A

```
#include <bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()

using namespace std;

typedef long long ll;

const int MAXN = (int) 1e6 + 5;
const int MOD = (int) 1e9 + 7;

int a[MAXN];
int d[MAXN];
int n;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n;

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    sort(a + 1, a + n + 1);

    if (a[n] <= 0) {
        cout << "1\n";
        return 0;
    }

    int t = n;

    while (t > 2 && a[t - 2] + a[t - 1] > a[t] && a[t - 1] > 0) {
        t--;
    }

    d[1] = (int) 2e9;

    for (int i = 2; i <= n; i++) {
        d[i] = a[i] - a[i - 1];
    }
    vector<pair<int, int>> values;

    for (int i = 1; i <= n; i++) {
        values.pb({d[i], i});
    }

    sort(all(values));
    set<int> S;

    for (int i = 1; i <= n; i++) {
        S.insert(i);
    }

    ll ans = 0;
    int ptr = 0;

    for (int j = t; j <= n; j++) {
        while (ptr < sz(values) && values[ptr].first < a[j]) {
            int idx = values[ptr].second;
            S.erase(idx);
            ptr++;
        }

        int i = *prev(S.lower_bound(j));
        ans += (j - i);
    }

    cout << ans<< '\n';
    return 0;
}
```

B

```
#include <bits/stdc++.h>

#define pb push_back
#define mp make_pair
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()

using namespace std;

typedef long long ll;

const int MAXN = (int)1e6 + 5;
```

```
//vector<vector<int>> where(MAXN);
vector<int> where[MAXN];
int a[MAXN];
int n;

int index_lim;
int fenw_max[MAXN];

int pref_max(int p) {
    int res = 0;

    for (; p > 0; p -= p & -p) {
        res = max(res, fenw_max[p]);
    }

    return res;
}

void update_max(int p, int x) {
    for (; p <= n; p += p & -p) {
        fenw_max[p] = max(fenw_max[p], x);
    }
}

void add(int x) {
    int prv = 0;

    for (int pos : where[x]) {
        update_max(prv + 1, pos);
        prv = pos;
    }

    index_lim = min(index_lim, prv);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n;

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        where[a[i]].pb(i);
    }

    index_lim = n;
    for (int x = 0; x <= n - 1; x++) {
        add(x);

        int i = 1, ans = 0;

        while (i <= n) {
            int j;

            if (i > index_lim) {
                j = n + 1;
            }
            else {
                j = pref_max(i);
            }

            ans++;

            if (j == i) {
                ans = -1;
                break;
            }

            i = j;
        }

        cout << ans << " \n"[x == n];
    }

//    cerr << (double)clock() / CLOCKS_PER_SEC << endl;
    return 0;
}
```

C

```
// #pragma GCC optimize("-Ofast")
#include <bits/stdc++.h>

#define pb push_back
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()

using namespace std;

typedef long long ll;

const int MAXN = (int)1e5 + 5;
const int K = 300;
const int I = MAXN / K + 2;

int P[I][MAXN], Q[I][I];
ll S[MAXN], T[MAXN];
ll a[MAXN], b[I];
int p[MAXN];
int n, m, q;

void sqrtUpdI(int p, ll x) {
	S[p] += x;
	T[p / K] += x;
}

void sqrtUpd(int id, int l, int r, int x) {
	int cl = l / K, cr = r / K;

	if (cl == cr) {
		for (int i = l; i <= r; i++) {
			P[id][i] += x;
		}

		return;
	}

	for (int i = l, j = (cl + 1) * K; i < j; i++) {
		P[id][i] += x;
	}

	for (int i = cl + 1; i < cr; i++) {
		Q[id][i] += x;
	}

	for (int i = cr * K; i <= r; i++) {
		P[id][i] += x;
	}
}

ll sqrtGetI(int l, int r) {
	int cl = l / K, cr = r / K;
	ll ret = 0;

	if (cl == cr) {
		for (int i = l; i <= r; i++) {
			ret += S[i];
		}

		return ret;
	}
```

```
        for (int i = l, j = (cl + 1) * K; i < j; i++) {
                ret += S[i];
        }

        for (int i = cl + 1; i < cr; i++) {
                ret += T[i];
        }

        for (int i = cr * K; i <= r; i++) {
                ret += S[i];
        }

        return ret;
}

int sqrtGet(int id, int p) {
        return P[id][p] + Q[id][p / K];
}

int sqrtGet(int id, int l, int r) {
        return sqrtGet(id, r) - (l ? sqrtGet(id, l - 1) : 0);
}

void build() {
        for (int i = 0; i * K < n; i++) {
                int l = i * K, r = min(n, (i + 1) * K);

                for (int j = l; j < r; j++) {
                        int x = p[j];
                        int id = x / K;
                        ++P[i][x];
                        --P[i][min(n, (id + 1) * K)];
                        ++Q[i][id + 1];
                }

                for (int j = 1; j < n; j++) {
                        P[i][j] += P[i][j - 1];
                }

                for (int j = 1; j < l; j++) {
                        Q[i][j] += Q[i][j - 1];
                }
        }
}

void update(int l, int r, int x) {
        int cl = l / K, cr = r / K;
        if (cl == cr) {
                for (int i = l; i <= r; i++) {
                        sqrtUpdl(p[i], x);
                }

                return;
        }

        for (int i = l, j = (cl + 1) * K; i < j; i++) {
                sqrtUpdl(p[i], x);
        }

        for (int i = cl + 1; i < cr; i++) {
                b[i] += x;
        }

        for (int i = cr * K; i <= r; i++) {
                sqrtUpdl(p[i], x);
        }
}

ll query(int l, int r) {
        ll res = sqrtGetl(l, r);

        for (int i = 0; i < l; i++) {
                res += b[i] * sqrtGet(i, l, r);
        }

        return res;
}

void push(int t) {
        if (!b[t]) {
                return;
        }

        int l = t * K, r = min(n, (t + 1) * K);

        for (int i = l; i < r; i++) {
                sqrtUpdl(p[i], b[t]);
        }

        b[t] = 0;
}

void exchange(int a, int b) {
        int ca = a / K, cb = b / K;
```

```
	if (ca != cb) {
		push(ca);
		push(cb);

		if (p[a] < p[b]) {
			sqrtUpd(ca, p[a], p[b] - 1, -1);
			sqrtUpd(cb, p[a], p[b] - 1, 1);
		}
		else {
			sqrtUpd(ca, p[b], p[a] - 1, 1);
			sqrtUpd(cb, p[b], p[a] - 1, -1);
		}
	}

	swap(p[a], p[b]);
}

int main() {
	ios::sync_with_stdio(0);
	cin.tie(0);
	cin >> n >> q;

	for (int i = 0; i < n; i++) {
		cin >> p[i];
		p[i]--;
	}

	build();

	for (int i = 1; i <= q; i++) {
		int tp;
		cin >> tp;

		if (tp == 1) {
			int l, r, x;
			cin >> l >> r >> x;
			l--;
			r--;
			update(l, r, x);
		}
		else if (tp == 2) {
			int l, r;
			cin >> l >> r;
			l--;
			r--;
			cout << query(l, r) << '\n';
		}
```

```
                else {
                        int a, b;
                        cin >> a >> b;
                        a--;
                        b--;
                        exchange(a, b);
                }
        }

        return 0;
}
```