Dynamic Programming Revisited

Roundgod

Nanjing University

wcysai@foxmail.com

July 21, 2019

How to solve it?

- 解决动态规划问题，一般来说有两个步骤:

How to solve it?

- 解决动态规划问题，一般来说有两个步骤:
- 设计状态
  - 理解了问题的本质才能设计出正确的状态
  - 根据题目限制设计合适的状态(背包问题的$O(nW)$dp以及$O(n\sum C)$dp)
  - 状态定义和转移的时候可能涉及到一些trick
  - 复杂度为所有**可到达状态**的转移复杂度之和

How to solve it?

- 解决动态规划问题，一般来说有两个步骤:
- 设计状态
  - 理解了问题的本质才能设计出正确的状态
  - 根据题目限制设计合适的状态(背包问题的$O(nW)$dp以及$O(n\sum C)$dp)
  - 状态定义和转移的时候可能涉及到一些trick
  - 复杂度为所有**可到达状态**的转移复杂度之和
- 复杂度优化
  - 数据结构维护,斜率优化,分治优化,Knuth优化,wqs二分.....
  - 这部分是在确定了状态以及转移之后的
  - 有时根据题目的限制，也会有特殊的优化方式

Network Rumour

- 你拥有一个由N台电脑组成的网络，对于每对电脑(i,j)你知道将一个文件从i 传输到j所需要的时间$a_{i,j}$。注意将文件从i传输到j所需要的时间和将文件从j传输到i所需要的时间可能不同，你不能同时让一个电脑向多台电脑传输文件。

Network Rumour

- 你拥有一个由N台电脑组成的网络，对于每对电脑(i, j)你知道将一个文件从i 传输到j所需要的时间$a_{i,j}$。注意将文件从i传输到j所需要的时间和将文件从j传输到i所需要的时间可能不同，你不能同时让一个电脑向多台电脑传输文件。
- 初始时在1号电脑上有一个重要文件，你想要把这个文件传输到全部N台电脑上。当一个电脑收到文件的时候，它可以立即开始将这个文件传输给其他电脑，你需要计算将文件传输到全部N台电脑上所需的最短时间。

Network Rumour

- 你拥有一个由N台电脑组成的网络，对于每对电脑(i, j)你知道将一个文件从i 传输到j所需要的时间$a_{i,j}$。注意将文件从i传输到j所需要的时间和将文件从j传输到i所需要的时间可能不同，你不能同时让一个电脑向多台电脑传输文件。
- 初始时在1号电脑上有一个重要文件，你想要把这个文件传输到全部N台电脑上。当一个电脑收到文件的时候，它可以立即开始将这个文件传输给其他电脑，你需要计算将文件传输到全部N台电脑上所需的最短时间。
- Constraints: $2 \leq N \leq 13, 0 \leq a_{i,j} \leq 10000$
- Time Limit: 1s, Memory Limit: 128MB
- Source: CSAcademy IOI 2016 Training Round #3 Network Rumour

Network Rumour

- 考虑整个传输的过程，可以用一个根节点为1的有根树来表示。

Network Rumour

- 考虑整个传输的过程，可以用一个根节点为1的有根树来表示。
- 我们令$dp_{i,mask}$表示一棵根为i，子树包含所有在$mask$中的节点的树所需的最短时间

Network Rumour

- 考虑整个传输的过程，可以用一个根节点为1的有根树来表示。
- 我们令$dp_{i,mask}$表示一棵根为i，子树包含所有在$mask$中的节点的树所需的最短时间
- 枚举第一棵子树的$submask$以及根节点进行转移，转移的复杂度为$O(n^2 \cdot 3^n)$.

