**地大校赛解题报告**

1. **Degree of Dependency**

题意：在给定随机变量X和Y对应的n个取样值{(x1,y1),(x2,y2),...,{xn,yn}}时，根据公式（2）计算X相对于Y的依赖度D（X,Y）。

题解：本题的关键是了解D(X,Y)的计算过程，考察的是快速学习和理解能力。

具体地，D（X，Y）的计算过程是由一串双样本KS检测的乘积组成，对于每一个KS检测，其输入为X的全局取样值和X的局部取样值X|Y<y组成。

一个直观的算法是不经过任何预处理，按照该计算过程进行计算，这样时间复杂度为:O（m\*n^2）=一个ks检测的时间O(n^2)\*ks检测的次数m。

由于n可达10^5，因此该算法会TLE。

一个优化是观察到ks检测的两个取样样本X和X|Y<y中X是不变的，而X|Y<y则始终是X的一个真子集。利用这一个观察可以在D(X,Y)计算之前进行预处理。

具体如下：

1.利用“桶”的想法可预处理出样本X的累积概率分布CDF（需要关注的）。

2.同时可构建X的取样值与其对应桶编号值之间的映射关系。

通过这两点预处理，整体复杂度将为 O（m\*n）,可顺利AC通过。

1. **The Least Subsequence**

题意：求一个段落中包含了所有关键字的最短子段落的长度。

题解：一个暴力的解法是枚举每一个可能的子段落，检验其是否包含所有关键字。

但是该暴力解法复杂度为O(n^3),显然是会TLE的。

这道题的标程采用DP（动态规划）来解决。为了方便处理，对输入单词进行预处理，将关键字集合keywords和段落均映射成一个整数数组。

之后再进行DP，具体分为两步：

1.先计算出next数组，其中next[i]:min{ k| k>i &&ai=ak};即与ai值相同的右边最邻近的位置；由于keyword的最多有100种，因此可以在O(n)内求得next数组。

2.定义状态dp[i]:max{k |k<i&&子段落[k,i]满足条件}，即从ai往左数，能全部包含关键字的最段子段落。

然后可在O(n)时间内求解dp数组，具体方法如下：

由于dp[j]对应的最小子序列为[k,j],则对于a[k]而言会满足如下两个条件：

a)、a[k]!=0;

b)、不存在k<t<=j，有a[k]=a[t],因为如果有的话，[k+1,j]长度会更小；

因此，在已知dp[j]=k的前提下，可按如下递推方程求得dp[j+1]的值。

a)、如果a[j+1]=0,dp[j+1]=dp[j]；

b)、如果a[j+1]!=0 && a[j+1]!=a[k], dp[j+1]=dp[j];

c)、如果a[j+1]==a[k],可令k值不断加一，直到遇到next[k]>j+1时,停止，因为此时意味着该处数值ak在右边没有重复的了，如果再继续前进，就不会包含关键字a[k]了。

此时该位置记为kk，有dp[j+1]=j+1-kk+1；

最终答案为：max{ i-dp[i]+1};

综合上述两步，整个DP算法的复杂度为O(n),可顺利AC通过。

1. **Language whith Minimum Cost**

题意：求构造一种由n个单词所组成语言的最小代价，其中每个单词是基于d个字符构造而成。

题解：首先并未想到什么直观、暴力的算法，由于可构造的语言数目太多，根本无法进行枚举。

这道题的标程是一种树形DP算法，时间复杂度为O(n^2\*d)。具体描述如下：

由题意可知，该语言的特点类似于哈夫曼码，假设已得到了包含n个单词的一种语言，则这n个单词对应的字典树满足：叶子节点代表一个单词，非叶子节点不表示单词，共有n个叶子节点；反之，每一颗有n个叶子节点的字典树对应一种该语言，并且该语言的代价可以由该字典树的边权值得到；因此，一种语言与一颗字典树是一一对应的。

基于此，可设计一个树形DP算法如下：

状态DP[i][j]:构建包含i个单词、并且这i个单词的第一个字符只涉及到j个字符值的语言的最小代价值。

对应到字典树则是，有i个叶子节点，并且从根节点出发有j个子树，这样的字典树中，所获得的最小代价；

状态转移方程如下：

dp[i][j] =min ( dp[i-k][j-1] +k\*w[j]+ans[k] ) // ( 1=<k<=i-j+1)

其中dp[i-k][j-1]//有j-1棵子树，并且叶子节点有i-k个时字典树的最小代价 +( k\*w[j]+ans[k])// 第一颗子树有k个叶子节点时的最小代价；

ans[k]:包含k个叶子节点的字典树的最小代价；

ans[k] =min { dp[k][ j]} // (1<j<=min( k,d)};

最终答案为：ans [n].

注意进行DP时需要对d个字符按其代价w进行从小到大的排序。

1. **The Circumcircle of a Triangle**

题意：坐标系中有n条直线，由ax+by=c的形式表示。保证任意的两条不同直线都不平行，任意的三条不同直线都不交于同一点。这些直线中任意的三条都可以构成一个三角形。求这些三角形中有多少三角形的外接圆经过原点。

题解：将外接圆经过原点这一判定条件转化为：原点向直线做垂线，三个垂足在同一条直线上。那么最直接的就是找出来垂足，然后数三个点在一条直线的三元组个数。再转化一下，就是两点确定一条直线，然后再看每一条直线上有多少个点，用组合数算。但是这样不好写，最后可以转换成枚举i,要求j和k相对于i的斜率必须一样。如果有重合的点，那么任意一个第三个点都可以。

1. **Interesting Numbers**

水题，直接暴力打表能过。

1. **Prefix and Suffix**

题意：给你两个字符串str1,str2,让你求一个最大长度的子串t，t是str1的前缀，并且是str2的后缀。

题解：水题，将str1作为模式串，将str2作为主串，直接kmp

1. **Hat’s Words**

题意：给你一些单词，让你对每一个单词进行判断，看它是不是能由两个单词拼接而成。

题解：首先将所有的单词存入字典树中，注意在存的时候，初始falg都为false,说明这个字母不是一个单词的结尾，如果某一个字母是一个单词的结尾，则flag=ture（查找判断时用到），然后对每一个单词进行判断，判断时将单词拆分，如果拆分的两个单词都在字典树中找的到，那么就说明这个单词可由这两个拆分的单词拼接而成，从而输出这个单词，并跳出来，进行下一个单词的判断。

1. **Travel around the World**

题意：需要确定两条路径使得它们的乘积最大。

题解：由于N比较小，可以枚举一下，遍历所有边，将边切断之后可以保证分别存在于两个图中的路径不会相交，这样问题就简化为求每个图（树）的最长路径，搜索一下即可。

1. **Travel around the World-II**

题意：要求在路线最短的情况下，遍历城市中的每一个景点（每个景点至少走一次）。

题解：可以反过来想，在路线最短的情况下，最终要走完所有的景点，势必有些景点需要走两次（如果路分支的话，去的时候走一次，回的时候需要再走一次），那么问题就归结为如何寻找一条最长的路径，使得这些路径上的所有景点只需走一次，而除了这路径上的景点之外，别的都需要走两次，所以最后可以得出公式： result = 所有路径长度之和 \* 2 - 最长路径。

1. **Bob’s Program**

数据比较水，随便水水就能过。

dp[value\_of\_x][2]用于保存当前x为该值时，处于步骤0或者1时（步骤0代表step2，步骤1代表step3）的状态，然后搜索即可，遇到之前访问过的状态或者满足终止条件时退出。