# 切树游戏 (cut) 解题报告

Claris

2017 年 3 月 7 日

## 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

## 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

$q$ 次操作，每次要么修改点权，要么查询有多少非空连通子树的点权异或和为 $k$。

# 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

$q$ 次操作，每次要么修改点权，要么查询有多少非空连通子树的点权异或和为 $k$。

- 存在 20% 的数据，$n \le 2000, m = 64$，不存在修改操作。

# 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

$q$ 次操作，每次要么修改点权，要么查询有多少非空连通子树的点权异或和为 $k$。

- 存在 20% 的数据，$n \leq 2000, m = 64$，不存在修改操作。
- 存在 20% 的数据，$n \leq 30000, m = 8$，树退化成链。

## 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

$q$ 次操作，每次要么修改点权，要么查询有多少非空连通子树的点权异或和为 $k$。

- 存在 20% 的数据，$n \leq 2000, m = 64$，不存在修改操作。
- 存在 20% 的数据，$n \leq 30000, m = 8$，树退化成链。
- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。

# 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

$q$ 次操作，每次要么修改点权，要么查询有多少非空连通子树的点权异或和为 $k$。

- 存在 20% 的数据，$n \leq 2000, m = 64$，不存在修改操作。
- 存在 20% 的数据，$n \leq 30000, m = 8$，树退化成链。
- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。
- 存在 25% 的数据，$n \leq 30000, m = 4$。

## 切树游戏 (cut.c/cpp/pas)

给定一棵 $n$ 个点的无根树，权值范围为 $[0, m)$。

$q$ 次操作，每次要么修改点权，要么查询有多少非空连通子树的点权异或和为 $k$。

- 存在 20% 的数据，$n \leq 2000, m = 64$，不存在修改操作。
- 存在 20% 的数据，$n \leq 30000, m = 8$，树退化成链。
- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。
- 存在 25% 的数据，$n \leq 30000, m = 4$。
- 对于 100% 的数据，$n, q \leq 30000, m \leq 128$，修改操作不超过 10000 个。

# 20 分做法 1

- 存在 $20\%$ 的数据，$n \leq 2000, m = 64$，不存在修改操作。

# 20 分做法 1

- 存在 20% 的数据，$n \leq 2000, m = 64$，不存在修改操作。
- 树形 DP，设 $f[i][j]$ 表示考虑 $i$ 的子树，$i$ 点必选，且异或和为 $j$ 的方案数。

# 20 分做法 1

- 存在 20% 的数据，$n \le 2000, m = 64$，不存在修改操作。
- 树形 DP，设 $f[i][j]$ 表示考虑 $i$ 的子树，$i$ 点必选，且异或和为 $j$ 的方案数。
- $ans[k] = \sum_{i=1}^{n} f[i][k]$。

# 20 分做法 1

- 存在 20% 的数据，$n \leq 2000, m = 64$，不存在修改操作。
- 树形 DP，设 $f[i][j]$ 表示考虑 $i$ 的子树，$i$ 点必选，且异或和为 $j$ 的方案数。
- $ans[k] = \sum_{i=1}^{n} f[i][k]$。
- 时间复杂度 $O(nm^2)$。

## 20 分做法 2

- 存在 20% 的数据，$n \leq 30000, m = 8$，树退化成链。

## 20 分做法 2

- 存在 20% 的数据，$n \leq 30000, m = 8$，树退化成链。
- 每个非空连通子树都等价于一个区间。

# 20 分做法 2

- 存在 20% 的数据，$n \le 30000, m = 8$，树退化成链。
- 每个非空连通子树都等价于一个区间。
- 线段树维护整个序列，每个节点维护：
  pre[i]：前缀异或和为 $i$ 的方案数。
  suf[i]：后缀异或和为 $i$ 的方案数。
  sum[i]：子区间异或和为 $i$ 的方案数。
  all：区间异或和。

