赚奖金 解题报告

绍兴市第一中学 王文涛

1 试题大意

给定一个长度为n的字符串s，你需要对于每个i $(0 \le i < n)$确定一个t_i $(0 \le t_i \le n-i)$。确定t_i的同时你可以获得w_{i,t_i}的奖金，其中对于某个i，w_i以一个分成k段的分段函数的形式给出，每一段的数都相同。接下来给出m“条件”。一个“条件”以这样的形式给出：给定l, r $(0 \le l < r \le n)$和一个整数p，如果对于任意i $(0 \le i < n)$都有$LCP(s[i,n), s[l,r)) \le t_i$(其中$s[i,j)$表示字符串$s$下标从$i$到$j-1$的字符构成的子串，$LCP(a,b)$表示字符串$a,b$的最长公共前缀长度)，就说这个“条件”被满足了，可以获得p元奖金，否则就不能获得奖金。求最多可以获得多少奖金。

2 数据范围

<table>
<tr><th>数据点编号</th><th>$n \le$</th><th>$m \le$</th><th>$\sum k \le$</th><th>数据点编号</th><th>$n \le$</th><th>$m \le$</th><th>$\sum k \le$</th></tr>
<tr><td>1</td><td>10</td><td>10</td><td rowspan="10">无限制</td><td>11</td><td rowspan="4">200</td><td rowspan="4">20100</td><td rowspan="4">无限制</td></tr>
<tr><td>2</td><td>17</td><td>17</td><td>12</td></tr>
<tr><td>3</td><td>20</td><td>20</td><td>13</td></tr>
<tr><td>4</td><td rowspan="7">200</td><td>17</td><td>14</td></tr>
<tr><td>5</td><td rowspan="4">200</td><td>15</td><td rowspan="3">1000</td><td rowspan="2">2000</td><td rowspan="2">2000</td></tr>
<tr><td>6</td><td>16</td></tr>
<tr><td>7</td><td>17</td><td rowspan="4">200000</td><td rowspan="4">400000</td></tr>
<tr><td>8</td><td>18</td><td rowspan="3">200000</td></tr>
<tr><td>9</td><td rowspan="2">20100</td><td>19</td></tr>
<tr><td>10</td><td>20</td></tr>
</table>

3 算法介绍

3.1 算法1

直接枚举所有t_i的值，对每种情况看看能满足哪些条件，计算答案。

时间复杂度$O(n!m)$，期望得分5分。

3.2 算法2

首先我们可以发现，如果对于j, j'满足$j < j'$有$w_{i,j} < w_{i,j'}$，那么取$t_i = j'$必然比取$t_i = j$要优。因此，我们可以修改$w_{i,j}$为$\max_{j \le j' \le n-i} w_{i,j'}$，这样$w_i$数组就满足单调不增了。此时在满足条件不变的情况下$t_i$一定是越小越好。

所以，枚举每个条件满不满足，共2^m种可能的情况，对每种情况求出所有t_i的值计算答案。具体实现时可以按二进制数值顺序依次计算出i $(0 \le i \le n)$

时间复杂度：$O(2^m n)$，期望得分20分。

3.3 算法3

我们发现，t_i越小，w_{i,t_i}越大；t_i越大，满足的条件越多。所以我们就是要找到两者的“平衡”。这让人联想到最小割建模。

由于最小割的目标是最小化割边的流量之和，而原问题的目标是最大化奖金，所以要设法把最大化问题转化为最小化问题。

令网络流图中源点为S，汇点为T。我们对所有i, j $(0 \le i < n,\ 0 \le j \le n - i + 1)$建点$A_{i,j}$。然后，我们从$A_{i,j}$到$A_{i,j+1}$连流量为$10^9 - w_{i,j}$的边，从$S$到$A_{i,0}$连流量为$\infty$的边，从$A_{i,n-i+1}$ 到T连流量为∞的边。这样i对应的一条链上必然会割流量最小的一条边，设这条边是从$A_{i,j}$到$A_{i,j+1}$的，那么我们就令$t_i = j$。这样，割尽可能小的边就对应了权尽可能大的边。

