Merge Sort feladatok és megoldások

Szerző: Nikházy László
Előadó: Varga Péter

2025. október 24.

1. feladat: Inverziók száma (Inversion Count)

https://www.spoj.com/problems/INVCNT/

Adott egy $A[0 \ldots n-1]$ tömb n darab különböző pozitív egész számmal. Ha $i < j$ és $A[i] > A[j]$, akkor az (i, j) párt **inverziónak** nevezzük. Készíts programot, amely meghatározza a tömbben található inverziók számát!

Bemenet

A standard bemenet első sora egy t számot tartalmaz — a tesztesetek számát.
Ezután minden teszteset az alábbi formátumban érkezik:

- egy sorban az n elem száma ($1 \leq n \leq 200\,000$),
- a következő n sor mindegyike egy $A[i]$ számot tartalmaz ($1 \leq A[i] \leq 10^7$),
- ezután egy üres sor következik.

Kimenet

Minden tesztesetre egy sorban írjuk ki az inverziók számát!

Példa

Bemenet	Kimenet
2 3 3 1 2 5 2 3 8 6 1	2 5

Korlátok

$1 \leq t \leq 10$, $1 \leq n \leq 200\,000$, $1 \leq A[i] \leq 10^7$
Időlimit: 1.5 másodperc
Memórialimit: 1536 MB

Inverziók száma - megoldási ötlet

Oszd meg és uralkodj: Az inverziók száma három részre bontható:

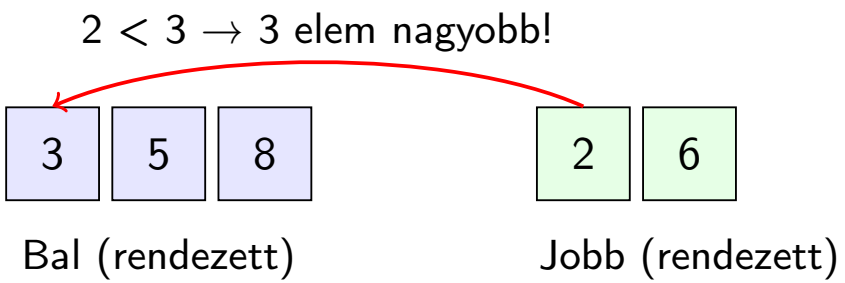
- az inverziók a **bal oldali** részben,
- az inverziók a **jobb oldali** részben,
- és a **keresztinverziók**, amikor az egyik elem balról, a másik jobbról származik.

A bal és jobb részeket rekurzívan rendezzük és számoljuk, a **keresztinverziókat pedig az összefésülés során** határozzuk meg.

Inverziók száma – Keresztinverziók az összefésülésnél

Cél: Az összefésülés során számoljuk meg, hány **keresztinverzió** keletkezik.

Ötlet: Amikor a jobb rész egy eleme kerül be a végső sorozatba, miközben a bal rész aktuális eleme nagyobb nála, akkor a bal részben **minden hátralevő elem inverziót alkot** ezzel a jobboldali elemmel.



Mivel $2 < 3$, a bal részben a **3, 5, 8** mind nagyobb, tehát ezek mind inverziót alkotnak 2-vel.

$\Rightarrow$ Hozzáadunk **3 inverziót** a számlálóhoz.

Inverziók száma – kód

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

// Két rendezett résztömböt összefésül és közben megszámolja az inverziókat
ll mergeAndCount(vector<int>& a, int l, int m, int r) {
    vector<int> left(a.begin() + l, a.begin() + m + 1);
    vector<int> right(a.begin() + m + 1, a.begin() + r + 1);

    int i = 0, j = 0, k = l;
    ll invCount = 0;

    while (i < left.size() && j < right.size()) {
        if (left[i] <= right[j]) {
            a[k++] = left[i++];
        } else {
            // minden hátralévő bal oldali elem inverziót alkot right[j]-vel
            invCount += (left.size() - i);
            a[k++] = right[j++];
        }
    }

    while (i < left.size()) a[k++] = left[i++];
    while (j < right.size()) a[k++] = right[j++];

