动态规划(下)

九章算法强化班 第6章

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
知乎: http://zhuanlan.zhihu.com/jiuzhang
官网: http://www.jiuzhang.com

Overview

I. 区间类DP
   I. Stone Game
   II. Burst Ballons
   III. Scramble String
II. 背包类DP
   I. BackPackI
   II. BackPackII
   III. K SUM
   IV. **Minimum Adjustment Cost**

区间类Dp

特点：

1. 求一段区间的解max/min/count
2. 转移方程通过区间更新
3. 从大到小的更新

Stone Game

http://www.lintcode.com/en/problem/stone-game/
http://www.jiuzhang.com/solutions/stone-game/

[3,4,5,6]

贪心反例：

[1,1,1,1]

Stone-Game

- 死胡同：容易想到的一个思路从小往大，枚举第一次合并是在哪？
- 记忆化搜索的思路，从大到小，先考虑最后的0-n-1 合并的总花费
- State:
  - dp[i][j] 表示把第i到第j个石子合并到一起的最小花费
- Function:
  - 预处理sum[i,j] 表示i到j所有石子价值和
  - dp[i][j] = min(dp[i][k]+dp[k+1][j]+sum[i,j]) 对于所有k属于{i,j}
- Intialize:
  - for each i
    - dp[i][i] = 0
- Answer:
  - dp[0][n-1]

Burst Ballons

http://www.lintcode.com/en/problem/burst-balloons/

http://www.jiuzhang.com/solutions/burst-ballons/

Burst Ballons

- 死胡同：容易想到的一个思路从小往大，枚举第一次在哪吹爆气球？
- 记忆化搜索的思路，从大到小，先考虑最后的0-n-1 合并的总价值

- State:
  - dp[i][j] 表示把第i到第j个气球打爆的最大价值
- Function:
  - 对于所有k属于{i,j}, 表示第k号气球最后打爆。
    - midValue = arr[i- 1] * arr[k] * arr[j+ 1];
    - dp[i][j] = max(dp[i][k-1]+dp[k+1][j]+midvalue)
- Intialize:
  - for each i
    - dp[i][i] = 0
- Answer:
  - dp[0][n-1]

Scramble String

http://www.lintcode.com/en/problem/scramble-string/
http://www.jiuzhang.com/solutions/scramble-string/
[abcd, dcab]

Scramble String

- 看 f[great][rgreat] 这个参考例子
- f[gr|eat][rgreat] =
  - f[gr][rg] && f[eat][eat]
  - f[gr][at] && f[eat][rgr]

Scramble String

- State:
  - dp[x][y][k] 表示是从s1串x开始，s2串y开始，他们后面k个字符组成的substr是Scramble String
- Function:
  - 对于所有i属于{1,k}
  - s11 = s1.substring(0, i); s12 = s1.substring(i, s1.length());
  - s21 = s2.substring(0, i); s22 = s2.substring(i, s2.length());
  - s23 = s2.substring(0, s2.length() - i); s24 = s2.substring(s2.length() - i, s2.length());
  - for i = x -> x+k
    - dp[x][y][k] = (dp[x][y][i] && dp[x+i][y+i][k-i]) || dp[x][y+k-i][i] && dp[x+i][y][k-i])
- Intialize:
  - dp[i][j][1] = s1[i]==s[j].
- Answer:
  - dp[0][0][len]

区间DP

- coin in a line III
- stone game
- scramble string

- 这种题目共性就是区间最后求[0,n-1] 这样一个区间
- 逆向思维分析 从大到小就能迎刃而解

- 逆向=》分治类似

背包类Dp

特点：

1. 用值作为DP维度
2. Dp过程就是填写矩阵
3. 可以滚动数组优化

BackPack

http://www.lintcode.com/en/problem/backpack/
http://www.jiuzhang.com/solutions/backpack/

Backpack

- State:
  - f[i][S] “前i”个物品，取出一些能否组成和为S
- Function:
  - f[i][S] = f[i-1][S - a[i]] or f[i-1][S]
- Intialize:
  - f[i][0] = true; f[0][1..target] = false
- Answer:
  - 检查所有的f[n][j]
- O(n*S) ， 滚动数组优化

BackPack 马甲题型 变1

硬币凑整

给1，2，5，10硬币无数多个，请问凑80元的方案总数

http://www.lintcode.com/en/problem/backpack-iv/

http://www.jiuzhang.com/solutions/backpack-iv/

Backpack

- State:
  - f[i][S] “前i”个物品，取出一些能否组成和为S
- Function:
  - f[i][S] = f[i-1][S - a[i]] or f[i-1][S]
- Intialize:
  - f[i][0] = true; f[0][1..target] = false
- Answer:
  - 检查所有的f[n][j]
- O(n*S) ， 滚动数组优化

BackPack 马甲题型 变2

把一个[1,24,5,6]数组尽量平分。

Backpack II

http://www.lintcode.com/en/problem/backpack-ii/
http://www.jiuzhang.com/solutions/backpack-ii/

贪心反例:
背包容量 = 9
A=[4,5,7]
V=[3,4,6]

Backpack II

- 状态 State
  - f[i][j] 表示前i个物品当中选一些物品组成容量为j的最大价值
- 方程 Function
  - f[i][j] = max(f[i-1][j], f[i-1][j-A[i-1]] + V[i-1]);
- 初始化 Intialization
  - f[0][0]=0;
- 答案 Answer
  - f[n][s]
- O(n*s)

K Sum

http://www.lintcode.com/en/problem/k-sum/
http://www.jiuzhang.com/solutions/k-sum/

K Sum

- n个数，取k个数，组成和为target
- State:
  - f[i][j][t]前i个数取j个数出来能否和为t
- Function:
  - f[i][j][t] = f[i - 1][j - 1][t - a[i-1]] + f[i - 1][j][t]
- Intialization
  - f[i][0][0] = 1
- Answer
  - f[n][k][target]

Minimum Adjustment Cost

http://www.lintcode.com/en/problem/minimum-adjustment-cost/

最小调整代价

- State:
  - f[i][v] 前i个数，第i个数调整为v，满足相邻两数<=target，所需要的最小代价
- Function:
  - f[i][v] = min(f[i-1][v'] + |A[i]-v|, |v-v'| <= target)
- Answer:
  - f[n][a[n]-target~a[n]+target]

- O(n * A * T)

Summary

- 区间类DP问题
  - 从大到小去思考
  - 主要是通过记忆化来直观理解DP的思路

- 背包DP问题
  - 用值作为DP维度
  - Dp过程就是填写矩阵
  - 可以滚动数组优化

今日重点题型

- Backpack II
  - 有价值的背包题目才有价值
- Stone-Game
  - 区间类DP的入门题

Thank You