字符串哈希

Hash 的思想

Hash 的核心思想在于，将输入映射到一个值域较小、可以方便比较的范围。

> 这里的“值域较小”在不同情况下意义不同。
>
> 在哈希表中，值域需要小到能够接受线性的空间与时间复杂度。
>
> 在字符串哈希中，值域需要小到能够快速比较（$10^9、10^{18}$ 都是可以快速比较的）。
>
> 同时，为了降低哈希冲突率，值域也不能太小。

我们定义一个把字符串映射到整数的函数 f，这个 f 称为是 Hash 函数。

我们希望这个函数 f 可以方便地帮我们判断两个字符串是否相等。

具体来说，我们希望在 Hash 函数值不一样的时候，两个字符串一定不一样。

另外，反过来不需要成立。我们把这种条件称为是单侧错误。

我们需要关注的是什么？

时间复杂度和 Hash 的准确率。

通常我们采用的是多项式 Hash 的方法，即 $f(s)=\sum s[i]\times b^i\pmod{M}$。

这里面的 b 和 M 需要选取得足够合适才行，以使得 Hash 函数的值分布尽量均匀。

如果 b 和 M 互质，在输入随机的情况下，这个 Hash 函数在 $[0,M)$ 上每个值概率相等，此时单次比较的错误率为 $\frac 1 M$。所以，哈希的模数一般会选用大质数。

Hash 的实现

参考代码：（效率低下的版本，实际使用时一般不会这么写）

```
using std::string;

const int M = 1e9 + 7;
const int B = 233;

typedef long long ll;

int get_hash(const string& s) {
  int res = 0;
  for (int i = 0; i < s.size(); ++i) {
    res = (ll)(res * B + s[i]) % M;
  }
```

```
  return res;
}

bool cmp(const string& s, const string& t) {
  return get_hash(s) == get_hash(t);
}
```