第四天

笔记信息	
作者	Gingmzmzx
时间	2023-10-2
教师	周天宝

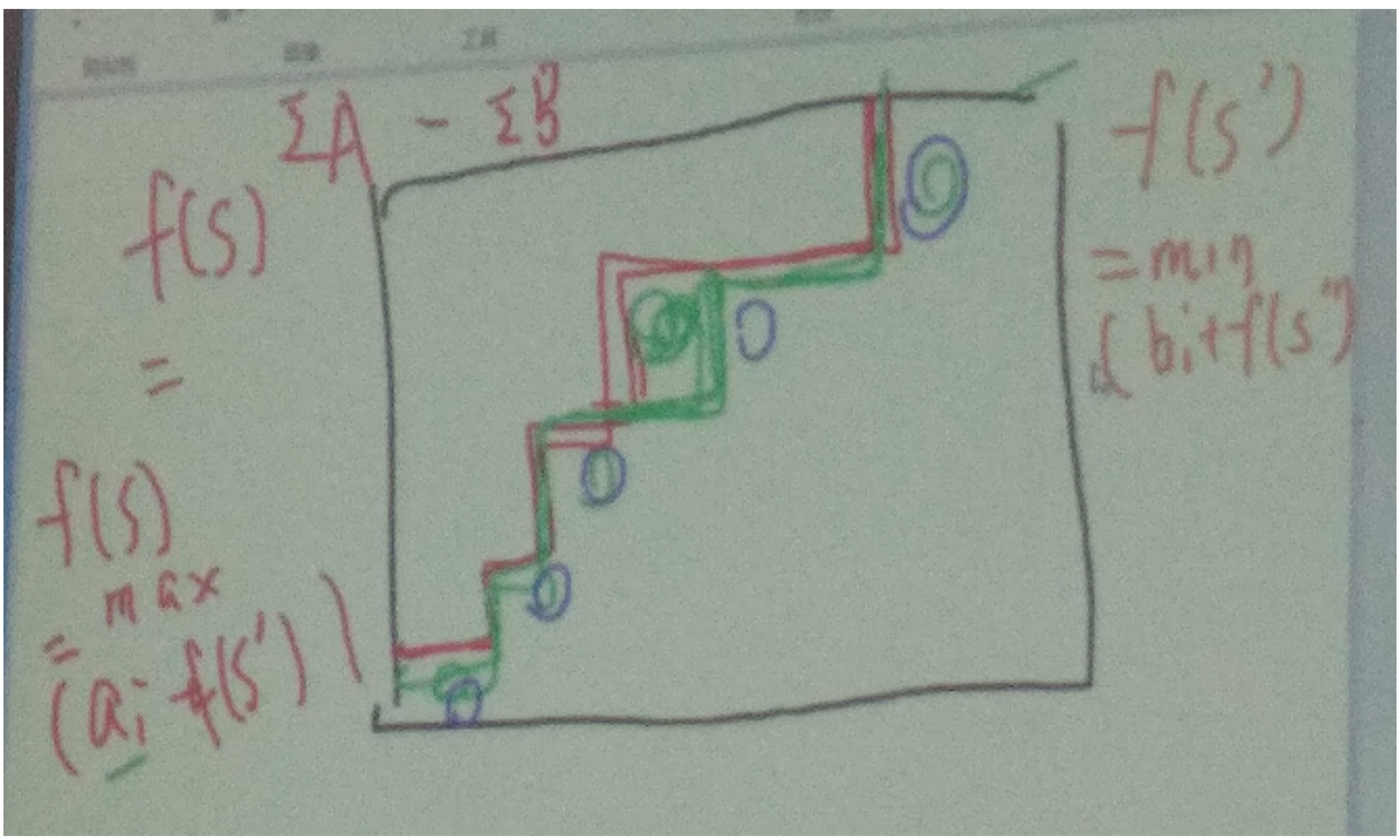
日常膜拜

上午

一、状态压缩

Luogu P4363 一双木棋

- 小提示：状态压缩压边界线就可以
- 图解：



Luogu P5369 最大前缀和

- 统计每个子集作为最大前缀和的次数。
- 如果某个前缀和是最大前缀和，那么这个前缀的所有后缀和都 >0 ，后面的所有前缀和都 ≤ 0
- 设 $f[S]$ 为将 S 排列成所有后缀 >0 的方案数。
- 设 $g[S]$ 为将 S 排列成所有前缀 ≤ 0 的方案数。
- 答案即为 $\sum sum[S]f[S]g[U-S]$ ， U 为全集。

