区间模型

主讲人：邓哲也

区间模型

区间DP，故名思义就是在区间上 DP。

先算出小区间的 DP 得到最优解。

再去得到大区间的最优解。

一般的状态是设 f[i][j] 为区间 [i, j] 的最优解。

一般 f[i][j] 都可以由 [i, j] 的子区间的最优解更新得到。

合并果子

有 n 堆果子，第i堆有 a[i] 个。

合并的时候只能合并相邻两堆，产生的代价为两堆果子的总数。

经过 n-1 轮合并后变成了一堆，求总代价的最小值。

样例： （输出 9）

3

1 2 3

合并果子

设计状态：

用 f[i][j] 来表示合并 [i, j] 这个区间内的果子产生的最小代价。

思考转移：

因为合并之后 [i, j] 就成了一堆，因此在合并之前一定是两堆。

我们可以枚举分界线，也就是枚举 i≤k<j，假设合并前的两堆分别是 [i, k], [k + 1,j]

合并果子

这两堆本身就有 f[i][k] + f[k + 1][j] 的代价。

而合并这两堆的代价与 k 没关系，因为就是 a[i] + a[i + 1] + ...

+ a[j]

我们可以设一个 a 的前缀和 sum

这样 f[i][j] = min(f[i][j], f[i][k] + f[k + 1][j] + sum[j] – sum[i – 1])

合并果子

实现方法 1： 记忆化 DP

```
int dp(int i, int j){
	if (i == j) return 0;
	if (f[i][j] != -1) return f[i][j];
	int ans = 1e9;
	for (int k = i;k < j;k ++)
		ans = min(ans, dp(i, k) + dp(k + 1, j) + sum[j] – sum[i – 1]);
	return f[i][j] = ans;
}
```

合并果子

实现方法 2： 直接 dp （要先算小区间的答案）

```
for(int len = 2;len <= n;len ++){
        for(int i =1;i + len – 1<= n;i ++){
                int j = i + len – 1;
                for(int k = i;k < j;k ++)
                        f[i][j] = min(f[i][j], f[i][k] + f[k + 1][j] + sum[j] – sum[i –
1]);
        }
}
```

括号匹配

给出一个只有 () [] 四种字符组成的字符串，取出一个最长的子序列使得他们满足括号匹配。

样例：

([]) （答案：4）

([][][) （答案：6）

括号匹配

设计状态:

f[i][j] 表示区间 [i,j] 中的最长匹配子序列。

思考转移:

如果 S[i] 和 S[j] 可以匹配，那么 f[i][j] = f[i + 1][j – 1] + 2;

另外 [i, j] 的答案也可以由两个子区间的答案合并得来。

f[i][j] = max(f[i][j], f[i][k] + f[k + 1][j])

括号匹配

```
for(int len = 2;len <= n;len ++)
    for(int i = 1;i + len – 1 <= n;i ++){
        int j = i + len – 1;
        if (s[i] == ‘(‘ && s[j] == ‘)’ || s[i] == ‘[‘ && s[j] == ‘]’)
            f[i][j] = f[i + 1][j – 1] + 2;
        for(int k = i;k < j;k ++)
            f[i][j] = max(f[i][j], f[i][k] + f[k + 1][j]);
    }
```

下节课再见