Part #0 区间 dp 是什么

关于动态规划，其实说白了，就是一种递推。当我们解决大问题的时候，先把它 **分解** 为若干个子问题，再把它 **合并** 成当前所需的结果。有点类似于递归的感觉，而递归会重复计算，普通 dp 与记忆化搜索不会。

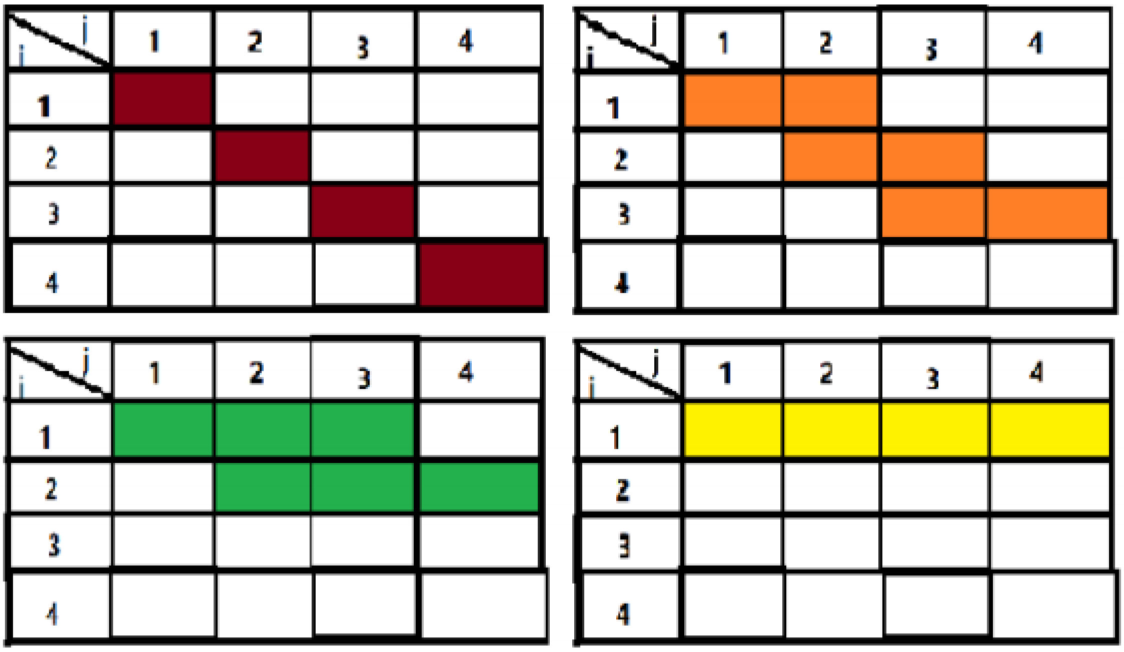
而区间 dp，就是这类问题中的一小类。在我们的 **分解** 操作时，它需要取任意区间的子问题结果。这类问题就叫做区间 dp。

对于大多数问题，区间 dp 的架构像这样：$\mathrm{dp}_{i,j}$ 被拆分为若干个 $\mathrm{dp}_{l,r}$（其中 $i \le l \le r \le j$），通过这些小区间计算出来的 dp 值推算 $\mathrm{dp}_{i,j}$。 而 dp 的两个下标代表一个区间的左端点与右端点。

具体问题具体分析，我们当然需要结合问题来进行学习。然而，区间 dp 是有“套路”的，我们接下来把一个个花里胡哨的题目，透过现象去看它的本质，然后把它们 **归类** 为一个个套路。

Part #1 区间 dp 架构

通常，我们在做此类问题时，是由小区间推到为大区间。所以，我们要先枚举小区间，再枚举大区间，也就是 **枚举长度** 。我们的 DP 顺序应是这样的：



伪代码：

```
# 初始化 dp[i][i] 或 dp[i][i-1] 因题而异
for len=start to len=end len++ # 枚举长度，start 和 end 分别为最短与最长的长度
    for i=1,j=len to j=n i++,j++ # 枚举对应长度的区间 (i,j)
        ( for k=i to k=j-1 k++ ) # 可能要枚举中间点
            # 进行 dp 操作
```