对于第k个条件，我们建点C_k，从S到C_k连流量为p的边。然后对于每个i，从C_k到$A_{i,LCP(s[l,r),s[i,n))}$连流量为∞的边。这样，如果不割S到C_k的边，所有i对应的链上割的边必然在C_k连出去的边之后，也就会满足条件；如果割了S 到C_k的边，也就失去了这个条件的奖金。

因此，我们只要用$10^9 \times n + \sum p$减去整张图的最小割就可以得到答案了。

时间复杂度：$O(maxflow(n^2 + m, n^2 + mn))$，期望得分40分。

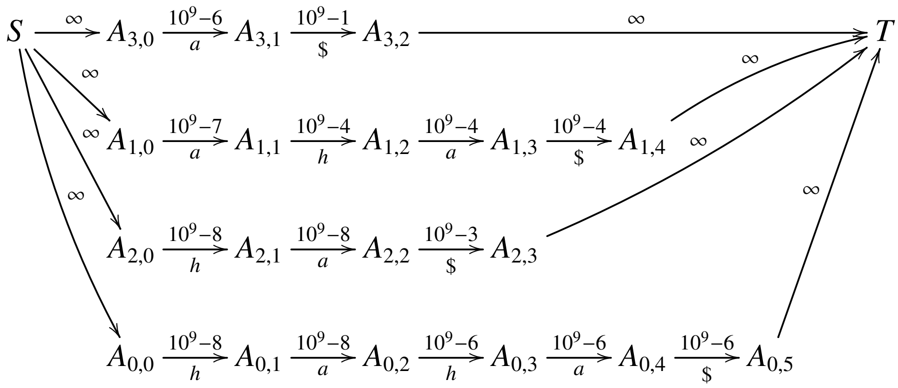
3.4 算法4

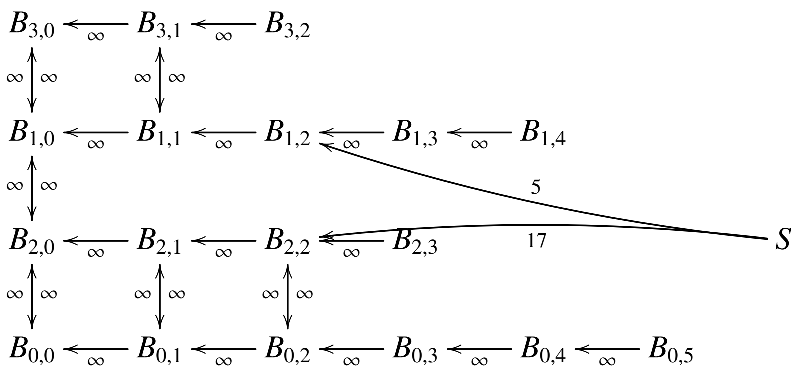
考虑优化算法3的连边。

我们要想办法利用LCP的性质。想一想我们可以用什么来快速地求这类LCP？没错！一种方法是用后缀数组。对于某个条件，我们只要找到后缀$s[l,n)$在后缀数组中的那一行（设为第$rank_l$行），那么从第$rank_l$行往上走，LCP将不断地与$height$取min，也就是LCP会呈现阶梯状单调不增的样子。往下走也是一样。

于是，我们可以用连边来模拟这个过程。我们求出后缀数组。然后对于所有i, j $(0 \le i < n,\ 0 \le j \le n-i+1)$建两个点$A_{i,j}, B_{i,j}$，并从$B_{i,j}$向$A_{i,j}$连流量为$\infty$的边。对于所有$A_{i,j}$与$S, T$这些点之间的连边仍旧按照算法3中的方式。然后，我们从$B_{i,j}$到$B_{i,j-1}$连流量为∞的边。如果对于后缀数组中某两个相邻后缀（设为sa_{i-1}和sa_i）的LCP长度（设为$height_i$），我们对所有j $(0 \le j \le height_i)$，在$B_{sa_{i-1},j}$与$B_{sa_i,j}$之间连双向的流量为∞的边。最后，对于每个条件，我们从S到$B_{l,r-l}$连流量为p的边。整张图就等价于算法3中的图。

