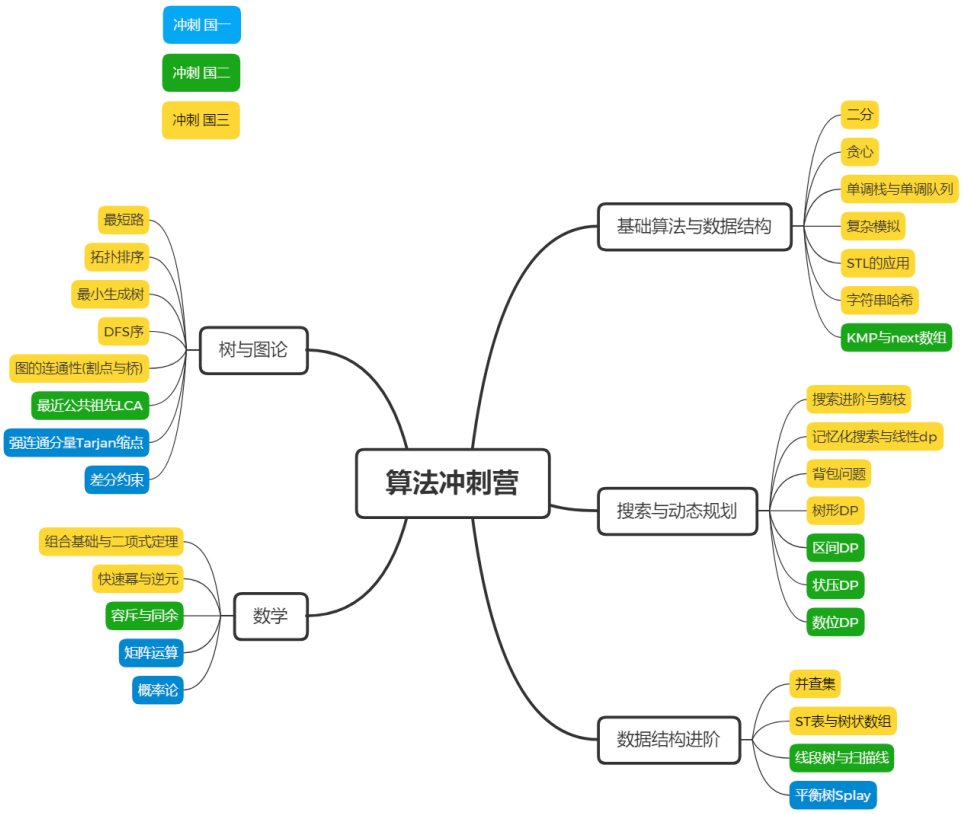
蓝桥杯30天算法冲刺集训



Day-7 线性DP

> *[!NOTE]*
>
> *线性DP没什么好讲的前置知识点的内容，有的话也是在省赛免费讲义里面有，所有这里直接给出四个线性DP题目，帮助大家练习。*

线性DP只需要大家记住两步即可：

1. DP的状态是什么
2. DP的转移怎么转

DP没有唯一解，不同的状态表示，代表了不同的转移。

蓝桥杯真题

最优包含(蓝桥杯C/C++2019B组国赛)

这个题的话我们可以设定$dp[i][j]$代表在S中取出前i个字符与在T中取出前j字符组成最优包含情况下，需要修改S前i个字符串中字符的数量。

我们有了这个状态表示，就需要考虑如何转移了。

我们首先看数据范围返现字符串长度都是小于等于1000的，那么能够确定复杂度是可以到$O(N^2)$级别的，那么我们可以直接两个循环枚举i,j。

然后对于枚举的i,j目前只有两种情况，也就是$S_i=T_j$和$S_i\neq T_j$

我们分类讨论如下：

1. $S_i=T_j$，说明当前的状态直接从之前的状态转移过来就好，不需要进行字符的修改，于是$dp[i][j]=dp[i-1][j-1]$
2. $S_i\neq T_j$，说明要满足当前的状态，需要修改S_i这个字符，那么就可以由两种状态转移过来一种就是$dp[i][j]\leftarrow dp[i-1][j-1]+1$，这一种状态表示我可以直接算上要修改这个字符。还有一种状态是$dp[i][j]\leftarrow dp[i-1][j]$也就是说我可以不修改当前的字符，用之前的$S$中前$i-1$个字符和$T$中前$j$个字符的答案转移即可，相当于这个字符是后面添加上去的

