NOIP2016模拟题 Day2
解题报告

ZYF & WMJ

October 3, 2016

1 最长不下降子序列

1.1 算法一

直接生成整个序列，然后求出其最长不下降子序列。

时间复杂度：$O(n^2)$ / $O(n \log n)$

期望得分：10分/ 30分

1.2 算法二

1.2.1 一阶递推取模数列的周期性

若数列$\{a_n\}$满足$a_{n+1} = f(a_n) \bmod P, n \in \mathbf{N}$，则称其为一阶递推取模数列。可以发现，此类型数列必然存在周期不超过P的循环节，具体证明如下：

显然$0 \leqslant a_n < P$，故a_n最多只有P个不同的取值。根据抽屉原理，在数列前$P+1$项中必然存在两项$u, v(u < v)$满足$a_u = a_v$，那么有$f(a_u) = f(a_v) \Rightarrow a_{u+1} = a_{v+1}$。以此类推，$\forall n \geqslant v, a_n = a_{n-T}$均成立，其中$T = v - u + 1 \leqslant P$。

1.2.2 周期数列的LIS

先考虑一个简化后的问题：给定一个周期为T的纯循环数列$a_1, a_2, \cdots, a_T, \cdots, a_{nT}$，求该数列的最长不下降子序列。

设$F[n][i][j]$表示以a_i开头且以a_{nT+j}结尾的LIS长度（不包括a_i本身），那么我们有：

$$F[n][i][j]=\max_{1\leqslant k\leqslant T,a[i]\leqslant a[k]\leqslant a[j]}(F[n-1][i][k]+F[1][k][j]),\quad n\in\mathbf{N}^*$$

仔细观察，我们发现本题$C[i][j]=\max\limits_{k=1}^{n}(A[i][k]+B[k][j])$的状态转移形式与矩阵乘法$C[i][j]=\sum\limits_{k=1}^{n}A[i][k]\cdot B[k][j]$非常相似，那么我们能不能用类似矩阵乘法快速幂的算法来优化这个转移呢？

答案当然是肯定的，首先我们来证明本题中Max-Add广义矩阵乘法的结合律。

设A,B,C为三个$n\times n$的矩阵，$G_1=(A\times B)\times C$，$G_2=A\times(B\times C)$，则：

$$\begin{aligned}G_1[i][j]=&\max_{k_2=1}^{n}\left(\max_{k_1=1}^{n}A[i][k_1]+B[k_1][k_2]\right)+C[k_2][j]\\=&\max_{1\leqslant k_1,k_2\leqslant n}(A[i][k_1]+B[k_1][k_2]+C[k_2][j])\\=&\max_{k_1=1}^{n}A[i][k_1]+\left(\max_{k_2=1}^{n}B[k_1][k_2]+C[k_2][j]\right)\\=&G_2[i][j]\end{aligned}$$

由于本题中要求$a[i]\leqslant a[k]\leqslant a[j]$，会导致有些转移不合法，此时我们令所有不合法的转移$G[u][v]=-\infty$即可，单位矩阵也不难找到：

$$I=\begin{bmatrix}0&-\infty&\cdots&-\infty&-\infty\\-\infty&0&\cdots&-\infty&-\infty\\\cdots&\cdots&\cdots&\cdots&\cdots\\-\infty&-\infty&\cdots&0&-\infty\\-\infty&-\infty&\cdots&-\infty&0\end{bmatrix}$$

于是，我们直接套用矩阵乘法快速幂优化这个转移方程就好了。

1.2.3 完整算法流程

记$pos[v]$表示数字v在数列中第一次出现的位置，用$O(D)$的时间找出该数列的循环周期T。接下来我们显然可以把$a_1\sim a_n$分成三段$L+S+R$，其中S是一个周期为T的纯循环数列，数列L,R的长度均不超过D。

预处理$low[v]$表示在数列L中最后一个数不大于v的LIS，$up[v]$表示数列R中第一个数不小于v的LIS。我们用上文所述方法处理完纯循环数列S后，暴力枚举整个数列的LIS在S内那一部分的开头和结尾，再利用预处理出的数组更新答案即可。

