Slope Trick：一种特殊的动态规划优化方式

张瑞珉

南京大学

2024.12.12

内容提纲

内容目录

动态规划是什么

- 动态规划算法是一种通过把原问题分解为相对简单的子问题，然后自底向上逐一求解的方式求解复杂问题的方法 [1]。**子问题也能被分解为更简单的子问题。**

动态规划是什么

- 动态规划算法是一种通过把原问题分解为相对简单的子问题，然后自底向上逐一求解的方式求解复杂问题的方法 [1]。**子问题也能被分解为更简单的子问题。**
- 具体的说，我们会用“状态”来描述一个子问题。一个状态可能包含很多元素。

状态的形式化定义

一个问题的状态可能具有多个元素。记 $state_{problem} = \{arg_1, arg_2, \ldots, arg_k\}$ 表示问题 $problem$ 的状态为 $state_{problem}$，其 k 个元素分别为 $arg_1, arg_2, \ldots, arg_k$。

动态规划是什么

- 在使用动态规划算法时，除了一些最基础的子问题可以直接求出答案以外，其余问题均需要由它的子问题推导得出。推导的过程类似于一个函数，自变量为它的子问题的解，经过函数的运算后我们能得到这个问题的解。我们称这个函数为“转移方程”。

转移方程的形式化定义

对于每一个问题，它的转移方程都可能是独特的。我们记 $sol_{problem}$ 为问题 $problem$ 的解，那么 $sol_{problem} = func_{problem}(sol_{subproblem_1}, sol_{subproblem_2}, \ldots, sol_{subproblem_m})$，其中 $func_{problem}$ 表示问题 $problem$ 的转移方程，且其有 m 个子问题。

动态规划是什么

举例

一个 n 行 m 列的网格，你初始在 $(1,1)$ 要走到 (n,m)，每次可以向下或者向右走一个单位，求方案数。

动态规划是什么

解法

定义子问题：令 $state_{problem} = \{x, y\}$，即子问题为走到某个点 (x, y) 的方案数。
最基础的子问题的解：$sol_{1,1} = 1$，代表从 $(1, 1)$ 走到 $(1, 1)$ 只有 1 种方案。当 $x = 1$ 或者 $y = 1$ 时，也只有 1 种方案。
设计转移方程：对于一个点 (x, y)，我们只有两种方式可以到达它：由 $(x - 1, y)$ 或者 $(x, y - 1)$ 到达它。
所以转移方程为 $sol(x, y) = sol(x - 1, y) + sol(x, y - 1)$ （如果 $x \geq 2$ 且 $y \geq 2$）。

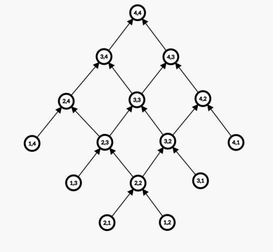

图: 如何转移

内容目录

一类特殊的动态规划问题

- 有一类问题，它的一个特征是在使用动态规划算法分析后，得出其子问题的状态需要两个元素表示，且其中一个元素的定义类似于为“时间”，另一个元素我们称之为 x，类似于坐标。

- 有一类问题，它的一个特征是在使用动态规划算法分析后，得出其子问题的状态需要两个元素表示，且其中一个元素的定义类似于为“时间”，另一个元素我们称之为 x，类似于坐标。
- “时间轴”可以理解为将这个动态规划的子问题按照时间进行分层，转移只能由 $time$ 层的子问题转移至 $time+1$ 层的子问题。

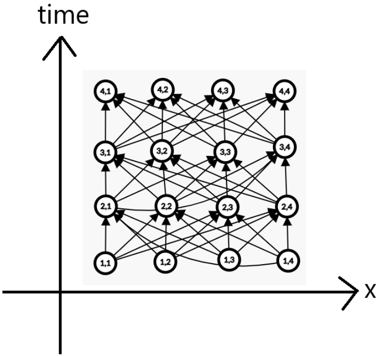

图: 如何转移

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
- 1. 转移方程相当于对于所有的 `dp[time][x]` 都加上一个 $|x-a|$，其中 a 对于一个固定的时刻是一个常数。

1. For each x from $-\infty$ to $+\infty$:
   1. Update $dp[time+1][x] \leftarrow dp[time][x] + |x-a|$

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
  1. 转移方程相当于对于所有的 dp[time][x] 都加上一个 $|x-a|$，其中 a 对于一个固定的时刻是一个常数。