Luogu P5492 随机算法

- 设 $f[S]$ 为 S 内随机一个排列的正确方案数。
- 枚举 S 排列的第一个元素 x ，加入 x 会导致删除 x 及与其相邻的点集 c_x 。
- 那么如果 $\max[S]=\max[S-c_x]+1$ ，就令 $f[S] += f[S-c_x]$ 。

拓扑序计数

- 题目描述:

  > 给定一张有向无环图，求其合法拓扑序个数
  > $n\leqslant 20, m\leqslant \frac{n(n-1)}{2}$

- 题解:
  - 设 $f(S)$ 表示 S 诱导子图的拓扑序个数。
  - 转移时，枚举 S 中拓扑序最靠前的节点 i ： $f(S)=\sum_{i\in S,\mathrm{ind}_i=0} f(S-\{i\})$ 。其中 $\mathrm{ind}_i=0$ 表示 i 在 S 中没有入度（而非整个图中）。

边子集拓扑序计数

- 题目描述:

  > - 给定一张有向图（不一定是DAG）。设其边集为E
  > - 对于$T\subseteq E$定义$f(T)$为保留T时的合法拓扑序个数
  > - 求$\sum_{t\subseteq E} f(T)$
  > - $n\leqslant 20, m\leqslant n(n-1)$

- 题解:
  - 设 $g(S)$ 表示 S 诱导子图的答案。我们输出 $g(U)$ 。
  - 与其统计每个边集 T 的拓扑序方案数，不如统计每种拓扑序能在多少种 T 下合法。
  - 转移时仍然枚举 S 中拓扑序最靠前的节点 i 。
  - 要想让 i 最靠前合法，就需要让 i 在 S 内的 $ind=0$。换句话说，不能保留从 $S-\{i\}$ 连向 i 的边，而 i 连向 $S-\{i\}$ 的边可以任意保留。
  - 所以 $g(S)=\sum_{i\in S} f(S-\{i\})2^{cnt_{i,S-\{i\}}}$ 。其中 $cnt_{i,S-\{i\}}$ 表示 i 连向 $S-\{i\}$ 的边数。

Luogu P2831 愤怒的小鸟

- 每个优的抛物线至少经过两只猪。用 $\begin{cases} y_1=ax_1^2+bx_1 \\ y_2=ax_2^2+bx_2 \end{cases}$ 可解出一组 (a,b) 。共有 $O(n^2)$ 组。预处理出每两只猪 (i,j) 形成的抛物线能消灭哪些猪 $c[i][j]$ 。
- 设 $f[S]$ 为消灭 S 集合猪最少需要多少只小鸟。枚举第一条抛物线，则 $f[S]=\min(f\big[S-c[i][j]\big]+1)$ 。
- 固定 i 为 S 中的最小元，则只需枚举 j 即可。
- 复杂度为 $O(n2^n)$

Luogu P3959 宝藏

- 状态中需记录当前层数。
- 设 $f[x][i][S]$ 为 x 的子树编号集合为 S， x 在第 i 层的最小子树代价。
- $f[x][i][S] = \min(f[x][i][T] + f[y][i+1][S-T] + dis(x,y) \times i)$
- 枚举每个点作为起始点。

二、优先队列优化DP

经典例题：滑动窗口

- 题目描述（`GitHub Copilot`）：
  > 给定一个长度为n的序列，求其所有长度为k的连续子序列中的最大值
  > $n \leqslant 10^6, k \leqslant 10^5$

  > 洛谷上的题目：Luogu P1886 滑动窗口，与 `GitHub Copilot` 描述的题目稍有差别

- 题解（`GitHub Copilot`）：
  > - 用一个优先队列维护当前窗口内的最大值
  > - 每次窗口滑动时，将窗口左端点的值出队，右端点的值入队
  > - 时间复杂度$O(n \log k)$