还有一种方法也是可以的，这篇文章中“关路灯”类题目也用了这种枚举方法。（感谢金钩爷 @SharpnessV）

首先我们看这种方法为什么不行：

```
for(int i=1;i<=n;i++)
    for(int j=i;j<=n;j++)
        // dp
```

假设我们程序运行到 $i=1,j=5$，转移到区间 $(1,3)$ 和 $(4,5)$ 的时候，$i=4,j=5$ 还没有运行。

我们得出结论：将左端点枚举到 i 的时候，比 i 编号大的左端点一定要先枚举到。所以一种新的做法出现了：

```
for(int i=n;i>=1;i--)
    for(int j=i;j<=n;j++)
        //dp
```

我们倒序枚举左端点，就可以保证 dp 顺序正确。

Part #2 神奇的“套路”们

Part #2.1 石子合并类套路

此类问题，是常见的找中间点 k 分割的问题。

- 例 2.1.1 [ZhimaOI #201 合并石子（未注册 zhimaoi.cn 的人请注册）]

首先，对于一个区间 (i,j) ，我们把这个区间内所有石子给合并起来的上一步是啥？

1. 将区间 (i,k) 之间的所有石子全部合并（其中 $i\le k<j$）；
2. 将区间 $(k+1,j)$ 之间的所有石子全部合并。

我们将这两次合并的结果相加，再加上 (i,j) 之间所有值的和即可。当然，这个 k 还得枚举来选择最优的。

得转移方程式：$\text{dp}_{i,j}=\min\{\text{dp}_{i,k}+\text{dp}_{k+1,j}+sum_{i,j}\}\quad(i\le k<j)$

- 例 2.1.2 [Luogu P1880 [NOI1995] 石子合并]

和例 2.1.1 似乎是一样的，然而它是一个环。我们的做法是 **破环为链** ，将这个数列重复两遍，取区间长度为 n 的最值，这样“环”就被我们考虑到了。

因为这题和上题相似，所以我只提供这题代码。

- 例 2.1.3 [Luogu P3146 [USACO16OPEN]248 G]

和上述题目十分相似。切断区间 (i,j) 的 k ，如果有 $\text{dp}_{i,k}=\text{dp}_{k+1,j}$ ，那么可以合并。得转移方程式：

$$\text{dp}_{i,j}=\max\{\text{dp}_{i,k}+1\}\quad(i\le k<j,\text{dp}_{i,k}=\text{dp}_{k+1,j})$$

- 例 2.1.4 [HDU 4283 You Are the One]

比较难的题目了。

在区间 (i,j) 内，我们首先确定这个 i 的不高兴的值。这玩意可以枚举！

假设取中间点 k，区间 (i,k) 的人进栈，显然 i 在这些人中最后一个出栈，等了 $k-i$ 个人，这些人的值我们已经算好了。区间 $(k+1,j)$ 的人自然需要傻傻地等待 (i,k) 之间的所有人，就是 $k-i+1$ 个人。这个值要加进去。所以 $\text{dp}_{i,j}$ 是最小的 $\text{dp}_{i+1,k}+\text{dp}_{k+1,j}+cost(i,i)\times(k-i)+cost(k+1,j)\times(k-i+1)$（$cost(i,j)$ 表示 $i\sim j$ 这些人的权值和，k 为区间内一个数）。