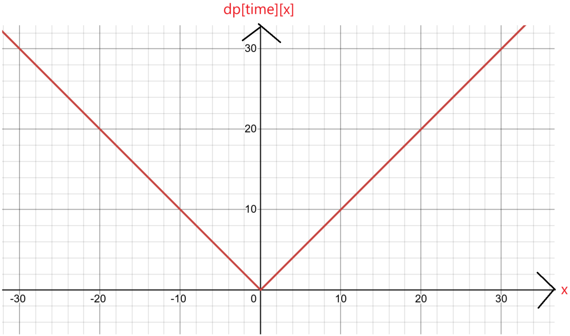

图: $|x|$

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
  1. 转移方程相当于对于所有的 `dp[time][x]` 都加上一个 $|x-a|$，其中 a 对于一个固定的时刻是一个常数。

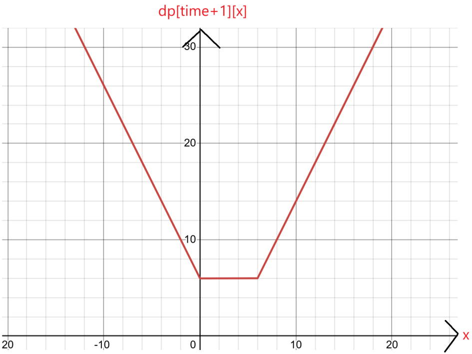

图: |x|+|x-6|

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
  2. 转移方程相当于将整个图像沿 x 轴平移后取中间所有图像在某个 x 处的较小值。

---

1. For each x from $-\infty$ to $+\infty$:
   1. Update:

$$dp[time+1][x] \leftarrow \min\big(dp[time][x-a],\\ dp[time][x-a+1], \ldots, dp[time][x]\big)$$

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
  2. 转移方程相当于将整个图像沿 x 轴平移后取中间所有图像在某个 x 处的较小值。

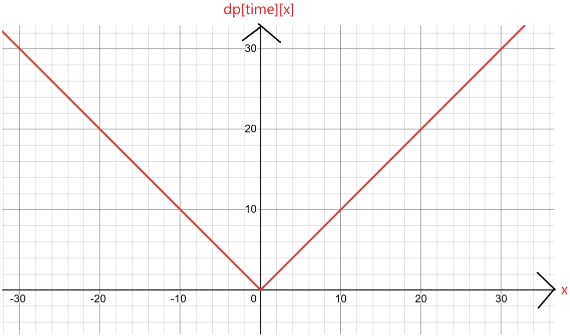

图: |x|

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
  2. 转移方程相当于将整个图像沿 x 轴平移后取中间所有图像在某个 x 处的较小值。

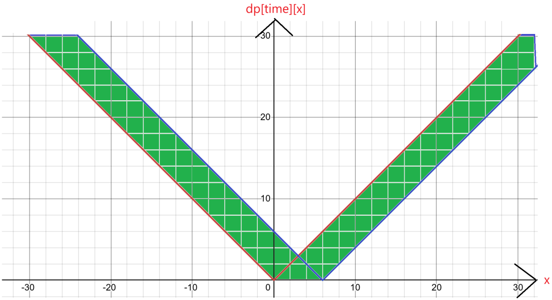

图: |x| 向右平移 6 个单位

一类特殊的动态规划问题

- 它的另一个关键性质是，在由当前时刻的 dp 数组转移至下一个时刻的 dp 数组时，转移方程有以下两种方式：
  2. 转移方程相当于将整个图像沿 x 轴平移后取中间所有图像在某个 x 处的较小值。

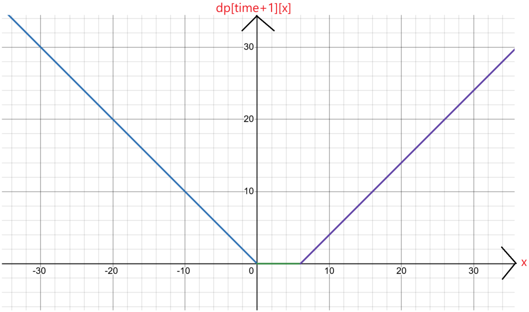

图: 取 min

一类特殊的动态规划问题

- 传统的动态规划做法需要按照伪代码更新全部的随着 time 推移的 dp 数组的值。

一类特殊的动态规划问题

- 传统的动态规划做法需要按照伪代码更新全部的随着 time 推移的 dp 数组的值。
- 本文所介绍的 `Slope Trick` [2] 可以更高效的解决这一类特殊问题。

内容目录

核心思想

- 我们可以发现，对于这两种转移有一个很好的性质：无论怎么变化，我们所得到的一定是一个分段函数，且每段的斜率不降。我们称之为一个“凸包”。

核心思想

- 我们可以发现，对于这两种转移有一个很好的性质：无论怎么变化，我们所得到的一定是一个分段函数，且每段的斜率不降。我们称之为一个“凸包”。
- 我们记斜率突变的位置为“转折点”。通过记录转折点 x 坐标，就可以很方便的推出 dp 数组在所有的 x 处的值，而不用记录整个数组的值。