- 实现代码（`GitHub Copilot`）：

```cpp
#include <iostream>
#include <queue>
using namespace std;
const int N = 1e6 + 10;
int n, k, a[N];
priority_queue<pair<int, int>> q;

int main() {
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= k; i++) q.push({a[i], i});
    cout << q.top().first << ' ';
    for (int i = k + 1; i <= n; i++) {
        q.push({a[i], i});
        while (q.top().second <= i - k) q.pop();
        cout << q.top().first << ' ';
    }
    return 0;
}
```

经典例题：中位数

- 题目描述（`GitHub Copilot`）：
  > 给定一个长度为n的序列，求其所有长度为k的连续子序列中的中位数
  > $n \leqslant 10^6, k \leqslant 10^5$

- 题解（`GitHub Copilot`）：
  > - 用两个优先队列维护当前窗口内的最大值和最小值
  > - 每次窗口滑动时，将窗口左端点的值出队，右端点的值入队
  > - 时间复杂度$O(n \log k)$

- 实现代码（`GitHub Copilot`）：

```cpp
#include <iostream>
#include <queue>
using namespace std;
const int N = 1e6 + 10;
int n, k, a[N];
priority_queue<int, vector<int>, greater<int>> q1; // 小根堆
priority_queue<int> q2; // 大根堆

int main() {
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= k; i++) {
        if (q2.empty() || a[i] <= q2.top()) q2.push(a[i]);
        else q1.push(a[i]);
        if (q1.size() > q2.size()) {
            q2.push(q1.top());
            q1.pop();
        }
        if (q2.size() > q1.size() + 1) {
            q1.push(q2.top());
            q2.pop();
        }
    }
    cout << (q2.size() > q1.size() ? q2.top() : (q1.top() + q2.top()) / 2) << endl;
    for (int i = k + 1; i <= n; i++) {
        if (a[i - k] <= q2.top()) q2.pop();
        else q1.pop();
        if (q2.empty() || a[i] <= q2.top()) q2.push(a[i]);
        else q1.push(a[i]);
        if (q1.size() > q2.size()) {
            q2.push(q1.top());
            q1.pop();
        }
        if (q2.size() > q1.size() + 1) {
            q1.push(q2.top());
            q2.pop();
        }
        cout << (q2.size() > q1.size() ? q2.top() : (q1.top() + q2.top()) / 2) << endl;
    }
    return 0;
}
```

简单应用：Luogu P2216 理想的正方形

Luogu P2219 修筑绿化带

- 求出每一块花园中，花坛应该放在哪里。如果花园的右下角为 (x,y)，那么花坛的右下角就应位于 $(x-a+c+1,y-b+d+1)$~$(x-1,y-1)$ 这个矩形内。
- 求出这个矩形内最小的 sum 。

求最大全0正方形

- 题目描述：（Luogu P4259待修）
  > 给定一个$n \times m$的0/1矩阵，求其最大的全0正方形的边长
  > $n, m \leqslant 1000$

- 题解：
  - 带log的做法：枚举每个点作为左上角，二分正方形边长
  - 不带log的做法：
    - 令 $f[i][j]$ 表示矩形 (i,j) 位置往下延伸 0 的个数。
    - $f[i][j]=\begin{cases}0, a_{i,j}=1\\ f[i+1][j], a_{i,j}=0\end{cases}$
    - 枚举正方形的上边界 u，依次从 $1\sim n$ 枚举正方形的右边界。维护一个左边界的指针，并用单调队列维护左右边界之中的 f 值，需要时刻保证 $r-l+1\le \min f$ 。
    - 复杂度为 $O(n^2)$ 。
      - `GitHub Copilot` 这样说：
        这是一个经典的计算机科学问题，可以使用动态规划来解决。我们可以定义一个二维数组dp，其中$dp_{i,j}$表示以(i,j)为右下角的最大全0正方形的边长。那么，我们可以得到以下状态转移方程：
        $$dp_{i,j} = \begin{cases}0 & \text{if } i = 0 \text{ or } j = 0 \text{ or } matrix_{i,j} = 1 & \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1}) + 1 & \text{otherwise}\end{cases}$$
        其中，$matrix_{i,j}$表示矩阵中第i行第j列的元素。
        最终，我们只需要遍历dp数组，找到其中的最大值即可。
        这个方法的时间复杂度是$O(nm)$，因为我们需要遍历整个矩阵来计算dp数组，然后再遍历一次dp数组来找到最大值。