# 20 分做法 2

- 存在 $20\%$ 的数据，$n \le 30000, m = 8$，树退化成链。
- 每个非空连通子树都等价于一个区间。
- 线段树维护整个序列，每个节点维护：
  pre[i]：前缀异或和为 $i$ 的方案数。
  suf[i]：后缀异或和为 $i$ 的方案数。
  sum[i]：子区间异或和为 $i$ 的方案数。
  all：区间异或和。
- 单次合并信息的复杂度为 $O(m^2)$。

# 20 分做法 2

- 存在 $20\%$ 的数据，$n \le 30000, m = 8$，树退化成链。
- 每个非空连通子树都等价于一个区间。
- 线段树维护整个序列，每个节点维护：
  pre[i]：前缀异或和为 $i$ 的方案数。
  suf[i]：后缀异或和为 $i$ 的方案数。
  sum[i]：子区间异或和为 $i$ 的方案数。
  all：区间异或和。
- 单次合并信息的复杂度为 $O(m^2)$。
- 预处理 $O(nm^2)$，修改 $O(m^2 \log n)$，查询 $O(1)$。

## 30 分做法

- 存在 10% 的数据，$n \le 30000, m = 128$，树退化成链。

# 30 分做法

- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。
- 注意到信息的合并等价于异或卷积。

# 30 分做法

- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。
- 注意到信息的合并等价于异或卷积。
- 将权值进行 FWT 变换，那么在 FWT 意义下的卷积运算复杂度为 $O(m)$。

# 30 分做法

- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。
- 注意到信息的合并等价于异或卷积。
- 将权值进行 FWT 变换，那么在 FWT 意义下的卷积运算复杂度为 $O(m)$。
- 查询时得到答案之后再逆 FWT 变换回来即可。

# 30 分做法

- 存在 10% 的数据，$n \leq 30000, m = 128$，树退化成链。
- 注意到信息的合并等价于异或卷积。
- 将权值进行 FWT 变换，那么在 FWT 意义下的卷积运算复杂度为 $O(m)$。
- 查询时得到答案之后再逆 FWT 变换回来即可。
- 预处理 $O(nm\log m)$，修改 $O(m\log m + m\log n)$，查询 $O(m\log m)$。

# 65 分做法

- 存在 $25\%$ 的数据，$n \leq 30000, m = 4$。

# 65 分做法

- 存在 25% 的数据，$n \leq 30000, m = 4$。
- 将树形 DP 写成异或卷积的形式，则 $f[x] = a[x] \prod (f[son] + 1)$。

# 65 分做法

- 存在 25% 的数据，$n \leq 30000, m = 4$。
- 将树形 DP 写成异或卷积的形式，则 $f[x] = a[x] \prod (f[son] + 1)$。
- 将树进行轻重链剖分，并将重儿子和轻儿子的转移区分开。

# 65 分做法

- 存在 25% 的数据，$n \le 30000, m = 4$。
- 将树形 DP 写成异或卷积的形式，则 $f[x] = a[x]\prod(f[son]+1)$。
- 将树进行轻重链剖分，并将重儿子和轻儿子的转移区分开。
- 设 $g[x]$ 表示轻儿子的 $f+1$ 的卷积，$h[x] = a[x]g[x]$，则 $f[x] = h[x](f[heavy]+1)$。

## 65 分做法

- 存在 25% 的数据，$n \leq 30000, m = 4$。
- 将树形 DP 写成异或卷积的形式，则 $f[x] = a[x] \prod (f[son] + 1)$。
- 将树进行轻重链剖分，并将重儿子和轻儿子的转移区分开。
- 设 $g[x]$ 表示轻儿子的 $f + 1$ 的卷积，$h[x] = a[x]g[x]$，则 $f[x] = h[x](f[heavy] + 1)$。
- 在轻重链剖分之后，同一条重链上的点在 DFS 序上会按深度从小到大连续排列在一起。

