4月25日-5月1日训练报告

1 AT dp题单E

2022-04-25 20:32:32	E - Knapsack 2	vjudge0	C++ (GCC 9.2.1)	100	745 Byte	AC	22 ms	4364 KB	詳細

总结

思路： 01背包变式。

这题的特点是背包容量特别大，然后物品的价值比较小。所以按背包容量进行转移是不可能的。但是这题的物品价值比较小，我们可以按物品价值进行转移，数组中存放占用的背包容量，然后遍历dp数组找到符合题目容量要求的最大价值即可。

2 AT dp题单I、J

2022-04-27 21:19:21	J - Sushi	vjudge0	C++ (GCC 9.2.1)	100	865 Byte	AC	229 ms	112500 KB	詳細
2022-04-27 21:18:39	J - Sushi	vjudge0	C++ (GCC 9.2.1)	0	848 Byte	TLE	2205 ms	3652 KB	詳細
2022-04-27 20:32:23	I - Coins	vjudge0	C++ (GCC 9.2.1)	100	650 Byte	AC	70 ms	85644 KB	詳細

总结

思路：

概率dp，转移概率或者期望。

在J题中，盘子的下标对答案没有影响，我们只需要知道盘子中有k个寿司的盘子有几个即可。因此，我们令dp[i][j][k]为当1个寿司的盘子有i个，2个寿司的盘子有j个，3个寿司的盘子有k个时所需要的操作次数的期望。我们很容易得出，在这次操作中，清理没有寿司的盘子的概率为(n - i - j - k) / n,清理1个寿司的盘子的概率为i / n,清理2个寿司的盘子的概率为j / n,清理3个寿司的盘子的概率为k / n。因此，我们得出转移方程为：

dp[i][j][k] = dp[i][j][k] * (n - i - j - k) / n + dp[i - 1][j][k] * i / n + dp[i + 1][j - 1][k] * j / n + dp[i][j + 1][k - 1] * k / n + 1;

这里加1的意思是这次操作会使总操作次数的期望+1。

整理方程，可得转移方程为：

dp[i][j][k] = n / (i + j + k) + dp[i - 1][j][k] * i / (i + j + k) + dp[i + 1][j - 1][k] * j / (i + j + k) + dp[i][j + 1][k - 1] * k / (i + j + k);

到状态转移分析完毕。由于转移过程分析比较复杂，这里我使用记忆化dfs来实现。

3 AT dp题单K

2022-04-29 21:56:37	K - Stones	vjudge0	C++ (GCC 9.2.1)	100	1021 Byte	AC	63 ms	12924 KB	詳細

总结

思路：

博弈型dp，转移过程跟一般的线性dp一样，但是需要注意的是必须存在上一次操作是必败状态，这次操作才能是必胜状态，而不是上一次操作可能失败，这次操作就能胜利。