bloom_filter 实验报告

------闫世杰 2020200982

实现 bloom filter

思路:

先根据数据规模计算选择 hash_function 个数和 bloom_filter 长度,根据 hash_functions 将数据映射到 bloom_fliter 上,查询时计算检查查询数所在的位置即可

实现:

数据规模 10000000,将 bloom_fliter 长度设置为 8000000

hash_function = ln2*8 = 5

hash functions 选取(选择较为简单的 hash function):

multiplicative hashing:

1)hash1 = 101*x mod p mod m

liner hashing:

2)hash2 = 101*x+997 mod p mod m

polynomial hashing:

3)hash3 = 701+103x+101x^2 mod m

4)hash4 = 809+107x+103x^2 mod m

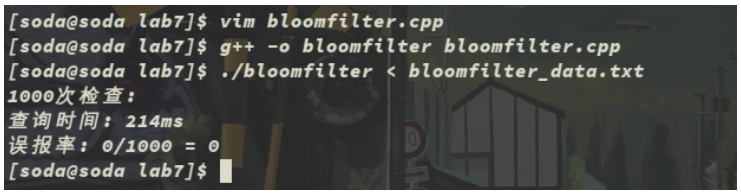
5)hash5 = 997+109x+107x^2 mod m

随机生成 check_num 个(个人设置为 1000)待查询数字,进行 bloom_filter 查询,将结果保存在 ans 数组中,再利用普通的遍历查询进行结果的核对

空间消耗: 用于 bloom_filter 空间仅有 8000w bit = 1000w byte = 250w int

时间消耗: O(1)的时间既可完成

由于读取数据以及进行了核对,线性遍历搜索进行核对时会花费较长时间,因此程序运行的总时间会比较长(1000 次查询大概需要 10s 左右),个人仅统计了 bloom_filter 查询的时间



对比 hash:

采用 open_address-liner probing 实现 hash

空间比较:将 load factor 设置成 2/3,因此需要储存 1500w int,比 bloom_filter(250w int)大了 1 个数量级

时间比较:由于可能会进行多次寻址,且如果数据不存在最差情况下可能会遍历 1000w 个数据,因此时间消耗会高于 bloom_filter,测试发现时间也高 1 个数量级