时间复杂度：$O(D^3 \log n)$

期望得分：100分

1.3 算法三

本题存在复杂度更优的算法，但是由于可拓展性不强，故在此只是简单描述一下。

考虑如何处理纯循环数列$a_1, a_2, \cdots, a_T, \cdots, a_{nT}$的LIS，剩下的步骤与算法二相同。

显然，$a_1, a_2, \cdots, a_{nT}$只含T个不相同的数字，故任何一个不下降子序列中，最多只会有$T-1$对相邻元素其值不相等。而根据找循环节的过程我们可以得出，一个周期中的每个数字都只出现一次，所以如果最终的LIS中有两段相等的元素A和B，我们可以把其中一段用另一段代替。例如$u*A+S+v*B$，我们可以将其用$A+S+(u+v-1)*B$代替且保持序列的长度不变。

由上述分析我们可以得出，必定存在一个最优解满足$L+k*V+R$的形式。而又因为每个周期中至少会选择一个数，所以子序列L和R最多包含T个周期，其长度不超过T^2。因此当$n>2T$时，我们对前T个周期和后T个周期做类似算法二最后一步的预处理，然后枚举中间部分的元素V，用$low[V]+up[V]+(n-2T)$的值更新答案即可。

时间复杂度：$O(D^2 \log D)$

期望得分：100分

2 完全背包问题

2.1 算法一

暴力DFS，加上一些搜索顺序上的优化即可。

时间复杂度：$O(???)$

期望得分：10分

2.2 算法二

由于所有询问背包的容量都比较小，我们考虑直接动态规划。设$f[k][i][j]$表示前k个物品中，大体积物品数量不超过i件，是否存在总体积为j的方案，则：

$$f[k][i][j] = \begin{cases} f[k-1][i][j] \text{ or } f[k][i-1][j-V[k]], & V[k] \geqslant L \\ f[k-1][i][j] \text{ or } f[k][i][j-V[k]], & V[k] < L \end{cases}$$

时间复杂度：$O(nC \cdot W + m)$

期望得分：30分

2.3 算法三

对于$n = 2$的数据，问题转化为求二元一次不定方程$ax + by = c$的非负整数解。首先令$g = \gcd(a, b)$，那么当$g \nmid c$时方程显然无解。否则，我们先使用拓展欧几里得算法计算出$ax + by = g$的一组特解(x_0, y_0)，根据数论相关知识，原不定方程的所有解必定可以表示成如下形式：

$$\begin{cases} x = \frac{c}{g}x_0 + \frac{b}{g}t \\ y = \frac{c}{g}y_0 - \frac{a}{g}t \end{cases} \quad (t \in \mathbf{Z})$$

于是我们令$x \geqslant 0, y \geqslant 0$，解出这两个关于参数$t$的不等式后，判断是否存在交集即可。至于数量限制，我们只需要再添加一个方程$x \leqslant C, y \leqslant C$或$x + y \leqslant C$就行了。

时间复杂度：$O(m \log V)$

期望得分：10分

2.4 算法四

记v_0为所有物品中的最小体积，那么当$v_0 \geqslant L$时，由于对大体积物品数的限制，方案的总体积必然不会超过$n \cdot V$，所以我们类似算法二直接动态规划，则有：

$$f[k][i][j] = f[k-1][i][j] \text{ or } f[k][i-1][j-V[k]]$$

显然，这个状态转移过程我们是可以用bitset优化的。

否则当$v_0 < L$时，如果存在总体积为D的方案，那么也必然存在总体积为$D + v_0$的方案，也就是说答案存在一定的单调性。于是，我们设$f[k][i][j]$表示前k个物品中，大体积物品数量不超过i件，所有方案中总体积的最小值S，并且要求$S \bmod v_0 = j$。于是，对于背包容量为W的询问，我们只需判断一下W与$f[n][C][W \bmod v_0]$的大小关系即可。

但当我们写出如下的状态转移方程时，我们会发现该状态转移图是会存在环的。

$$f[k][i][j]=\begin{cases}\min(f[k-1][i][j],f[k][i-1][(j-V[k]) \bmod v_0])+V[k], & V[k]\geqslant L\\ \min(f[k-1][i][j],f[k][i][(j-V[k]) \bmod v_0])+V[k], & V[k]<L\end{cases}$$

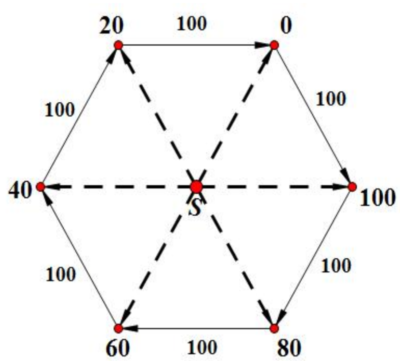
于是，我们对第二种转移，建出一个点数为v_0+1的有向图，源点S向每个节点j连一条边权为$f[k-1][i][j]$的边，每个节点j向节点$(j+V[k]) \bmod v_0$连一条边权为$V[k]$的边，使用SPFA或其他单源最短路算法实现即可。

