动态规划

——科普向（大雾

孙玥

概念

- 动态规划(dynamic programming)
- 动态规划算法通常用于求解具有某种最优性质的问题。
- 在这类问题中，可能会有许多可行解。

数字三角形(POJ 1163)

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

从最上面的顶点开始，每次可以移动到下一层左右两个点上，求路径上走过的点上的值加和最大值。

数字三角形(POJ 1163)

7
3 8
8 1 0
2 7 4 4
4 5 2 6 5

贪心？

搜索？

记录状态

数字三角形(POJ 1163)

- 状态转移方程
- dp[i][j] =

  num[i][j] + max(dp[i - 1][j - 1], dp[i - 1][j])

  7

  3 8

  8 1 0

  2 7 4 4

  4 5 2 6 5

dp[i][j]表示走到第i行第j列元素时获得的最大价值

最大子段和

- 给出一段数字序列，求其中一段连续的子序列之和最大
- a[] = {-2,11,-4,13,-5,-2}
- dp[i]表示以第i个元素结尾的最大子段和
- dp[i] = max(dp[i - 1], 0) + a[i]
- dp[0] = -inf

最大子段和

- 最大两段子段和：POJ(2379)

最长上升子序列(POJ 2533)

- 给出一个数字序列，求最长上升子序列。
- a[] = {1,7,3,5,9,4,8}
- dp[i]表示以第i个元素结尾的最长上升子序列长度
- dp[i] = max{dp[j] + 1 | 1 <= j < i, a[i] > a[j]}

最长公共子序列(POJ 1458)

- 求给出的两个序列的最长公共子序列
- x = “ABCBDAB”
- y = “BDCABA”
- dp[i][j]表示x的前i个字符与y的前j个字符的最长公共子序列长度
- dp[i][j] = {dp[i - 1][j - 1] + 1, x[i] == y[i]
- max(dp[i - 1][j], dp[i][j - 1]), x[i] != y[i]}

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

01背包(POJ 3624)

- 有n件物品，一个容量为c的背包，每个物品的体积和价值分别为w_i和v_i，将物品装入背包求获得的最大价值
- n = 4, c = 6, w = {1, 2, 3, 2}, v = {4, 6, 12, 7}
- dp[i][j]表示用前i件物品装满容量为j的背包获得的最大价值。
- dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - w[i]] + v[i])
- dp[i][j] = dp[i - 1][j]

01背包(POJ 3624)

- 背包九讲

dp问题类型

- 区间dp
- 状态压缩dp
- 树形dp
- 数位dp
- ……
- 矩阵快速幂优化递推公式