Labeled Connected Graphs

- 给定n和m,你需要计算，对于所有不同的n个顶点的带标号的连通图，顶点1 和顶点n的最短距离之和是多少，答案对m取模。

Labeled Connected Graphs

- 给定n和m,你需要计算，对于所有不同的n个顶点的带标号的连通图，顶点1 和顶点n的最短距离之和是多少，答案对m取模。
- Constraints: $2 \leq n \leq 400, 10^6 + 3 \leq m \leq 10^9 + 9$,保证$m$是一个质数。
- Time Limit: 2s, Memory Limit: 512MB
- Source: Bytedance — Moscow Workshops ICPC Programming Camp, The Finals (from Moscow SU Red Panda), L, Labeled Connected Graphs

Labeled Connected Graphs

- 固定顶点1，可以看出其他顶点都是对称的，故题目可以转化为: 对于所有不同的n个顶点的带标号的连通图,计算所有顶点和顶点1的距离之和，最后将答案除以$n-1$即可。

Labeled Connected Graphs

- 固定顶点1，可以看出其他顶点都是对称的，故题目可以转化为: 对于所有不同的n个顶点的带标号的连通图,计算所有顶点和顶点1的距离之和，最后将答案除以$n-1$即可。
- 考虑到最短路的性质，我们以分层的方式构造这个图。

Labeled Connected Graphs

- 可以发现，如果枚举每层添加的顶点个数进行转移的话，我们需要知道上一层的顶点个数来计算两层之间加边的方式，同时还需要知道当前是第几层来计算对答案的贡献。

Labeled Connected Graphs

- 可以发现，如果枚举每层添加的顶点个数进行转移的话，我们需要知道上一层的顶点个数来计算两层之间加边的方式，同时还需要知道当前是第几层来计算对答案的贡献。
- 于是很容易想到令$f_{i,j,k}$和$g_{i,j,k}$分别表示当前使用了i个顶点，在第j层，上一层的顶点数是k的图的个数以及答案

Labeled Connected Graphs

- 可以发现，如果枚举每层添加的顶点个数进行转移的话，我们需要知道上一层的顶点个数来计算两层之间加边的方式，同时还需要知道当前是第几层来计算对答案的贡献。
- 于是很容易想到令$f_{i,j,k}$和$g_{i,j,k}$分别表示当前使用了i个顶点，在第j层，上一层的顶点数是k的图的个数以及答案
- 枚举前一层的顶点个数进行转移，复杂度是$O(n^4)$,不能通过。

Labeled Connected Graphs

- trick: 定义贡献的时候，假设当前是在第j层，那么对于所有还未添加到图中的顶点，令其对答案的贡献为j,这样就可以不用记录当前的层数，可以消去一维。

Labeled Connected Graphs

- trick: 定义贡献的时候，假设当前是在第j层，那么对于所有还未添加到图中的顶点，令其对答案的贡献为j,这样就可以不用记录当前的层数，可以消去一维。
- 令$f_{i,j}$和$g_{i,j}$分别表示当前使用了i个顶点，上一层的顶点数是j的图的个数以及答案

Labeled Connected Graphs

- trick: 定义贡献的时候，假设当前是在第j层，那么对于所有还未添加到图中的顶点，令其对答案的贡献为j,这样就可以不用记录当前的层数，可以消去一维。
- 令$f_{i,j}$和$g_{i,j}$分别表示当前使用了i个顶点，上一层的顶点数是j的图的个数以及答案

$$f_{i,j} = \sum_{k=1}^{i-j} f_{i-j,k} \cdot (2^k - 1)^j \cdot 2^{\frac{j(j-1)}{2}} \cdot \binom{n-(i-j)}{j}$$

$$g_{i,j} = f_{i,j} \cdot (n - i + j) + \sum_{k=1}^{i-j} g_{i-j,k} \cdot (2^k - 1)^j \cdot 2^{\frac{j(j-1)}{2}} \cdot \binom{n-(i-j)}{j}$$

- 枚举前一层的顶点个数进行转移，复杂度是$O(n^3)$,可以通过。

Telegraph

- 你现在想要用.和_构造一种对于N个字符的编码，使得不存在两个不同的字符A和B，对于其中一个的编码是另一个编码的前缀。例如如果A的编码是._,那么B的编码可以是_._, 但不能是._.
- 已知发送一个.需要一秒，发送一个_需要两秒，现在给定一份报文中N种字符的出现频率，第i种字符的出现频率是f_i.请问在最优编码下，发送这份报文所需的时间是多少?