时间复杂度：$O(nm\cdot SSSP(V+1,2V))$

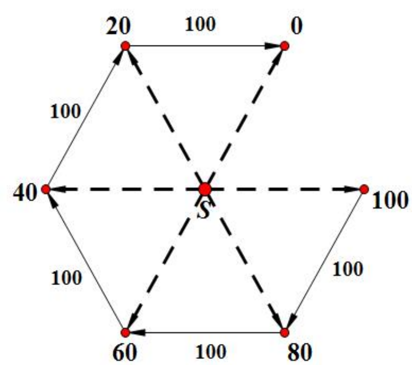
期望得分：100分

2.5 算法五

其实，我们并不需要使用最短路算法。经过分析可以发现，不考虑源点S的情况时，其余所有节点的入边和出边数都是1，那么这v_0个节点必定可以划分成若干个互不相交的简单环。而对于每个环，其转移图的形式类似下左图所示：



(a)



(b)

我们找到所有与S相连的顶点中，对应边权最小的顶点P，那么S到P的最短路必然是直接走虚线边$S\rightarrow P$。所以，直接删去连向P点的实边，之后该图变成了一个拓扑图，于是我们递推即可。

时间复杂度：$O(mnV)$

期望得分：100分

3 最近公共祖先

3.1 算法一

对于每组询问，枚举所有的黑点并暴力计算它们的最近公共祖先，求出最大权值。

时间复杂度：$O(mn^2)$

期望得分：10分

3.2 算法二

对于算法一，使用RMQ优化求LCA的过程，或直接预处理出每两个点之间的LCA。

时间复杂度：$O(mn)$

期望得分：20分

3.3 算法三

对于关于点u的询问，首先若$subtree(u)$中含有黑点，我们可以用$w[u]$来更新答案。此外，u点与其他任何黑点的LCA都只能是它的祖先。于是我们枚举u点的所有祖先节点f，设节点u在f的子节点g所对应的子树中，那么如果$subtree(f) - subtree(g)$中存在黑点v，此时必有LCA(u, v)=f，我们用$w[f]$更新答案即可。

用概率期望相关知识容易证明：本题随机方式所生成的树，树的高度$h = O(\log n)$。

时间复杂度：$O(mh)$

期望得分：40分

3.4 算法四

由于所有修改操作都在询问之前，我们可以考虑直接求出所有节点的答案。

设$f[x]$表示只考虑子树x之外的黑点时LCA的最大权值，那么对于x的所有子节点y，我们可以得到状态转移方程：

$$f[y] = \begin{cases} max(f[x], v[x]), & subtree(x) - subtree(y)\text{中存在黑色节点} \\ f[x], & subtree(x) - subtree(y)\text{中不存在黑色节点} \end{cases}$$

最后，如果子树x中存在黑色节点，我们还要用$w[x]$来更新它的答案。

时间复杂度：$O(m + n)$

期望得分：20分

3.5 算法五

考虑每个节点对询问的贡献，显然一个节点x的权值$w[x]$只可能贡献给x子树中的点。当我们将一个节点x从白色修改为黑色时，首先我们用$w[x]$更新$subtree(x)$中的所有节点。然后依次枚举x的每个祖先节点f，设节点x在f的子节点g所对应的子树中，则$subtree(f) - subtree(g)$中所有节点的答案都能用$w[f]$来更新。同时可以发现，如果子树f在修改操作之前已存在黑色节点，那么用$w[f]$更新完后我们可以直接结束枚举。因为在之前对子树f中的黑点操作时，所有f的祖先节点的贡献值都以相同的方式更新过答案了。这样的话，由于黑点数目只会增加，故总更新次数最多为$n + m$次。

接下来，我们需要考虑如何高效实现子树更新操作。根据DFS序列的性质，一棵子树的所有节点必定对应着序列的一段连续区间。于是，我们按节点的DFS序建一棵线段树来维护每个节点的答案，高效实现区间操作和单点查询即可。

时间复杂度：$O(m \log n + n \log n)$

期望得分：100分