**电梯早高峰问题，假设只有一部电梯，怎么样降低早高峰电梯门口等待的人多的问题？**

1. 硬件上：考虑能否增加电梯；
2. 人员安排：协调各公司使用弹性工作制；提倡低楼层人员爬楼梯分流；改善等待方式，扇形—>排队；
3. 电梯的功能：在早高峰时，电梯几乎在每一层都停，使得电梯一致处于低速状态，效率很低，可以设定一个时段，在该时段内，电梯只在固定的几个楼层停。

**五大缓存问题**

**gc引用计数法的缺点，除了循环引用，说一到两个**

考虑stw、访问局部性、堆大小（这个没听清楚），然后问我局部访问性知道吗，我以为是jvm里的新知识就说了不知道，然后他解释了一下，我才知道噢原来就是os的局部性原理。。。

然后他问你觉得引用计数\*\*\*影响到哪几点，面试官看我没反应就说ok，然后跟我解释了，主要是一个链式更新、堆的使用效率的问题，引用计数法优点在于他是实时删除的。

**子查询与关联查询区别**

**表关联的效率要高于子查询，子查询更加灵活，容易控制。**

on条件是在生成临时表时使用的条件，它不管on中的条件是否为真，都会返回左边表中的记录。

where条件是在临时表生成好后，再对临时表进行过滤的条件（这时已经和left join没关系了），条件不为真的就全部过滤掉。

**Bit-map**

32位机器上，对于一个整型数，比如int a=1 在内存中占32bit位，这是为了方便计算机的运算。但是对于某些应用场景而言，这属于一种巨大的浪费，因为我们可以用对应的32bit位对应存储十进制的0-31个数，而这就是Bit-map的基本思想。

适合处理int十倍以内的数据量。

Bit-map算法利用这种思想处理大量数据的排序、查询以及去重。

Bitmap在用户群做交集和并集运算的时候也有极大的便利。

**已知某个文件内包含一些电话号码，每个号码为8位数字，统计不同号码的个数。**

8位最多99 999 999，大概需要99m个bit，大概10几M字节的内存即可。

排序也是一样的做法，不过要求所有数据不能重复。

**在2.5亿个整数中找出不重复的整数，内存不足以容纳这2.5亿个整数。**

采用2-Bitmap（每个数分配2bit，00表示不存在，01表示出现一次，10表示多次，11无意义）进行，共需内存232\*2bit=1GB内存，还可以接受。然后扫描这2.5亿个整数，查看Bitmap中相对应位，如果是00变01，01变10，10保持不变。所描完事后，查看bitmap，把对应位是01的整数输出即可。

**Bit-map扩展——Bloom Filter(布隆过滤器)**

Bloom Filter使用k个相互独立的哈希函数（Hash Function），它们分别将集合中的每个元素映射到{1,…,m}的范围中。对任意一个元素x，第i个哈希函数映射的位置hi(x)就会被置为1（1≤i≤k）。p.s.如果一个位置多次被置为1，那么只有第一次会起作用，后面几次将没有任何效果。

Bloom Filter 在判断y是否属于这个集合时，对y应用k次哈希函数，若所有hi(y)的位置都是1（1≤i≤k），就认为y是集合中的元素，否则就认为y不是集合中的元素。

有一定的误判率--在判断一个元素是否属于某个集合时,有可能会把不属于这个集合的元素误判为属于这个集合。因此,它不适合那些"零误判"的应用场合。在能容忍低误判的应用场景下,布隆过滤器通过极少的误判换区了存储空间的极大节省。