核心思想

- 对于一个固定的 $time$，我们按照伪代码模拟推导的过程为 $dp[time+1] = f(dp[time])$。
  然而，$SlopeTrick$ 的核心思想是通过转化从而快速求解 dp 数组的所有值。下文的 $dp[time]$ 代表在 $time$ 时刻，对于所有的 x，$(x, dp[time][x])$ 所形成的折线。
  首先找到一个可逆函数 g，使得 $g(dp[time])$ 能得到一个新的数据结构 $p[time]$，其存储了 $dp[time]$ 的所有转折点的位置。
  然后我们找到一个函数 h，使得 $p[time+1] = h(p[time])$。
  最终，我们由 $dp[time+1] = g^{-1}(p[time+1])$ 推出 $dp[time+1]$。

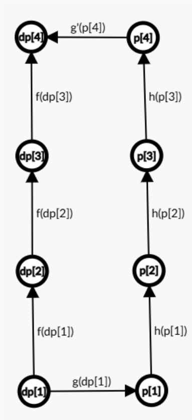

图: 转移过程

内容目录

具体实现

我们考虑用一种数据结构维护 $p[time]$。

- 第一种转移方程：$dp[time+1][x] = dp[time][x] + |x - a|$
  可以发现实质上 $p[time+1]$ 相较于 $p[time]$ 只多了一个在 a 处的转折点。我们只需要在 $p[time]$ 的基础上插入一个 a 即可。

具体实现

我们考虑用一种数据结构维护 $p[time]$。

- 第一种转移方程：$dp[time+1][x] = dp[time][x] + |x-a|$
  可以发现实质上 $p[time+1]$ 相较于 $p[time]$ 只多了一个在 a 处的转折点。我们只需要在 $p[time]$ 的基础上插入一个 a 即可。
- 第二种转移方程：
  $dp[time+1][x] = min\{dp[time][x-a], dp[time][x-a+1], \ldots, dp[time][x]\}$
  如果 $a>0$，那么相当于对于所有对应斜率大于等于 0 的转折点，他们的 x 坐标集体加上了 a。
  如果 $a<0$ 同理，对于所有对应斜率小于 0 的转折点，他们的 x 坐标集体减去了 a。

具体实现

我们考虑用一种数据结构维护 $p[time]$。

- 第一种转移方程：$dp[time+1][x] = dp[time][x] + |x-a|$
  可以发现实质上 $p[time+1]$ 相较于 $p[time]$ 只多了一个在 a 处的转折点。我们只需要在 $p[time]$ 的基础上插入一个 a 即可。
- 第二种转移方程：
  $dp[time+1][x] = min\{dp[time][x-a], dp[time][x-a+1], \ldots, dp[time][x]\}$
  如果 $a>0$，那么相当于对于所有对应斜率大于等于 0 的转折点，他们的 x 坐标集体加上了 a。
  如果 $a<0$ 同理，对于所有对应斜率小于 0 的转折点，他们的 x 坐标集体减去了 a。
- 考虑到这两种操作方式，我们通常使用堆来作为存储 p 数组的数据结构。

内容目录

复杂度分析

- 传统动态规划模拟所需要的时间复杂度为 $O(nm)$，其中 n 是 $time$ 的最大值，m 是可以取到的 x 的数量。

复杂度分析

- 传统动态规划模拟所需要的时间复杂度为 $O(nm)$，其中 n 是 $time$ 的最大值，m 是可以取到的 x 的数量。
- 然而，使用 `Slope Trick` 时，插入一个元素的时间复杂度为 $O(\log n)$，第二种操作只需要给对应的部分打上标记而不用实际去修改值，所以它的总时间复杂度为 $O(n \log n)$。

内容目录

总结

- `Slope Trick` 是一种不常见的技巧，然而它在解决这一类特殊的动态规划问题时能够起到奇效。它的代码量很短，运行效率也很高，无法被一般动态规划做法所取代。

总结

- `Slope Trick` 是一种不常见的技巧，然而它在解决这一类特殊的动态规划问题时能够起到奇效。它的代码量很短，运行效率也很高，无法被一般动态规划做法所取代。
- 同时，它也可以融合闵可夫斯基求和等算法，解决更多的动态规划问题。

内容目录

refs

[1] CBW2007 等. 动态规划部分简介. https://oi-wiki.org/dp/, 2021.
[2] Doan Duc Trung Dang. Slope trick explained. https://codeforces.com/blog/entry/77298, 2019.

内容目录

致谢

感谢梁红瑾老师和林荣恩学长的悉心指导，没有他们我无法将这份论文呈现出来。
感谢各位参与我的答辩。
谢谢大家！