具体看本题提供的代码。

- 例 $2.1.5$ CodeForces 149D Coloring Brackets

数组要记录区间左右端点的染色情况。

如果左右端点是一对匹配的括号，那么推到 $(i+1,j-1)$，再分类讨论。

否则，推到 (i,k) 和 $(k+1,j)$ ，其中 k 是与 i 匹配的括号。再分类讨论。具体的转移方程不是特别难，根据题意可以自行推导。

提供本题代码。

Part #2.2 取一端/两端套路

- 例 $2.2.1$ POJ 2955 Brackets

这题我是真不知道归 2.1 还是归类 2.2 ，其实都有。

它和石子合并大致相同。可以枚举 (i,j) 中间点 k，得 $\text{dp}_{i,j}=\max\{\text{dp}_{i,k}+\text{dp}_{k+1,j}\}$ 。但，要注意，如果左端点与右端点括号匹配，长度可以是抹掉左右端点的值加上 2，因为这对括号可以用了。特判一下即可。

- 例 $2.2.2$ Luogu P1005 [NOIP2007 提高组] 矩阵取数游戏

取一端的经典问题。从小区间推向大区间。转移方程 $\text{dp}_{i,j}$ 会转移到 $\text{dp}_{i,j-1}$ 或者 $F_{i+1,j}$ 。也可以是大区间推向小区间：$\text{dp}_{i-1,j}$ 和 $\text{dp}_{i,j+1}$。本题以大区间推向小区间来讲。

问题如何知道一个数字取得时候是第几个取的。如果取的是第 $i-1$ or $j+1$ 个，则是区间外的数量，即 $m-j+i-1$ 。

注意这题需要做 n 次 dp，并且需要高精度/`__int128`。

- 例题 $2.2.3$ Luogu P3205 [HNOI2010]合唱队

也是取一端的经典问题。但是这题要考虑两种情况，一种是一个区间内最后插入左端点，一个是最后插入右端点。当然一个转移为 $(i+1,j)$，一个转移为 $(i,j-1)$。在转移过去的时候，左/右端点在符合条件的情况下均可以，需要判断。

- 例题 $2.2.4$ POJ 3280 Cheapest Palindrome

非常古老的问题了。

如果区间 (i,j) 的左端点与右端点相同，直接推锅给 $(i+1,j-1)$。

否则分四种情况：

1. 在 i 前插入 str_j，那么推锅给 $(i,j-1)$
2. 删掉 str_j ，推锅给 $(i,j-1)$
3. 在 j 后面插入 str_i，那么推锅给 $(i+1,j)$
4. 删掉 str_i ，推锅给 $(i+1,j)$。

计算一下每次加上的权值，然后就做完了。

- 例 $2.2.5$ ZhimaOI #520 队列（未注册 zhimaoi.cn 的人请注册）

恰，很有意思嘛。

首先贪心肯定是不对滴。Hack Data ：

```
4
1 6 1000000 5
```

我们对于每一个区间 (i,j) 开两个数组，一个记录 **当前小明取的情况**，一个记录 **当前小华取的情况** 。值是小明能够得到的最多的和。然后分类讨论：

1. 小明则希望自己能得多的，于是在 $\mathrm{dp}_{i,j-1}+a_j$ 和 $\mathrm{dp}_{i+1,j}+a_i$ 之间取最大值。注意这里的 dp 是小华取的情况。
2. 小华则希望小明得的少，于是在 $\mathrm{dp}_{i,j-1}$ 与 $\mathrm{dp}_{i+1,j}$ 里面取最小值。注意这里的 dp 是小明取的情况。

因为分小华和小明，所以我们要分两个 dp 数组。

Part #2.3 关路灯套路

- 例 2.3.1 Luogu P1220 关路灯

分两种情况：老王在左端点；老王在右端点。转移到的区间决定于老王在的位置。老王这么做显然会让区间外以及现在要关的路灯都等相应的时间。

设 $F_{i,j,0/1}$ 分别为老王在 i/j 处，关掉 (i,j) 之间的灯所需最小耗能。转移方程我则以代码形式表示：

```
f[i][j][0]=min(f[i][j][0],f[i+1][j][0]+(a[i+1]-a[i])*(sum[i]+sum[n]-sum[j]));
f[i][j][0]=min(f[i][j][0],f[i+1][j][1]+(a[j]-a[i])*(sum[i]+sum[n]-sum[j]));
f[i][j][1]=min(f[i][j][1],f[i][j-1][1]+(a[j]-a[j-1])*(sum[i-1]+sum[n]-sum[j-1]))
f[i][j][1]=min(f[i][j][1],f[i][j-1][0]+(a[j]-a[i])*(sum[i-1]+sum[n]-sum[j-1]));
```