有了状态转移的方式，我们再考虑初始值的情况，首先可以看到是求的最小值的，那么我们先把所有的$dp[i][j]$赋值为一个极大值，既可以用来取min也可以用来表示不合法的状态，也就是说转移结束后还是极大值的话，那么就相当于这个$dp[i][j]$无法满足题目的要求。

接着只需要把$dp[i][0]_{i\in[1,|S|]}$都赋值为一个初始值0即可

```
N = 1007

def main():
    s = input().strip()
    t = input().strip()
    n = len(s)
    m = len(t)
```

```
    dp = [[float('inf')] * (m + 1) for _ in range(n + 1)]
    for i in range(n + 1):
        dp[i][0] = 0

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if s[i - 1] == t[j - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            else:
                dp[i][j] = min(dp[i - 1][j], dp[i - 1][j - 1] +
1)

    print(dp[n][m])

if __name__ == "__main__":
    main()
```

排列数(蓝桥杯C/C++2019B组国赛)

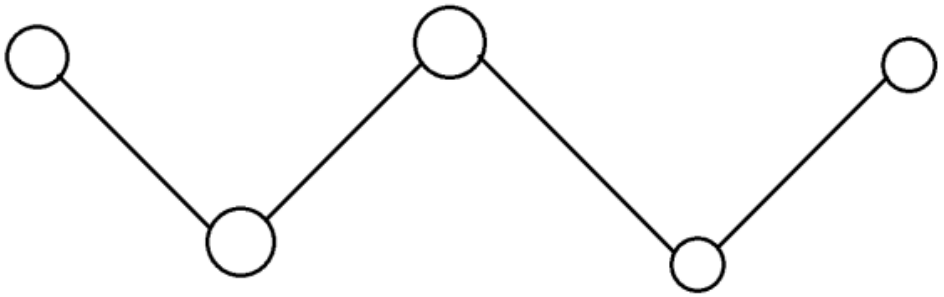
还是首先我们要确定状态是什么，我们可以设$dp[i][j]$代表$1 \sim i$的所有排列中有j个折点。那么我们是不是就可以想到答案其实就是$dp[n][k-1]$了