Telegraph

- 你现在想要用.和_构造一种对于N个字符的编码，使得不存在两个不同的字符A和B，对于其中一个的编码是另一个编码的前缀。例如如果A的编码是._,那么B的编码可以是_._, 但不能是._.
- 已知发送一个.需要一秒，发送一个_需要两秒，现在给定一份报文中N种字符的出现频率，第i种字符的出现频率是f_i.请问在最优编码下，发送这份报文所需的时间是多少?
- Constraints: $1 \leq N \leq 750, 1 \leq f_i \leq 10^5$
- Time Limit: 1s, Memory Limit: 128MB
- Source: CSAcademy IOI 2016 Training Round #3 Telegraph

Telegraph

- 一种编码可以表示为一个二叉树，树的边权为1或2。每个叶子节点表示一个字符，字符的编码由根到它的路径表示，发送单个字符所需要的时间是它到根的距离。

Telegraph

- 一种编码可以表示为一个二叉树，树的边权为1或2。每个叶子节点表示一个字符，字符的编码由根到它的路径表示，发送单个字符所需要的时间是它到根的距离。
- 很容易证明我们应当把到根距离短的叶子节点分配给出现频率高的字符，把到根距离长的叶子节点分配给出现频率低的字符。

Telegraph

- 一种编码可以表示为一个二叉树，树的边权为1或2。每个叶子节点表示一个字符，字符的编码由根到它的路径表示，发送单个字符所需要的时间是它到根的距离。
- 很容易证明我们应当把到根距离短的叶子节点分配给出现频率高的字符，把到根距离长的叶子节点分配给出现频率低的字符。
- 另一个有用的observation是这棵树应该是严格二叉的: 每个节点要么是叶子，要么恰好有两个孩子。

Telegraph

- 一种编码可以表示为一个二叉树，树的边权为1或2。每个叶子节点表示一个字符，字符的编码由根到它的路径表示，发送单个字符所需要的时间是它到根的距离。
- 很容易证明我们应当把到根距离短的叶子节点分配给出现频率高的字符，把到根距离长的叶子节点分配给出现频率低的字符。
- 另一个有用的observation是这棵树应该是严格二叉的: 每个节点要么是叶子，要么恰好有两个孩子。
- 于是我们很容易想到$dp_{i,a,b,l}$表示建造一棵深度为i,其中有a个叶子到根的距离为$i-1$,b个叶子到根的距离为i,在前$i-1$层有l个叶子时的最优答案，枚举a个叶子中保留为叶子的个数进行转移，时间复杂度是$O(n^5)$.

Telegraph

- 利用刚才介绍过的trick，可以不用记录当前树的深度，将未添加的叶子到根的距离视为当前根的深度即可，这样复杂度可以优化至$O(n^4)$.

Telegraph

- 利用刚才介绍过的trick，可以不用记录当前树的深度，将未添加的叶子到根的距离视为当前根的深度即可，这样复杂度可以优化至$O(n^4)$.
- another trick: 因为同一深度叶子不可区分，我们可以假设保留为叶子的是一些叶子的前缀。这样可以不用枚举保留为叶子的个数进行转移，而是每次枚举一个叶子进行转移。

Telegraph

- 利用刚才介绍过的trick，可以不用记录当前树的深度，将未添加的叶子到根的距离视为当前根的深度即可，这样复杂度可以优化至$O(n^4)$.
- another trick: 因为同一深度叶子不可区分，我们可以假设保留为叶子的是一些叶子的前缀。这样可以不用枚举保留为叶子的个数进行转移，而是每次枚举一个叶子进行转移。
- 令$dp_{a,b,l}$表示a个叶子在倒数第二层，b个叶子在最后一层，在之前层有$l-1$个叶子的答案。转移的时候我们分两种情况:

Telegraph

- 利用刚才介绍过的trick，可以不用记录当前树的深度，将未添加的叶子到根的距离视为当前根的深度即可，这样复杂度可以优化至$O(n^4)$.
- another trick: 因为同一深度叶子不可区分，我们可以假设保留为叶子的是一些叶子的前缀。这样可以不用枚举保留为叶子的个数进行转移，而是每次枚举一个叶子进行转移。
- 令$dp_{a,b,l}$表示a个叶子在倒数第二层，b个叶子在最后一层，在之前层有$l-1$个叶子的答案。转移的时候我们分两种情况:
- Case 1: 倒数第二层的第一个叶子保留，转移到$dp_{a,b-1,l+1}$.
- Case 2: 倒数第二层的所有叶子拓展到下面两层，转移到$dp_{a+b,a,l}$

Telegraph

- 利用刚才介绍过的trick，可以不用记录当前树的深度，将未添加的叶子到根的距离视为当前根的深度即可，这样复杂度可以优化至$O(n^4)$.
- another trick: 因为同一深度叶子不可区分，我们可以假设保留为叶子的是一些叶子的前缀。这样可以不用枚举保留为叶子的个数进行转移，而是每次枚举一个叶子进行转移。
- 令$dp_{a,b,l}$表示a个叶子在倒数第二层，b个叶子在最后一层，在之前层有$l-1$个叶子的答案。转移的时候我们分两种情况:
- Case 1: 倒数第二层的第一个叶子保留，转移到$dp_{a,b-1,l+1}$.
- Case 2: 倒数第二层的所有叶子拓展到下面两层，转移到$dp_{a+b,a,l}$
- 这样时间复杂度变为$O(n^3)$,可以通过本题。

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:
- 如果所有格子的颜色都是黑色或者所有格子的颜色都是白色，那么这个网格图的complexity是0
- 否则，用竖线或者横线将网格图划分成两部分，假设划分之后两个网格图的complexity分别是c_1和c_2,令m表示所有的划分方式下$\max c_1, c_2$的最小值，那么这个网格图的complexity是$m+1$。

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:
- 如果所有格子的颜色都是黑色或者所有格子的颜色都是白色，那么这个网格图的complexity是0
- 否则，用竖线或者横线将网格图划分成两部分，假设划分之后两个网格图的complexity分别是c_1和c_2,令m表示所有的划分方式下$\max c_1, c_2$的最小值，那么这个网格图的complexity是$m+1$。
- 给定一个$H \times W$的网格图，计算它的complexity

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:
- 如果所有格子的颜色都是黑色或者所有格子的颜色都是白色，那么这个网格图的complexity是0
- 否则，用竖线或者横线将网格图划分成两部分，假设划分之后两个网格图的complexity分别是c_1和c_2,令m表示所有的划分方式下$\max c_1, c_2$的最小值，那么这个网格图的complexity是$m+1$。
- 给定一个$H \times W$的网格图，计算它的complexity
- Constraints: $1 \leq H, W \leq 165$
- Time Limit: 5s, Memory Limit: 512MB
- Source: Atcoder Grand Contest 033 D Complexity

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:
- 如果所有格子的颜色都是黑色或者所有格子的颜色都是白色，那么这个网格图的complexity是0
- 否则，用竖线或者横线将网格图划分成两部分，假设划分之后两个网格图的complexity分别是c_1和c_2,令m表示所有的划分方式下$\max c_1, c_2$的最小值，那么这个网格图的complexity是$m+1$。

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:
- 如果所有格子的颜色都是黑色或者所有格子的颜色都是白色，那么这个网格图的complexity是0
- 否则，用竖线或者横线将网格图划分成两部分，假设划分之后两个网格图的complexity分别是c_1和c_2,令m表示所有的划分方式下$\max c_1, c_2$的最小值，那么这个网格图的complexity是$m+1$。
- 给定一个$H \times W$的网格图，计算它的complexity

