长度为 L 的字符串，问其中最多有几个字符串是相等的。

直接比较两个长度为 L 的字符串是否相等时间复杂度是 $O(L)$ 的。

因此需要枚举 $O(n^2)$ 对字符串进行比较，时间复杂度 $O(n^2L)$

如果我们把每个字符串都用一个哈希函数映射成一个整数。

问题就变成了查找一个序列的众数。

时间复杂度变为了 $O(nL)$

字符串中的哈希

一个设计良好的字符串哈希函数可以让我们先用 O(L) 的时间复杂度预处理，之后每次获取这个字符串的一个子串的哈希值都只要 O(1) 的时间。

这里我们就重点介绍 BKDRHash

BKDRHash

BKDRHash 的基本思想就是把一个字符串当做一个 k 进制数来处理。

```
int k = 19, M = 1e9 + 7;
int BKDRHash(char *str){
    int ans = 0;
    for (int i = 0; str[i]; i ++)
        ans = (1LL * ans * k + str[i]) % M;
    return ans;
}
```

BKDRHash

假设字符串 s 的下标从 1 开始，长度为 n。

```
ha[0] = 0;
for (int i = 1;i <= n;i ++)
    ha[i] = (ha[i - 1] * k + str[i]) % M;
```

我们知道 ha[i] 就是 s[1..i] 的 BKDRHash

那么现在询问 s[x..y] 的 BKDRHash ，你能快速求解吗？

BKDRHash

注意到 $ha[y] = s[1]k^{y-1} + s[2]k^{y-2} + \cdots + s[x-1]k^{y-x+1} + s[x]k^{y-x} + \cdots + s[y]$

注意到 $ha[x-1] = s[1]k^{x-2} + s[2]k^{x-3} + \cdots + s[x-1]$

而我们要求的 $s[x..y]$ 的哈希值为 $s[x]k^{y-x} + \cdots + s[y]$

可以发现 $s[x..y] = ha[y] - ha[x-1]k^{y-x+1}$

因此我们预处理出 ha 数组和 k 的幂次，每次询问 $s[x..y]$ 的哈希值，只要 O(1) 的时间。

边学边练

阿轩在纸上写了两个字符串，分别记为A和B。利用在数据结构与算法课上学到的知识，他很容易地求出了“字符串A从任意位置开始的后缀子串”与“字符串B”匹配的长度。

不过阿轩是一个勤学好问的同学，他向你提出了Q个问题：在每个问题中，他给定你一个整数x，请你告诉他有多少个位置，满足“字符串A从该位置开始的后缀子串”与B匹配的长度恰好为x。

例如：A=aabcde，B=ab，则A有aabcde、abcde、bcde、cde、de、e这6个后缀子串，它们与B=ab的匹配长度分别是1、2、0、0、0、0。因此A有4个位置与B的匹配长度恰好为0，有1个位置的匹配长度恰好为1，有1个位置的匹配长度恰好为2。

$1 \leqslant N,M,Q \leqslant 200000$

边学边练

核心问题就是：给定两个字符串 A，B。

求出 A 的每个后缀子串和 B 的最长公共前缀。

标准做法是扩展 KMP，时间复杂度线性。

我们来用 Hash 试试看。

边学边练

前面已经提到，我们可以用 O(n)预处理 O(1)处理出一个子串的哈希值。

求字符串 A[i..n] 与字符串 B[1..m] 的最长公共前缀？

二分！

二分长度 mid

计算出 A[i..i+mid-1] 和 B[1..mid] 的哈希值，比较是否相等。

因此时间复杂度是 O(log n) 的！

边学边练

```
ll getha(int x, int y){
    return ha[y] - ha[x - 1] * p[y - x + 1];
}
ll gethb(int x, int y){
    return hb[y] - hb[x - 1] * p[y - x + 1];
}
```

边学边练

```
int main(){
    scanf(″%d%d%d″, &n, &m, &Q);
    scanf(″%s″, a + 1);
    scanf(″%s″, b + 1);
    p[0] = 1;
    for(int i = 1;i <= max(n, m);i ++) p[i] = p[i - 1]
    * P; for(int i = 1;i <= n;i ++) ha[i] = ha[i - 1]
    * P + a[i];  for(int i = 1;i <= m;i ++) hb[i] =
    hb[i - 1] * P + b[i];
```

边学边练

```
for(int i = 1;i <= n;i ++){
    int l = 1, r = min(m, n - i + 1), mid;
    while(l <= r){
    mid = (l + r) >> 1;
    if(getha(i,i+mid-1) == gethb(1,mid)){
          l = mid + 1;
    } else {
          r = mid - 1;
    }
}
cnt[r]++;
}
```

下节课再见