

# Hash

这里我们只讲简单的hash

- by hjl

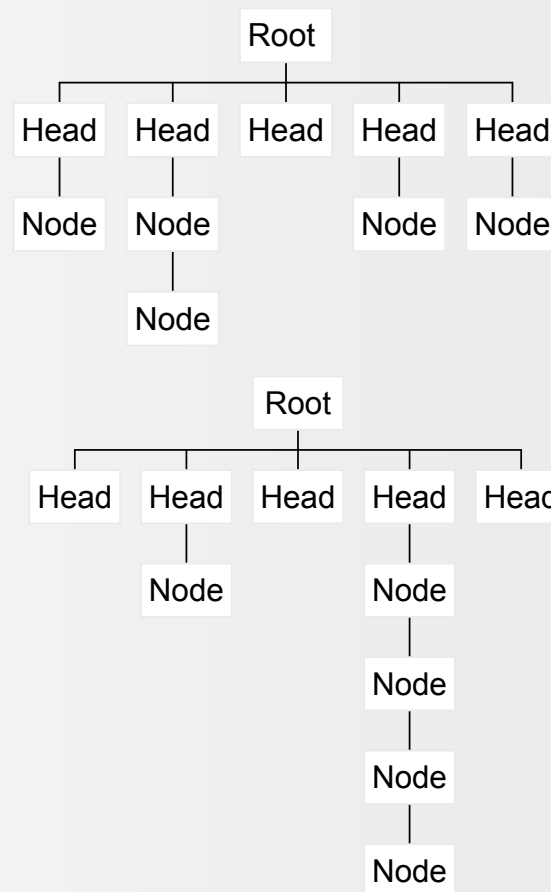
- 字典
- 编译器
- 搜索引擎
- （其实Hash就是一个映射）

## 从Hash表到Hash函数

- 既然Hash的应用如此广泛，一个好的Hash函数则显得尤为重要。
- 在Hash函数的帮助下，Hash表可以处理各种各样的数据
- 整数、实数、字符串、排列组合.....

# Hash函数优劣的评价

- 解决冲突是Hash表的关键
- 冲突越少，Hash表的效率就越高
- 数据分布越均匀，冲突越少
- Hash函数的随机性越好，数据分布越均匀



## 主要思想

- 一般是将要处理的東西的特征碼搞出來。
- 這個特征碼被稱之為Hash值。搞這個Hash值的函數被稱之為Hash函數。
- 上面都是扯淡，會運用就好。

## 常见的Hash函数？

- A.大数取模
  - 比方说我们要离散一个数字，那么用这个数字对一个大数取模，作为它的特征码。
  - 这个大数，一般是选用质数。
- B.随机数字
  - 生成一个随机数表。之后做一些奇奇怪怪的判断。比方说如果是一位数就乘上Ran[1]，如果在[33..55]之间就乘上Ran[8]什么的。注意最后还是要取模，限定范围。

## 常见的Hash函数？

- C.定址法。
- 就是 $H(x)=x$ 。没什么用的感觉。如果可以 $H(x)=x$ ，还要 $H(x)$ 干什么……
- D.平方中位数。
- 平方之后，取中间的log位作为Hash值。
- E.杂
- 几个套一起用，还可以自己加一点奇奇怪怪的东西。

- `int Hash[size];`
- `int GetHashCode(int x){`
- `int i=(x*233+x&5841)%size;//这个东西可以自己乱搞`
- `while(Hash[i]!=x)i=(i+1)%size;`
- `return i;`
- `}`
- 一般size开元素总数的5~10倍左右
- 这个GetHash是返回x在Hash数组中所在的位置



## 插入一个元素


- void insert(int x){
- int y=GetHash(x);
- Hash[y]=x;
- }

## 查找一个元素是否在 Hash 表里

- `bool search(int x){`
- `int y=GetHash(x);`
- `if(Hash[y]==0)return false;`
- `else return true;`
- `}`

## 模板题

- 题意:给出 $n$ 个正整数, 然后有 $m$ 个询问, 每个询问一个整数, 询问该整数是否在 $n$ 个正整数中出现过。
- $n, m \leq 1e5, a_i \leq 1e9$

- 
- 直接把所有元素都放到hash表里面
  - 查询就直接处理就好了