佳佳的魔法照片(Magic Photo)

这道题我认为是不用详细说的，主要的考点应该是排序。比较重要的一点是必须要保证稳定排序。而最多人喜欢用的快排却是不稳定的。标程解决这一问题的方法是：首先记录下每个人的初始编号；在快排中，比较两个人的大小时，如果两个人的权值相同，再比较其初始编号。至于第二遍排序（事实上只需要找到权值最大的 k 个人即可）的最好实现方法显然是堆排序，因为它在选出最大的 k 个人以后就可以立即结束，而不用将全部的人都排序。标程采用了两次快排，这是标程不如某些提交上来的程序快的主要原因。

佳佳的魔法药水(Magic Syrup)

首先，这道题就是赛前的投票中得票率第四的“图论”。当我把这题给 ayliyh 大牛看的时候，他的第一反应似乎也是构造图论模型，但是却一直没构造出来……这道题是图论题吗？我认为是的，因为这道题的算法至少有两个：一个是 Dijkstra，一个是 Bellman-Ford。没错，解决这道题只需要一个最短路的算法。只不过这里要应用的是最短路算法的思想，而不是照抄。下面以利用 Dijkstra 算法的思想解此题为例来说明。

每次，我们可以从所有未确定最终价格的药水中选一个价格最低的，它的当前价格肯定就是最优价格（即它的最优价格已确定）。然后我们找所有的用这个药水作为原料之一可以配制成的药水，更新它们的当前价格。事实上，只有在两个原料的价格都标记为最终确定后才更新药水的当前价格。至于最优方案的种数也很好确定，只需在每次更新价格时区分小于和等于两种情况即可。

这道题是我原创的所有题目中最满意的。它提供了一种考察某种算法的新思路。也就是说：这道题考察了 Dijkstra 算法；但是要想做出这道题，仅仅会背出 Dijkstra 算法是不行的，你必须深刻理解 Dijkstra 算法的思想精髓（也就是每次贪心地取最小值然后将其值固定）。

佳佳的魔杖(Magic Wand)

题目意思十分明确：给出一个数列和每个数的权值，选择若干个互相之间不包含的子序列使得这些子序列的和在某个范围内且所选择的权值之和（被重复选择则重复计算多次）最大。

首先需要说明的是，尽管很多有关区间选择的问题都有贪心的解决方法，但这个题目并没有什么显然的贪心结论。或许大家能想到各种可能的贪心方案，但下面的例子可以推翻几乎所有的贪心方案。

试想这样一个数据::

7 1 7

1 1 1 1 1 1 1

1 1 1 1 1 1 1

在长度上，任意一段或几段都满足要求。这个数据的答案应该是 16，它是由 4 个连续

的[1 1 1 1]得到的。需要注意的是，这种分割方案并没有保证“总是选择最短的”、“总是选择最长的”、“总是选择最先满足要求的”等条件。取的区间长度只是总长度的一半，原因仅仅是因为这样可以让每一个权值尽可能被计算多次。

这个题目可以用动态规划来解决。

设 f[i,j]表示在最后一个区间的起始点不超过 i，结束点不超过 j 的情况下可以获得的最大值。这个状态可能是由 f[i-1,j]转移过来的，也可能是由 f[i,j-1]转移过来的，还有可能是将 i..j 选为一个新的区间得到的。在最后一种情况中，为了避免包含关系，前一个区间的起始点不能超过 i-1，结束点不超过 j-1。于是，我们得到了类似于最长公共子序列(LCS)问题的转移方程，对所有的 i<=j:

f[i,j]= Max{ f[i-1,j] ,

f[i,j-1]　当 i<=j-1 时 ,

f[i-1,j-1] + M[j]+M[j+1]+...+M[i]　当 lo<=L[j]+L[j+1]+...L[i]<=hi 时 }

为了快速地得到 M[j]+M[j+1]+...+M[i]与 L[j]+L[j+1]+...L[i]，我们可以初始化两个数组 Sm 和 Sl。以处理 M 数组为例，Sm[i]表示 M[1]+M[2]+...+M[i]，需要计算 M[j]+M[j+1]+...+M[i]的值时，我们只需要计算 Sm[i]-Sm[j-1]就行了。处理 L 数组也是同样的方法。

虽然所有的 M[i]的总和最多也只有 10^9，但由于每个 M[i]都可能被计算多次，因此最后的结果有可能超过 longint。

佳佳的魔法阵(Magic Matrix)

首先，这道题应该是比较典型的深搜，需要搜索各种可能的 n*m 矩阵中的路径。

但是，单纯的、不加任何剪枝的搜索大约只能过 3 组。我们在这道题中必须用到搜索中最典型的两种剪枝：最优性剪枝和可行性剪枝。

最优性剪枝的思路相对简单：当我们搜索到某一条路径，发现它的某个 k1*|a-c|+k2*|b-d|已经超过了当前已经求出的最优解，那么肯定是不用沿这条路径继续搜索下去了，直接回溯即可。

可行性剪枝的基本思路就是：在搜索过程中判断当前是否还能够走成一条完整的路径。简单来说，当未走到过的方格是不连通的两个部分时，肯定就无法走完剩下的路径，必须剪枝。具体的判断方法是：如果 (当前方格的上下两个方格都被访问过 且 左右两个方格都没有被访问过) 或者 (当前方格的左右两个方格都被访问过 且 上下两个方格都没有被访问过)，则可以剪枝。这样可以在 O(1)的时间内做到可行性判断。

这道题的最高得分是 80 分，这也是本次比赛中唯一没有得满分的一题。希望大家能从本题中的到搜索剪枝方面的启迪。