算法设计 Project 实验报告

14307130078 张博洋

一、完成情况

完成了所有 Task。

二、Task1

用动态规划去计算编辑距离即可。

设 a[1...lena], b[1...lenb]是输入的字符串。

动态规划的过程：

设 f[i][j]表示 a 串前 i 个字符与 b 串前 j 个字符的编辑距离，转移方式共三种：

```
f[i][j] = min(
    f[i - 1][j] + 1,   // 将 a 串中的第 i 个字符删去
    f[i][j - 1] + 1,   // 在 a 串的第 i+1 字符前插入字符 b[j]
    f[i - 1][j - 1] + (a[i] != b[j]),
        // 若 a[i]==b[j]则表示不需要编辑
        // 若 a[i]!=b[j]则表示将 a 串的第 i 个字符替换为 b[j]
);
```

边界条件：

```
f[x = 1...lena][0] = x;
f[0][x = 1...lenb] = x;
```

输出编辑方案的办法：

用数组 g[i][j]记录 f[i][j]是从三种转移方式中的哪一种转移而来。在 f 全部计算完成后，从 f[i][j]沿 g[i][j]所指示的方向，一直回溯到 f[0][x]或 f[x][0]为止。然后在沿回溯路径打印编辑方案即可。

复杂度分析：

f 函数计算的范围为 f[0...lena][0...lenb]，每次转移时间为 O(1)，因此总的时间复杂度是 O(lena*lenb)

三、Task2

本题数据范围很大，但是给定的数据很弱。数据中的 de Bruijn 图中实际上只有 20 条长度为 326 的链。因此，此问题变为找 b 串中的一个字串，使得它与 a 串的编辑距离最小。此外，数据中给出的 a 串的长度为 7698，远大于各个 b 串的长度 326，所以有很大概率编辑距离就是 7698-326=7372（事实上，本题答案就是它）。

方法：枚举这 20 个链的每一个子串，调用 Task1 中的程序计算编辑距离，然后取最小者即可。

复杂度分析：枚举每一个子串的复杂度为 O(20*300*300)，计算编辑距离的复杂度为 O(300*7698)，因此总的时间复杂度为 O(20*300*300*300*7698)。实际上由于该算法常数很小，因此可以在一小时内针对此组数据计算出结果。

四、Task3

本题数据范围超大，而且数据不弱，因此必须寻找高效的做法。在与同学交流后，我实现了用 SPFA 在图上进行动态规划的办法：

动态规划的过程：

设 g[i][v]表示 a 串前 i 个字符与到顶点 v 为止的字符串的编辑距离，转移方式为：（设 pre_v 是 v 的前驱，f_v[x][y]表示 a 与顶点 v 对应的字符串的编辑距离）

```
g[i][j] = min(
    g[i - 1][v] + 1,   // 将 a 串中的第 i 个字符删去
    g[i][pre_v] + 1,   // 在 a 串的第 i+1 字符前插入字符 b[j]
    g[i - 1][pre_v] + (a[i] != v[k]),
        // 若 a[i]==v[k]则表示不需要编辑
        // 若 a[i]!=v[k]则表示将 a 串的第 i 个字符替换为 b[j]
    f_v[i][k],  // 只使用顶点 v 对应的字符串
);
```

计算 f_v 数组的办法：

直接计算时间复杂度太高，但是考虑到 k 只有 30 而 a 长度很大，实际上当 a 串很长时，k 将变为 a 的子序列（如同 Task2 中遇到的那样），此时可以直接用 len(a)-len(k)得到编辑距离（全部删除）。具体做法是：在计算过程中检查刚刚计算的编辑距离，如果发现编辑距离与两者坐标之差相等，即不再继续计算。

直接存储所有数组空间复杂度也太高，但是考虑到只使用 f_v[x=1...n][k]，可以省掉大小为 k 的那一维。

输出编辑方案的办法：

在动态规划计算过程中会产生大量的数据，而且只有到计算完成时，才知道具体哪些数据是有用的。由于内存太小无法存储这些数据，所以必须想办法减小数据量。具体做法是：在计算 i=1...100000 的过程中，每隔一段距离 cp=100 就往前回溯一次，把 g[i][v=1...m]这些节点具体回溯到 g[i-cp][v=1...m]中的哪些节点记录下来，然后就可以把 i-cp 到 i 这之间的计算结果释放掉。在所有数据计算完毕后，可以根据上述记录的数据得到大致的回溯路径。之后再次进行第二次计算，在重新计算的过程中，根据之前的记录可以精确回溯出答案路径。这样一来，如果 cp 的值选择合理，**空间复杂度可以降到 O(sqrt(n)*m)**。这样就不需要使用硬盘之类的外存储器了。

运行结果：

如果只计算答案 9539，Task3 程序运行时间为 74 分钟（使用多线程加速等优化）。如果需要输出编辑方案，Task3 程序运行时间为 162 分钟（因为要运行两遍动态规划所以时间加倍），消耗内存约 9GB（可以使用虚拟内存，实际上一台内存 4GB 的计算机也完全可以运行）。

此外由于直接让 Task3 输出编辑方案太麻烦，我采用了让 Task3 输出字符串，然后使用 Task1 程序计算编辑方案的办法。这样做的话，输出方案步骤的时间复杂度是 O(n*n)，空间复杂度是 O(sqrt(n)*n)，实际运行时间 7 分钟，消耗内存约 6GB。