2019 学军中学省选模拟赛

SGColin

目录

1 2019 省选联合训练 5

1.1 A. Base

1.1.1 Description

原题链接：[CodeForces 528 D] Fuzzy Search

给出两个只包含"Z,P,S,B" 的字符串 S，T，定义 T 串在 S 串中第 i 个位置匹配，当且仅当 $\forall p \le |T|, B_p \in \{A_{i+p-k}，\ldots, A_{i+p+k}\}$，问总匹配的位置数。

数据范围：$|S|, |T|, k \le 2 \times 10^5$

1.1.2 Solution

1.2 B. Xor

1.2.1 Description

类题链接：[CodeForces 888 G] Xor-MST

有一张 n 个点的完全图，每个点有一个 m 位二进制数的点权 v_i，连接 i 号点和 j 号点的边权为 $v_i \oplus v_j$。q 次询问，每次将所有点权加 Δ 并对 2^m 取模，问之后这张图的最小生成树边权和，询问中对点权的改变有后效性。

数据范围：$n, q \le 2 \times 10^4$，$m \le 14$。

1.2.2 Solution

首先如果存在点权相同的点，我们一定是在他们之间连边，边权为 0，因此只需要考虑不同点权的点之间的最小生成树。

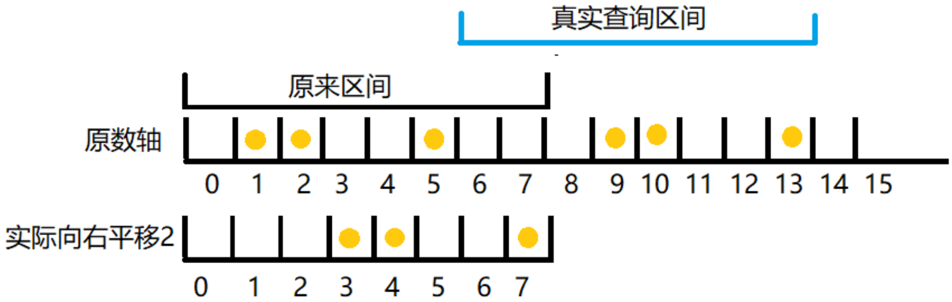
如果不带修，做法和 [CodeForces 888 G] Xor-MST 是一样的，考虑从高到低按位分治。将当前分治集合划分成当前位是 0 或 1 的两个集合，因为 2 的幂次特殊性，我们一定会让两个子集内部构成生成树，然后只连一条跨越两个子集的边，用 Trie 树求出这条边可能的最小边权即可。

事实上我们的操作每次都是将一个长度为 2^i 的连续值域段分成了两个长度为 2^{i-1} 的连续值域段，因此我们可以记录下来一个值域上的分治树（01Trie），每层节点的长度都是 2 的整次幂。那么就相当于是维护一棵 Trie，支持 Trie 在数轴上移动（模意义下），以及查询 Trie 上分治求出的 MST 的结果。

因为点权对 2^m 取模，因此得到的本质不同的点集只有 2^m 个。因此考虑把 $\Delta = 0, \dots, 2^m - 1$ 对应的 Trie 结构都记录下来。为了避免 Trie 在模意义下移动出现的分成两部分，我们将数轴倍长一份（变成长度为 2^{m+1}）接在后面。

我们把点集标记在数轴上，倍长的那部分也对应标记。考虑一次移动 Δ，其对应的 Trie 应该移动到哪里。注意我们维护的是**点集在长度为 2^m 的数轴上的相对位置**，那么将点集点权加 Δ，就相当于相对位置关系向左移动了 Δ。

比如 $m = 3$，初始点集为 $\{1, 2, 5\}$，$\Delta = 2$。那么变化之后的点集就是 $\{3, 4, 7\}$，此时我们要将查询的区间从 $[0, 7]$ 向左平移 2 个位置变成 $[-2, 5]$，因为我们把数轴倍长，所以在模意义下就是查区间 $[6, 13]$。可以发现值域区间的右移，实际上就是原数轴上查询区间的左移。



