memset0's Notebook

方知蓦然回首之时
那人却已不在灯火阑珊处

关于我
友情链接
文章聚合

LOJ6287 诗歌

2019-02-28 | 题解

一道非常巧妙的哈希题。

假设我们枚举 j 使得 $H_i - H_j = H_j - H_k$ 。可以发现题意要求的 $i < j < k$ 即 i 和 k 必须在 j 的两侧。

可以通过线段树 + 哈希维护

- 哈希值使得第 k 位的哈希值为 1 当且仅当满足 $H_i - H_j = k$ 的 i 在 j 的左侧出现；
- 哈希值使得第 k 位的哈希值为 0 当且仅当满足 $H_j - H_i = k$ 的 i 在 j 的右侧出现。

容易发现，若两哈希值不同，即存在一组 i 、 j 、 k 满足条件，我们直接输出 Yes 即可。

另外据称本地数据很弱，乱搞也能过 233 。

代码：

```
// ================================
//   author: memset0
```

memset0's Notebook

方知蓦然回首之时
那人却已不在灯火阑珊处

关于我
友情链接
文章聚合


```
//   date: 2019.02.26 14:23:10
//   website: https://memset0.cn/
// ================================
#include <bits/stdc++.h>
#define ll long long
namespace ringo {
template <class T> inline void read(T &x) {
    x = 0; register char c = getchar(); register bool f = 0;
    while (!isdigit(c)) f ^= c == '-', c = getchar();
    while (isdigit(c)) x = x * 10 + c - '0', c = getchar();
    if (f) x = -x;
}
template <class T> inline void print(T x) {
    if (x < 0) putchar('-'), x = -x;
    if (x > 9) print(x / 10);
    putchar('0' + x % 10);
}
template <class T> inline void print(T x, char c) { print(x), putchar(c); }

const int N = 3e5 + 10, mod1 = 998244353, mod2 = 1e9 + 7;
int n, m;
int a[N];

inline int dec(int a, int b, int p) { a -= b; return a < 0 ? a + p : a; }
inline int sub(int a, int b, int p) { a += b; return a >= p ? a - p : a; }
inline int mul(int a, int b, int p) { return (ll)a * b - (ll)a * b / p * p; }

struct hash {
    int a, b;
    inline hash() {}
    inline hash(int x) { a = b = x; }
    inline hash(int x, int y) { a = x, b = y; }
    friend inline hash operator + (const hash &a, const hash &b) { return has
    friend inline hash operator - (const hash &a, const hash &b) { return has
    friend inline hash operator * (const hash &a, const hash &b) { return has
    friend inline bool operator == (const hash &a, const hash &b) { return a.
} L, R, pow[N];
```

memset0's Notebook

方知暮然回首之时
那人却已不在灯火阑珊处

关于我

友情链接

文章聚合


```
struct node {
    int l, r, mid, siz;
    hash lsh, rsh;
} p[N << 2];

inline void maintain(int u) {
    p[u].lsh = p[u << 1].lsh * pow[p[u << 1 | 1].siz] + p[u << 1 | 1].lsh;
    p[u].rsh = p[u << 1].rsh + p[u << 1 | 1].rsh * pow[p[u << 1].siz];
}

void build(int u, int l, int r) {
    p[u].l = l, p[u].r = r, p[u].mid = (l + r) >> 1, p[u].siz = r - l + 1;
    if (l == r) { p[u].lsh = p[u].rsh = 0; return; }
    build(u << 1, l, p[u].mid), build(u << 1 | 1, p[u].mid + 1, r);
    maintain(u);
}

void modify(int u, int k) {
    if (p[u].l == p[u].r) { p[u].lsh = p[u].rsh = 1; return; }
    modify(k <= p[u].mid ? u << 1 : u << 1 | 1, k);
    maintain(u);
}

int query(int u, int k) {
    if (p[u].l == p[u].r) return p[u].lsh.a;
    return query(k <= p[u].mid ? u << 1 : u << 1 | 1, k);
}

#define LSH 0
#define RSH 1
hash query(int u, int l, int r, bool flag) {
    if (l == p[u].l && r == p[u].r) return flag == LSH ? p[u].lsh : p[u].rsh;
    if (r <= p[u].mid) return query(u << 1, l, r, flag);
    if (l > p[u].mid) return query(u << 1 | 1, l, r, flag);
    return flag == LSH ? query(u << 1, l, p[u].mid, flag) * pow[r - p[u].mid]
        query(u << 1, l, p[u].mid, flag) + query(u << 1 | 1, p[u].mid + 1, r,
```

```
}

void main() {
    read(n), pow[0] = 1, build(1, 1, n);
    for (int i = 1; i <= n; i++) read(a[i]);
    for (int i = 1; i <= n; i++) pow[i] = pow[i - 1] * hash(131);
    for (int i = 1, len; i <= n; i++) {
        len = std::min(a[i] - 1, n - a[i]);
        if (len) {
            L = query(1, a[i] - len, a[i] - 1, LSH);
            R = query(1, a[i] + 1, a[i] + len, RSH);
            if (!(L == R)) { puts("YES"); return; }
        }
        modify(1, a[i]);
    } puts("NO");
}

} signed main() { return ringo::main(), 0; }
```

巧妙的思路 线段树 哈希

用户名

邮箱

网址（选填）

可以在这里写评论哦 ~

提交评论

LJOJ5249 老夫
上一篇 «

多项式复合逆学习笔记
» 下一篇

浙ICP备19006255号-1
97717 visits · 24757 visitors · 74.48 W words

在这里输入关键字哦 ~（回车搜索）

memset0's Notebook

方知暮然回首之时
那人却已不在灯火阑珊处

关于我
友情链接
文章聚合

Theme Ringo by memset0
Proudly powered by Typecho

↑