Complexity

- 对于一个长方形的网格图，每个格子要么染成黑色要么染成白色，我们定义它的complexity如下:
- 如果所有格子的颜色都是黑色或者所有格子的颜色都是白色，那么这个网格图的complexity是0
- 否则，用竖线或者横线将网格图划分成两部分，假设划分之后两个网格图的complexity分别是c_1和c_2,令m表示所有的划分方式下$\max c_1, c_2$的最小值，那么这个网格图的complexity是$m+1$。
- 给定一个$H \times W$的网格图，计算它的complexity
- Constraints: $1 \leq H, W \leq 165$
- Time Limit: 5s, Memory Limit: 512MB
- Source: Atcoder Grand Contest 033 D Complexity

Complexity

- 常规的dp状态数就达到了$\frac{H(H-1)W(W-1)}{4}$，单次转移需要$O(H+W)$，无论是时间复杂度还是空间复杂度都不能接受。

Complexity

- 常规的dp状态数就达到了$\frac{H(H-1)W(W-1)}{4}$，单次转移需要$O(H+W)$，无论是时间复杂度还是空间复杂度都不能接受。
- 可以发现经过$O(\log H+\log W)$次划分后，原网格图可以被划分成一些1×1的网格，所以原网格图的complexity是$O(\log H+\log W)$级别的。

Complexity

- 常规的dp状态数就达到了$\frac{H(H-1)W(W-1)}{4}$，单次转移需要$O(H+W)$，无论是时间复杂度还是空间复杂度都不能接受。
- 可以发现经过$O(\log H+\log W)$次划分后，原网格图可以被划分成一些1×1的网格，所以原网格图的complexity是$O(\log H+\log W)$级别的。
- 试着计算如下两个值:
  - $f_{i,r,c,r'}$:左上角和左下角分别在(r,c)和(r',c)的complexity为i网格图的横向最大长度
  - $g_{i,r,c,c'}$:左上角和右上角分别在(r,c)和(r,c')的complexity为i的网格图的纵向最大长度

Complexity

- 常规的dp状态数就达到了$\frac{H(H-1)W(W-1)}{4}$，单次转移需要$O(H+W)$，无论是时间复杂度还是空间复杂度都不能接受。
- 可以发现经过$O(\log H+\log W)$次划分后，原网格图可以被划分成一些1×1的网格，所以原网格图的complexity是$O(\log H+\log W)$级别的。
- 试着计算如下两个值:
  - $f_{i,r,c,r'}$:左上角和左下角分别在(r,c)和(r',c)的complexity为i网格图的横向最大长度
  - $g_{i,r,c,c'}$:左上角和右上角分别在(r,c)和(r,c')的complexity为i的网格图的纵向最大长度
- 这样的总状态数是$O(HW(H+W)\log HW)$，经过一些预处理后可以达到$O(1)$转移，故总时间复杂度和空间复杂度都是$O(HW(H+W)\log HW)$。

Popcorn

- 为了准备观看TI9,你点了N种爆米花，对于第$i(1 \leq i \leq N)$种爆米花，你知道它的三个值：

  A_i：第i种爆米花成功爆出的时间

  B_i：第i种爆米花烧焦的时间

  C_i：第i种爆米花的数量

- 同时你还有M个容量无限大的装爆米花的袋子，你希望把爆米花划分到M个袋子里，并为第$i(1 \leq i \leq M)$个袋子选择一个时间$prep_i$,使得最后可食用的爆米花数量最多。其中放在第j个袋子里的第i种爆米花可食用，当且仅当$A_i \leq prep_j < B_i$。

Popcorn

- 为了准备观看TI9,你点了N种爆米花，对于第$i(1 \leq i \leq N)$种爆米花，你知道它的三个值：

  A_i：第i种爆米花成功爆出的时间

  B_i：第i种爆米花烧焦的时间

  C_i：第i种爆米花的数量
- 同时你还有M个容量无限大的装爆米花的袋子，你希望把爆米花划分到M个袋子里，并为第$i(1 \leq i \leq M)$个袋子选择一个时间$prep_i$,使得最后可食用的爆米花数量最多。其中放在第j个袋子里的第i种爆米花可食用，当且仅当$A_i \leq prep_j < B_i$。
- Constraints: $1 \leq M \leq N \leq 2 \times 10^5, 1 \leq A_i \leq B_i \leq 2 \times 10^5, \sum C_i \leq 10^9$
- Time Limit: 2s, Memory Limit: 256MB
- Source: CSAcademy Romanian IOI 2017 Selection #2 Popcorn