以下是样例构出来的图。第一张图是A点的连边，第二张图是B点的连边（没有画出$A_{i,j} \to B_{i,j}$的边）。





时间复杂度：$O(maxflow(n^2, n^2 + m))$，期望得分70分。

3.5 算法5

同样考虑优化算法3的连边。

观察$height$的连边，可以发现相当于给一个后缀区间里所有在某个位置之后的点连边。

于是，我们可以用线段树来代替算法3中的暴力连边。对于一个条件或一个height，建一个B点，找到对应的后缀区间，往函数式线段树上对应区间连边。对于一个条件还要往对应区间左右端点处height对应的B点连边。

时间复杂度：$O(maxflow(n^2 \log n + m, (n^2 + m) \log n))$，期望得分40到80分。

3.6 算法6

考虑继续优化算法5的连边。

对于连续的一段相同权值，只要不影响其他连边，割哪条边都一样，不妨就认为一定是割最后的那条边。

于是，我们用函数式线段树来维护A点的连边。从后往前扫，碰到一个分段函数的分界点就在函数式线段树中插入（当然前后要连边）。对于原来算法5中区间的连边仍然保留。

时间复杂度：$O(maxflow(n + m + \sum k \log n, (n + m) \log n + \sum k))$，期望得分80分。

3.7 算法7

我们重新回过头去想如何利用LCP的性质来优化连边。可以发现后缀树也可以用来求LCP。只是后缀树会把相同的前缀并成一条链，这样看上去不同的后缀割的位置会互相影响，似乎不能像后缀数组那样建图。但实际上是可以的。也就是说，如果我们把后缀数组中关于$height$的连边两端的点都并在一起，整张图仍然是正确的。

我们来证明这个性质。这个性质相当于是说，对于任意两个后缀（设为$s[i, n)$和$s[j, n)$），如果$t_i < LCP(s[i, n), s[j, n))$，那么必有$t_j = t_i$。假设$t_j > t_i$，由于$t_j$肯定是越小越好，所以$t_j$不能变成更小的$t_i$一定是为了满足某个$t_i < r - l \leq t_j$的

条件。可是t_i显然不满足这个条件。因此这样的情况是不存在的。$t_j < t_i$的情况只要把两个后缀交换一下进行上面的推导就好了。这样就证明了这个性质。

时间复杂度：$O(maxflow(n^2, n^2 + m))$，期望得分70分。

3.8 算法8

仔细一看可以发现，这个后缀树上的最小割问题可以直接用树形DP来解决。因为对于某个条件，对应在后缀树上的那个B点会往它所有祖先一路连边，根据A的连边方式可知，该点到根的路径上的所有A与A之间的边都不能割。也就是说，如果该点对应的A点与根连通，就可以获得该条件的奖金p，否则不能获得。因此，DP时可以用f_u表示如果点u的父亲与根连通，u的子树里最多可以获得多少奖金。如果割u到父亲的边，就可以获得相应w的奖金，否则就获得u上的所有奖金p以及所有儿子的f值之和。

对于前17个点，我们可以直接建出$O(N^2)$个点的后缀树，做一遍DP就好了。

时间复杂度：$O(n^2 + m)$，期望得分85分。

3.9 算法9

考虑优化算法8。由于分段函数每一段相同的值如果不会影响其他东西的话割哪一条边都是一样的，所以不需要建出每个点来。我们用后缀自动机或Ukkonen算法或后缀仙人掌建出缩边后的后缀树。对于后缀树中的每个点记录奖金之和p以及它到父亲的边的边权w。然后对于每个分段函数，找到分段点在后缀树上的位置插入一个点。对于每个条件，也找到对应的位置插入一个点，把奖金p累加到这个点上。然后对于每个分段函数，把后缀树上相应的链上所有边都加上相应的权值。这个可以作差然后线性地求一遍子树和。这样就可以在后缀树上DP了。

虽然DP是线性的，但是找位置需要在树上倍增、二分，所以还是有一个log。

时间复杂度：$O(n + m\log n + \sum k(\log n + \log m))$，期望得分100分。

4 总结

本题主要考察了最小割建模和字符串后缀数据结构的应用，难点在于将最

小割建模与字符串后缀数据结构、函数式线段树等结合起来，观察出题目的一些性质，并利用这些性质优化算法。

本题难度中等，预计集训队中约有5人能想到100分算法，一部分人能想到70分算法，所有人都能想到40分算法。