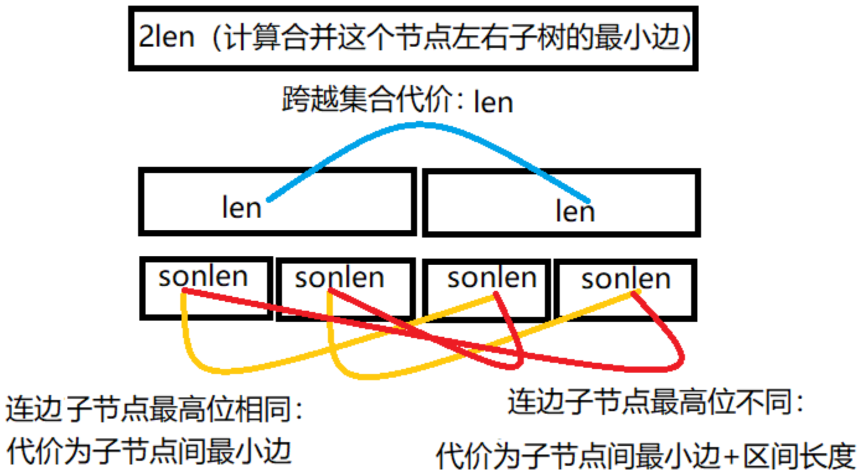
下面实现分治树的建树，这里我们每个节点都有可能作为多个节点的子节点。设 $f[i][x]$ 表示值域在 $[x, x+2^i-1]$ 这段区间内的分治节点编号，那么只需要初始化节点 $f[0][i], i \in S$，然后像 ST 表一样倍增的构造子树结构即可。

```
//s: 分治节点内的点集状压
//ans: 记忆化当前节点的子树是否计算过答案，-1代表未计算过
//ls, rs: 分治树上的左右子树
struct node{int s, ans, ls, rs;} c[N << 5];
inline void Build(int len) {
  //对点集中的点权建叶子
  for (int i = 0; i < len; ++i)
    if (app[i]) f[0][i] = ++ptr, c[ptr] = node(1, -1, 0, 0);
  //倍增建树
  for (int j = 1, k = 1; k < len; ++j, k <<= 1)
    for (int i = 0, ls, rs; i + k < len; ++i) {
      ls = f[j - 1][i];
      rs = f[j - 1][i + k];
      if (!ls && !rs) continue;
      f[j][i] = ++ptr;
      c[ptr] = node(((c[rs].s << k) | c[ls].s), -1, ls, rs);
    }
}
```

设 $dlt = \sum \Delta$，那么每次我们查询的**分治树根节点**就是 $f[m][dlt]$。接下来需要实现点集 MST 查询，按照不带修的思路，有代价构成：左子树代价 + 右子树代价 + 左右子树连通代价。

```
int Mst(int rt, int len) {
  if (c[rt].ans != -1) return c[rt].ans; //记忆化
  //对于边界小点集直接预处理
  if (len <= 16) return c[rt].ans = mst[c[rt].s];
  int sonlen = len >> 1;
  //左子树或右子树为空则不在当前点产生跨越集合的代价
  if (!c[rt].ls) return c[rt].ans = Mst(c[rt].rs, sonlen);
  if (!c[rt].rs) return c[rt].ans = Mst(c[rt].ls, sonlen);
  //跨域当前点代价构成：左子树代价 + 右子树代价 + 左右子树连通代价
  c[rt].ans = Mst(c[rt].ls, sonlen) + Mst(c[rt].rs, sonlen);
  c[rt].ans += Min_edge(c[rt].ls, c[rt].rs, sonlen) + sonlen;
  return c[rt].ans;
}
```

下面我们考虑将左右子树连通的最小边权如何处理。首先固定的代价就是跨越集合的代价，即子节点区间长度 len。剩余部分的最小化我们再分成子树的子树讨论，有




```
inline int Min_edge(int rt1, int rt2, int len) {
  if (!rt1 || !rt2) return inf; //不存在此情况
  //对于边界小点集间的最小边直接暴力预处理
  if (len == 8) return min_edge[c[rt1].s][c[rt2].s];
  int res = inf, sonlen = len >> 1;
  //同侧子节点查询
  res = min(res, Min_edge(c[rt1].ls, c[rt2].ls, sonlen));
  res = min(res, Min_edge(c[rt1].rs, c[rt2].rs, sonlen));
  if (res < inf) return res;
  //异侧子节点查询
  res = min(res, Min_edge(c[rt1].ls, c[rt2].rs, sonlen) + sonlen);
  res = min(res, Min_edge(c[rt1].rs, c[rt2].ls, sonlen) + sonlen);
  return res;
}
```

此题的实现就基本完成了。暴力预处理用的是暴力枚举和 Prim，其他细节建议参考代码。

关于复杂度分析，只能给出一个均摊的思路。预处理部分的复杂度为 $\mathcal{O}(size^2)$，所有分治树的节点个数为 $\mathcal{O}(n\log n)$，每个节点计算合并子节点的复杂度是节点代表的区间大小，加上预处理的小边界 $size$ 优化，最后的复杂度大概在

$$\mathcal{O}(\frac{n^2\log n}{size})$$

1.3 C. FFT

1.3.1 Description

原题链接：[AtCoder Grand Contest 019 F] Yes or No

1.3.2 Solution