好的，然后再看转移，首先看数据范围，发现数据范围都是小于等于500的，那么$O(N^2)$级别的算法是足够的，

也就是我们可以直接枚举i,j了，但是在转移之前我们需要明确所有的排列的可能性，也就是如下图所示，分为折点是奇数个和折点是偶数个。

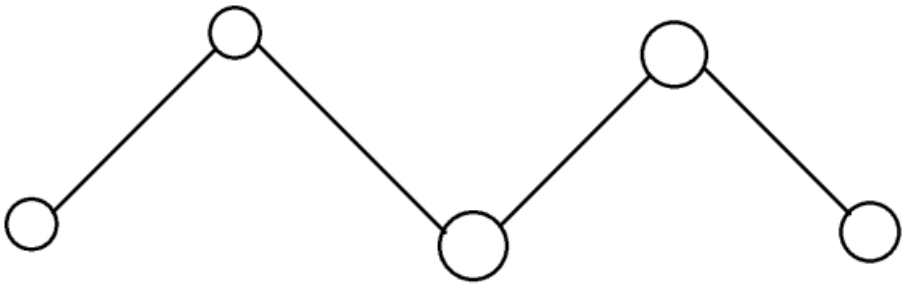
折点为奇数个的情况：

这种情况是有两种的首先是如下所示第一种，我们标号为1：



可以看到这是一个$i=5,j=3$的一种排列的情况，类似W形状，也就是说首位是一下一上的。

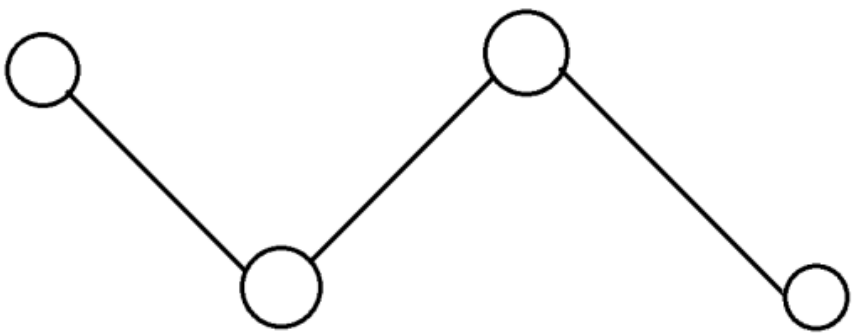
然后把这个图像倒过来就是第二种，我们标号为2:



可以看到这又是另一种$i=5,j=3$的情况，类似M形状，位置也是一上一下的。

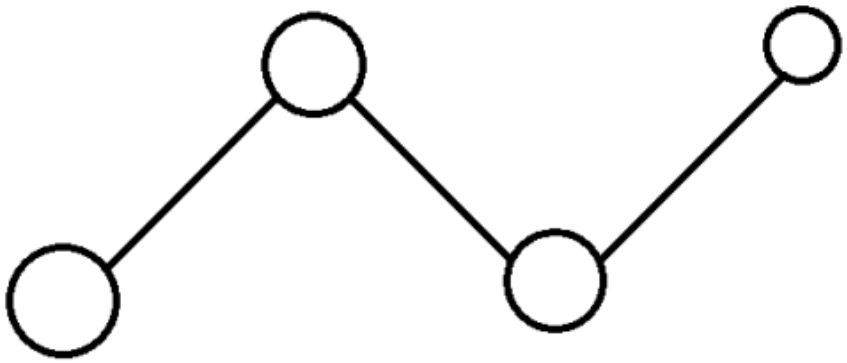
折点为偶数个的情况:

下图所示为折点为偶数个情况的第一种，我们标号为3:



可以看到这是$i=4,j=2$的情况，类似一个N，首位都是同下的。

把图像倒过来就是偶数的第二种情况，我们标号为4：



可以看到这又是另一种$i=4,j=2$的情况，一个N形状，首位都是同上的。

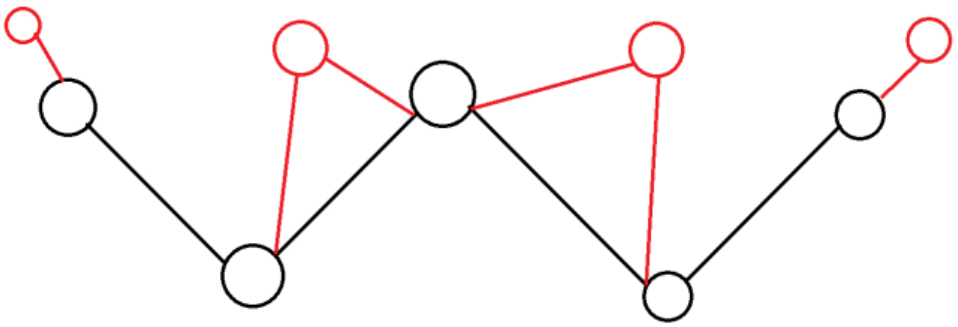
我们把一个位置上大于两边的数称作 山峰 ，一个位置上的数小于两边的数称作 山谷

好的，有了这四种情况，那么怎么转移呢？实际上只有三种转移的情况。

第一种情况：

第一种情况就是我们插入一个新的数$i+1$，对折点的变换没有影响，也就是可以$dp[i+1][j] \leftarrow dp[i][j]$这种转移方式，那么这种方式怎么转移呢？

