2357 数

明显2257数就是质因数分解后，质因数只有2、3、5、7的数，因此直接用这些素数相乘生成符合条件的数，排序，二分找即可。

另外可以使用stl中的set，可降低编程复杂度。

监狱

30%的分数直接枚举顺序即可；

70%的分数：我们发现，这是一个和区间有关的问题，因此，我们可以想到区间型DP

f[i,j]表示在[i,j]区间内所有该被释放的人都被释放的最小代价

则f[i,j]=max{f[i][k-1]+f[k+1][j]+(j-i)}

复杂度：O(N^2\*M)

100%的分数：

我们发现，动态规划所涉及到的区间并不是所有长度所有组合的都需要，只有被释放的人的的位置才能构成区间端点，因此，动归就只需要考虑M个状态就行了。

复杂度：O(M^3)

服务器信息储存

本题的难点在于问题的规模。如果问题的规模在100左右，那么这将是一道非常容易的题目。因为O(n3)的算法是很容易想到的：

（1）求出任意两点间的最短路径，时间复杂度为O(n3)；

（2）枚举任意两点，根据定义判断一个节点是否对另一个节点感兴趣，时间复杂度为O(n3)。

当然，对于30000规模的本题来说，O(n3)的算法是绝对不可行的，即便降到O(n2)也不行，只有O(nlog2n)或O(n)是可以接受的。

既然现在可以得到的算法与要求相去甚远，要想一鼓作气得到一个可行的算法似乎就不是那么容易了。我们不妨先来看看我们可以做些什么。

判断一个节点V是否对节点W感兴趣，就是要判断是否存在一个rank大于r(W)的节点U，δ(V, U)<δ(V, W)。所以，节点V到某个特定的rank的节点（集合）的最短距离是一个非常重要的值。如果我们可以在O(nlog2n)时间内求出所有节点到特定rank的节点（集合）的最短距离，我们就成功地完成了算法的一个重要环节。

用Dijkstva算法反过来求特定rank的节点（集合）到所有点的最短距离——以所有特定rank的节点为起点（rank=1, 2, 3, …或10)，求这一点集到所有点的最短距离。由于图中与每个点关联的边最多只有10条，所以图中的边数最多为5n。用Priority Queue（Heap, Winner Tree或Fibonacci Heap等）来实现Dijkstra算法，时间复杂度为O((n+e)log2n)（用Fibonacci Heap实现，更确切的时间复杂度是O(nlog2n+e))。这里，e＝5n，因而求一遍最短路径的时间复杂度为O(nlog2n)。由于1≤rank≤10，每个rank都要求一遍最短路径，所以求出每个节点到所有rank的最短路径长度的时间复杂度为O(10\*(5+1)nlog2n)，即O(nlog2n)。

求出所有点到特定rank的节点(集合)的最短距离，就完成了判断任意节点V对W是否感兴趣的一半工作。另一半是求任意节点V到W的最短距离。前面求节点到rank的最短距离时，利用的是rank范围之小——只有10种，以10个rank集合作起点，用Dijkstra算法求10次最短路径。但是，如果是求任意两点的最短路径，就不可能只求很少次数的最短路径了。一般来说，求任意两点最短路径是Ω(n2)的（这只是一个很松的下界），这样的规模已经远远超出了可承受的范围。但是，要判断V对W是否感兴趣，δ(V, W)又是必须求的，所以n次Dijkstra算法求最短路径肯定是逃不掉的（或者也可以用一次Floyd算法代替，但是时间复杂度一样，可认为等价)。那么，我们又能从哪里来降这个时间复杂度呢？

题目中提到：所有服务器储存的数据量（|B(V)|之和）不会超过30n。这就是说，最多只存在30n对(V, W)满足V对W感兴趣。所以，就本题来说，我们需要处理的点对最少可能只有30n个，求最短距离的下界也就变成Ω(30n)=Ω(n)了（当然，这也只是很松的下界）。虽说下界是Ω(n)，其实我们只需要有O(nlog2n)的算法就可以满足要求了。

从前面估算下界的过程中我们也看到，计算在下界中的代价都是感兴趣的点对（一个节点对另一个节点感兴趣），其余部分为不感兴趣的点对。我们如果想降低时间复杂度，就要避免不必要的计算，即避免计算不感兴趣的点对的最短路径。

我们来看当V对W不感兴趣时的情况。根据定义，δ(V, W)>δ(V, r(W)+1)。如果是以W为起点，用Dijkstra算法求最短路径的话。当扩展到V时，发现V对W不感兴趣，即δ(V, W)>δ(V, r(W)+1)。那么，如果再由V扩展求得到U的最短路径，则：

δ(U, W)=δ(V, W)+δ(U, V)，

δ(U, r(W)+1)=δ(V, r(W)+1)+δ(U, V)，

由于δ(V, W)>δ(V, r(W)+1)，

所以δ(V, W)+δ(U, V)>δ(V, r(W)+1)+δ(U, V)，即δ(U, W)>δ(U, r(W)+1)

所以，U对W也不感兴趣。因此，如果以W为起点，求其他点到W的最短路径，以判断其他点是否对W感兴趣，当扩展到对W不感兴趣的节点时，就可以不继续扩展下去了（只扩展对W感兴趣的节点）。

我们知道，所有感兴趣的点对不超过30n。因此，以所有点作起点，用Dijkstra算法求最短路径的时间复杂度可由平摊分析得为O(30(n+e)log2n)=O(30(n+5n)log2n)=O(nlog2n)。

由此，我们看到判断一节点是否对另一节点感兴趣，两个关键的步骤都可以在O(nlog2n)时间内完成。当然算法的系数是很大的，不过由于n不大，这个时间复杂度还是完全可以承受的。下面就总结一下前面得到的算法：

（1）分别以rank=1, 2, …, 10的节点（集合）作为起点，求该节点（集合）到所有点的最短距离（其实也就是所有点到该节点（集合）的最短距离）；

（2）以每个点作为起点，求该点到所有点的最短距离。当求得某节点的最短距离的同时根据求得的最短距离和该节点到rank大于起点的节点（集合）的最短距离，判断该节点是否对起点感兴趣。如果感兴趣，则找到一对感兴趣的点对，否则，停止扩展该节点，因为该节点不可能扩展出对起点感兴趣的节点。

总结解题的过程，可以发现解决本题的关键有三点：一是稀疏图，正因为图中边比较稀疏所以我们可以用Dijkstra+Priority Queue的方法将求最短路径的时间复杂度降为O(nlog2n)；二是rank的范围很小，rank的范围只有10，所以我们只用了10次Dijkstra算法就求得了所有点到特定rank的最短距离；三是感兴趣的点对只有很少，由于感兴趣的点对只有30n，我们通过只计算感兴趣点对的最短路径，将求点与点间最短路径的时间复杂度降到了O(nlog2n)。这三点，只要有一点没有抓住。本题就不可能得到解决。