Popcorn

- 显然同种爆米花一定要装进相同的袋子里。

Popcorn

- 显然同种爆米花一定要装进相同的袋子里。
- 令$dp_{i,j}$表示装了i个袋子，考虑右端点小于等于j的区间时的最大答案，则转移方程为$dp_{i,j} = \max_x dp_{i-1,x-1} + C(x,j)$,其中$C(x,j)$表示满足$A_i \leq x < B_i \leq j$的爆米花的总数量,这样的复杂度是$O(NM^2)$的。

Popcorn

- 显然同种爆米花一定要装进相同的袋子里。
- 令$dp_{i,j}$表示装了i个袋子，考虑右端点小于等于j的区间时的最大答案，则转移方程为$dp_{i,j} = \max_x dp_{i-1,x-1} + C(x,j)$,其中$C(x,j)$表示满足$A_i \le x < B_i \le j$的爆米花的总数量,这样的复杂度是$O(NM^2)$的。
- 优化1：考虑优化$dp_{i,*}$的计算过程，我们对于所有x维护$Val_x = dp_{i-1,x-1} + C(x,j)$.当从$j-1$到$j$的时候，更新$Val_j = dp_{i-1,j-1}$,且对于所有$B_i = j$的$i$，令$Val_{A_i}, Val_{A_i+1}, \ldots, Val_j$加上$C_i$,$dp_{i,j}$的值就是$\max_x Val_x$.
- 显然可以用线段树进行维护，复杂度变为$O(NM \log M)$.

Popcorn

- 优化2: 注意到任何时候如果对于$x < y$有$Val_x \leq Val_y$,x永远不会成为一个最优解，因此我们可以维护一个Val_x的单调下降的序列。我们要做的是: 在序列末端添加一个值，或者给一个后缀加上一个值。

Popcorn

- 优化2: 注意到任何时候如果对于$x < y$有$Val_x \leq Val_y$,x永远不会成为一个最优解，因此我们可以维护一个Val_x的单调下降的序列。我们要做的是: 在序列末端添加一个值，或者给一个后缀加上一个值。
- 在序列末端添加值显然很容易，为了能够快速给后缀加上值，我们维护差分序列，这样就变成了单点修改。任意时候差分序列中如果某个值大于等于0，我们就合并相邻两个数，将它们变成他们的和，这一段可以使用并查集维护。

Popcorn

- 优化2: 注意到任何时候如果对于$x < y$有$Val_x \leq Val_y$,x永远不会成为一个最优解，因此我们可以维护一个Val_x的单调下降的序列。我们要做的是: 在序列末端添加一个值，或者给一个后缀加上一个值。
- 在序列末端添加值显然很容易，为了能够快速给后缀加上值，我们维护差分序列，这样就变成了单点修改。任意时候差分序列中如果某个值大于等于0，我们就合并相邻两个数，将它们变成他们的和，这一段可以使用并查集维护。
- 时间复杂度变为$O(NM)$.

Popcorn

- 优化3: 观察(或猜测)到原题具有性质$ans_k - ans_{k-1} \geq ans_{k+1} - ans_k$,其中$ans_k$表示使用$k$个袋子的答案，所以可以使用wqs二分进行优化

Popcorn

- 优化3: 观察(或猜测)到原题具有性质$ans_k - ans_{k-1} \geq ans_{k+1} - ans_k$,其中$ans_k$表示使用$k$个袋子的答案，所以可以使用wqs二分进行优化
- 时间复杂度为$O(N(\log M + \log 10^9))$.