对于标号1，实际上是不是就是在山峰处和首位加上我们的$i+1$就可以了，也就是如下图所示



对于标号2，也是类似情况，但是首位不能加入$i+1$了，不过又多了一个山峰，所以总共的个数还是不变的

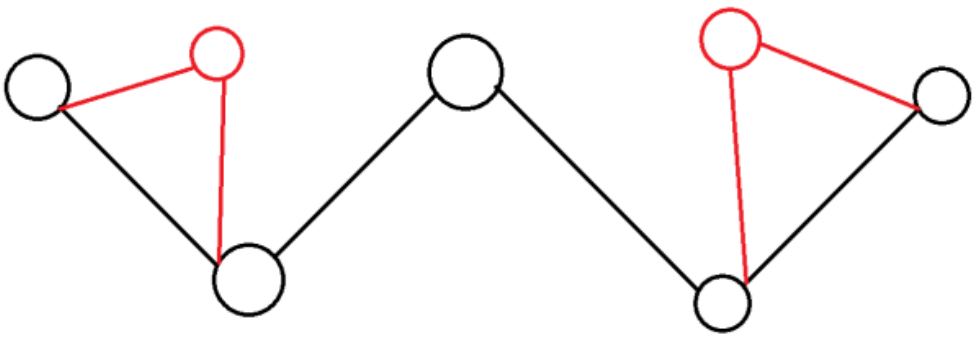
所以可以看出对于j是奇数的情况，$dp[i+1][j]=dp[i][j]\times(j+1)$其中$j+1$对于标号1来说就是$\lfloor\frac{j}{2}\rfloor\times 2+2$，对于标号2就是$\lceil\frac{j}{2}\rceil\times 2$其中$\lfloor\frac{j}{2}\rfloor$和$\lceil\frac{j}{2}\rceil$只差1所以，最终的贡献都是$j+1$

对于标号3和标号4也可以用类似的方法推理出来，留给同学们自己推理，不再赘述。

第二种情况：

第二种情况就是我们插入一个新的数$i+1$，对折点的变换有影响，且只增加一个折点，也就是可以$dp[i+1][j+1]\leftarrow dp[i][j]$这种转移方式，那么这种方式怎么转移呢？

还是我们看对于标号1来说是不是就如下图所示



可以看到直接在最开始和最末尾的两条边上加上，那么折点个数就是多一个，所以最终贡献为2

对于标号2来说，就是首尾各加上一个，最终贡献还是2，所以可以看出$dp[i+1][j+1]=dp[i][j]\times 2$

对于标号3和标号4也可以用类似的方法推理出来，留给同学们自己推理，不再赘述。

第三种情况：

第二种情况就是我们插入一个新的数$i+1$，对折点的变换有影响，且增加两个折点，也就是可以$dp[i+1][j+2]\leftarrow dp[i][j]$这种转移方式，那么这种方式怎么转移呢？

首先我们可以明确增加一个节点最多只有这三种情况，于是我们就可以直接用总共的折点摆放的可能性减去之前两种情况的即可，总共的折点的摆放的可能性是不是就是$i-1+2=i+1$种可能，也就是说$i-1$个间隔以及首尾都可以放的2个

所以可以看出$dp[i+1][j+2]=dp[i][j]\times(i+1-(j+1)-2)=dp[i][j]\times(i-j-2)$

OK，剩下就是初始状态了，初始状态是不是直接$dp[1][0]=1$然后$dp[i][0]_{i\in[2,n]}=2$即可

```python
MOD = 123456

def mod(a):
    return a % MOD

def main():
    n, k = map(int, input().split())
    dp = [[0] * 505 for _ in range(505)]
    dp[1][0] = 1

    for i in range(2, n):
        dp[i][0] = 2
        for j in range(i + 1):
            dp[i + 1][j] += mod(dp[i][j] * (j + 1))
            dp[i + 1][j + 1] += mod(dp[i][j] * 2)
            dp[i + 1][j + 2] += mod(dp[i][j] * (i - j - 2))

    print(dp[n][k - 1] % MOD)

if __name__ == "__main__":
    main()
```

练习题

小蓝与翻转游戏

小蓝与高速公路