Luogu P2254 瑰丽华尔兹

- 设 $f[t][i][j]$ 为 t 时刻到达 (i,j) 的最长路程。
- 假如 t 时刻钢琴是往上方滑动，那么 $f[t][i][j]=\max(f[t-1][i+1][j]+1, f[t-1][i][j])$ 。往其它方向同理。
- 复杂度为 $O(nmT)$ 。
- 由于时间可以被分为 K 段，每段滑动方向相同，则可以修改状态，令 $f[k][i][j]$ 为第 k 个时间段结束后到达 (i,j) 的最长路程。
- $f[k][i][j]=\max(f[k-1][i+s][j]+s), 0\le s\le t_k$ 。使用单调队列优化。复杂度 $O(nmK)$ 。

Luogu P4381 Island

- 题目大意：
  给一个集环树，求直径
- 题解：
  - 求若干个基环树的直径之和。
  - 对于一个基环树，找到它的环，求出环上每个节点 i 向环外延伸的最长距离 d_i 。
  - 将长为 m 的环破为长为 $2m$ 的链。如果从节点 i 外面走到节点 j 外面，则距离为 $j-i+d_i+d_j$，同时要求 $i<j<i+n$ 。使用单调队列优化求解。
  - 复杂度为 $O(n)$ 。

Luogu P5665 划分

- 贪心地使最后一段的和尽可能小。
- 不严谨证明：如果最后一段 $[l,r]$ 的和没到达下界，那么可以不断地把 a_l 分给前一段。由于 $s_i<s_{i+1}$，分给前一段的贡献是 $2a_ls_i$，分给后一段的贡献是 $2a_ls_{i+1}$，则分给前一段更优。同时分给前一段也有利于后面的分段。
- 所以设 $f[r]$ 为 $\max\{l-1$：最后一段分成$[l,r]$可行$\}$，应有 $sum_r-sum_{l-1}\ge sum_{l-1}-sum_{f[l-1]}$，即 $sum_r\ge 2sum_{l-1}-sum_{f[l-1]}$，单调队列即可。
- 复杂度为 $O(n)$ 。

Luogu P5824 十二重计数法

Luogu P3702 序列计数

- 用总方案数去掉不含质数的方案数。
- 分别用 DP 求解，设 $f[i][j]$ 为 i 个数字，和 $mod\, p = j$ 的方案数，矩阵快速幂加速。
- 复杂度 $O(p^3 \log n)$ 。

Luogu P3773 吉夫特

- 由 Lucas 定理可知 C_n^m是奇数 $\leftrightarrow (n\&m) = m$ 。
- 设 $f[T]$ 表示结尾值为 S 的序列个数。
- 转移时枚举子集 T ，如果 $pos[S] < pos[T]$ 则令 $f[T] += f[S]$ 。
- 复杂度为 $O(3^w)$ 。

Luogu P5664 Emiya 家今天的饭

- 对“每种食材最多在一半菜中”进行容斥。用总的方案数减去某种食材过多的方案数。不会有两种食材同时过多。
- 总方案数易求。
- 枚举过多的食材种类 t ，设 $f[i][j][k]$ 表示前 i 种烹饪方法，一共做了 j 个菜，其中 k 个使用 t 的方案数。转移时枚举第 $i+1$ 种烹饪方法是否选用，选用时是否使用 t 。最后要求 $k > \frac{j}{2}$ 。
- 只需记录 $2k - j$ 的值即可。最后要求此值 > 0 。
- 复杂度为 $O(n^2 m)$ 。

三、高维前缀和

没有记下来哦~

四、斜率优化

Luogu P3195 玩具装箱