其中，$sum_k=\sum\limits_{i=1}^{k} b_i$，$b_i$ 则为功率，a_i 为位置。

- 例 2.3.2 ZOJ 3469

和上一题思路一模一样！`Code` 不给了。

- 例 2.3.3 Luogu P2858 [USACO06FEB]Treats for the Cows G/S

也是关路灯类问题。注意：这题的起点可以是任意位置哦！

设 $\mathrm{dp}_{i,j}$ 表示取掉了 (i,j) 之间的零食获得的最大权值。注意我们倒推做，$\mathrm{dp}_{i,i}$ 是第 i 个零食最后一个拿，长度为 2 的区间的 dp 是这两个零食最后拿的最大全职。

对于长度为 len 的区间 (i,j) ，我们可以求得左/右端点是第 $(n-len+1)$ 个取的。这样就很好计算了。

Part #2.4 “找对象”套路

（相信你现在可以自己推导转移方程了！）

先解释一下：这类题目与三元组有关，所以我们可以给端点 (i,j) 找一个合适的 k 组成三元组。

- 例 2.4.1 LibreOJ #10149. 「一本通 5.1 例 3」凸多边形的划分

对于区间 (i,j) ，我们找一个点 k 与其组成三角形，计算出结果，然后讲区间划分为 (i,k) 与 (k,j)（画图试一下）。然后就是合并果子了？！

注意：需要使用 `__int128`。

- 例 2.4.2 POJ 1651 Multiplication Puzzle

$\mathrm{dp}_{i,j}$ 表示删掉 $(i+1,j-1)$ 之间的区间的数字的最大获得权值。我们找到和 i,j 一起删掉的 k。

首先要删掉 $(i+1,k-1)$，然后删掉 $(k+1,j-1)$（两个 dp 值加上），再加上 $a_i \times a_j \times a_k$，就是当前 k 所得到的权值，枚举 k 即可。 k 是中间的切断点。

- 例 $2.4.3$ HDU 5115 Dire Wolf

肥肠简单，和上一题一样。还是有些代码细节，所以提供本题代码。

Part #2.5 涂色套路

- 例 $2.5.1$ Luogu P4170 [CQOI2007]涂色

如果左右端点相同，涂 i 的时候可以顺带涂一下 j；涂 j 的时候可以顺带涂一下 i。也就是，一个端点可以忽略。否则，枚举中转点涂。

- 例 $2.5.2$ HDU 2476 String painter

先考虑空串变为 B 串。对于区间 (i,j)，如果有中转点 k，它的字母等于 j，那么涂 $(k,j-1)$ 的时候可以顺带涂一下 j。区间 $(i,k-1)$ 再考虑。

但是在 A 串的基础上就麻烦了。对于一个本来就匹配的字母，我们可刷可不刷。具体的操作，我们可以用 a_i 表示前 i 个字母转换的值。如果 $A_i = B_i$，a_i 可以是 a_{i-1}；否则仅仅可以是 $a_k + \mathrm{dp}_{k+1,i}$。

- 例 $2.5.3$ LightOJ 1422 Halloween Costumes

欢迎大家在 这里 提交。

看着不像涂色，可是它就是涂色！

首先，对于区间 (i,j)，我们找到要穿的衣服与 j 相同的 k。我们先转移到区间 (i,k)，这个值先加上。要注意：此时穿着的衣服是第 k 次演出所需衣服，和 j 相同。我们再加上区间 $(k+1,j-1)$。为啥是 $j-1$？因为 j 的衣服已经被 k 准备好了，到时候不断脱衣服直到 k 露出来。

这篇文章结束了！这是博客版文档，所有参考代码请在 这里 下载，或者登录洛谷，在 云剪切板 中查看。