# 65 分做法

- 存在 25% 的数据，$n \le 30000, m = 4$。
- 将树形 DP 写成异或卷积的形式，则 $f[x] = a[x] \prod (f[son] + 1)$。
- 将树进行轻重链剖分，并将重儿子和轻儿子的转移区分开。
- 设 $g[x]$ 表示轻儿子的 $f + 1$ 的卷积，$h[x] = a[x]g[x]$，则 $f[x] = h[x](f[heavy] + 1)$。
- 在轻重链剖分之后，同一条重链上的点在 DFS 序上会按深度从小到大连续排列在一起。
- 故 $f[x] = h[x] + h[x]h[x+1] + \ldots + \prod_{i=0}^{len} h[x+i]$。

# 65 分做法

- 那么答案等价于每条重链上每个子区间的 $h$ 的乘积之和，线段树维护。

## 65 分做法

- 那么答案等价于每条重链上每个子区间的 $h$ 的乘积之和，线段树维护。
- 修改 $x$ 的点权时，首先重新计算 $x$ 所在重链顶端节点 $y$ 的 DP 值以及整条重链的贡献。

# 65 分做法

- 那么答案等价于每条重链上每个子区间的 $h$ 的乘积之和，线段树维护。
- 修改 $x$ 的点权时，首先重新计算 $x$ 所在重链顶端节点 $y$ 的 DP 值以及整条重链的贡献。
- 其次需要重新计算 $h[father[y]]$，这需要另外按 BFS 序维护线段树来进行查询。

# 65 分做法

- 那么答案等价于每条重链上每个子区间的 $h$ 的乘积之和，线段树维护。
- 修改 $x$ 的点权时，首先重新计算 $x$ 所在重链顶端节点 $y$ 的 DP 值以及整条重链的贡献。
- 其次需要重新计算 $h[father[y]]$，这需要另外按 BFS 序维护线段树来进行查询。
- 每次修改一共修改 $O(\log n)$ 条重链。

# 65 分做法

- 那么答案等价于每条重链上每个子区间的 $h$ 的乘积之和，线段树维护。
- 修改 $x$ 的点权时，首先重新计算 $x$ 所在重链顶端节点 $y$ 的 DP 值以及整条重链的贡献。
- 其次需要重新计算 $h[father[y]]$，这需要另外按 BFS 序维护线段树来进行查询。
- 每次修改一共修改 $O(\log n)$ 条重链。
- 预处理 $O(nm^2)$，修改 $O(m^2 \log^2 n)$，查询 $O(1)$。

# 100 分做法

- 65 分做法的瓶颈在于合并信息，以及 $h[father[y]]$ 的计算。

# 100 分做法

- 65 分做法的瓶颈在于合并信息，以及 $h[father[y]]$ 的计算。
- 在 FWT 意义下，合并信息可以做到 $O(m)$。

# 100 分做法

- 65 分做法的瓶颈在于合并信息，以及 $h[father[y]]$ 的计算。
- 在 FWT 意义下，合并信息可以做到 $O(m)$。
- 而 $h[father[y]]$ 的计算，只需要维护非 0 部分的积以及 0 的个数。

# 100 分做法

- 65 分做法的瓶颈在于合并信息，以及 $h[father[y]]$ 的计算。
- 在 FWT 意义下，合并信息可以做到 $O(m)$。
- 而 $h[father[y]]$ 的计算，只需要维护非 0 部分的积以及 0 的个数。
- 考虑 $h[y]$ 对其的贡献，要么贡献非 0 部分，要么贡献一个 0，都是可逆的操作。

# 100 分做法

- 65 分做法的瓶颈在于合并信息，以及 $h[father[y]]$ 的计算。
- 在 FWT 意义下，合并信息可以做到 $O(m)$。
- 而 $h[father[y]]$ 的计算，只需要维护非 0 部分的积以及 0 的个数。
- 考虑 $h[y]$ 对其的贡献，要么贡献非 0 部分，要么贡献一个 0，都是可逆的操作。
- 预处理 $O(nm\log m)$，修改 $O(m\log m + m\log^2 n)$，查询 $O(m\log m)$。

# Thank you!