    return invCount;
}
```

Inverziók száma – kód

```
1
2   ll mergeSortAndCount(vector<int>& a, int l, int r) {
3       if (l >= r) return 0;
4       int m = (l + r) / 2;
5       ll inv = 0;
6       inv += mergeSortAndCount(a, l, m);
7       inv += mergeSortAndCount(a, m + 1, r);
8       inv += mergeAndCount(a, l, m, r);
9       return inv;
10  }
11
12  int main() {
13      ios::sync_with_stdio(false);
14      cin.tie(nullptr);
15
16      int t;
17      cin >> t;
18      while (t--) {
19          int n;
20          cin >> n;
21          vector<int> a(n);
22          for (int i = 0; i < n; i++) cin >> a[i];
23          cout << mergeSortAndCount(a, 0, n - 1) << "\n";
24      }
25  }
```

2. feladat: Merge Sort (Codeforces 873D)

`https://codeforces.com/problemset/problem/873/D`

A feladat röviden: keress egy olyan n elemű permutációt, amelyre a **merge sort** algoritmusban pontosan k hívás történik. (Részletes leírás a fenti linken.)

Bemenet

A bemenet két egész számot tartalmaz:

- n — az elemszám ($1 \leq n \leq 100\,000$)
- k — a kívánt hívások száma ($1 \leq k \leq 200\,000$)

Kimenet

Ha nincs megoldás, írjuk ki `-1`-et. Egyébként írjunk ki egy megfelelő permutációt!

Példa

Bemenet	Kimenet
3 3	2 1 3

A `mergesort` hívás háromszor fut le:
`(0,3)`, `(0,1)`, `(1,3)`.

Példa

Bemenet	Kimenet
4 1	1 2 3 4

A tömb eleve rendezett, ezért csak egy hívás történik.

Korlátok

$1 \leq n \leq 100\,000$, $1 \leq k \leq 200\,000$
Időlimit: 2 másodperc
Memórialimit: 256 MB

873D. Merge Sort – megoldási ötlet

Megfigyelés:

- A függvényhívások száma mindig **páratlan** (minden hívás vagy 0, vagy 2 további hívást generál).
- Tehát ha k páros, **nincs megoldás**.

Ötlet: Induljunk a rendezett permutációból, és **„rontsuk el”** fokozatosan az elemeket, hogy nőjön a hívások száma.

Definiáljuk az `unsort(l, r)` függvényt:

- Ha elértük a kívánt hívásszámot, hagyjuk a szakaszt rendezve.
- Különben cseréljük fel a két középső elemet, így a szakasz nem lesz rendezett.
- Ezután rekurzívan meghívjuk `unsort(l, mid)` és `unsort(mid, r)`.

Az `unsort` hívások száma éppen annyi lesz, mint a merge sort hívások száma a kapott permutációra.

873D. Merge Sort – kód

```cpp
#include <bits/stdc++.h>
using namespace std;

int n, k;
vector<int> a;

void unsort(int l, int r) {
    if (r - l <= 1 || k <= 1) return;
    k -= 2; // két új hívást "felhasználunk"
    int m = (l + r) / 2;
    swap(a[m - 1], a[m]); // a közepén felcserélünk két elemet
    unsort(l, m);
    unsort(m, r);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> k;
    if (k % 2 == 0) {
        cout << -1;
        return 0;
    }
    a.resize(n);
    iota(a.begin(), a.end(), 1); // kezdjük rendezett permutációval
    unsort(0, n);
    if (k != 1) cout << -1;
    else for (int i = 0; i < n; i++) cout << a[i] << "␣";
}
```

3. feladat: Intervallumon k-adik legkisebb elem

https://judge.yosupo.jp/problem/range_kth_smallest

Adott egy $a_0, a_1, \ldots, a_{N-1}$ egész számokból álló sorozat és Q darab lekérdezés, mindegyik lekérdezés a következő formátumú: $l_i\ r_i\ k_i$.
Feladat: írjuk ki a $(a_{l_i}, a_{l_i+1}, \ldots, a_{r_i-1})$ részintervallum $k_i + 1$-dik legkisebb elemét.

Bemenet

A standard bemenet:

- Egy sorban az N és Q számok ($1 \leq N \leq 200\,000$, $1 \leq Q \leq 200\,000$).
- Egy sorban N szám: $a_0, a_1, \ldots, a_{N-1}$ ($1 \leq a_i \leq 10^9$).
- Ezután Q sor, mindegyik: $l_i\ r_i\ k_i$ ($0 \leq l_i < r_i \leq N$, $0 \leq k_i < r_i - l_i$).

Kimenet

Minden lekérdezésre egy sorban írjuk ki a k-adik legkisebb elemet!

Példa

Bemenet	Kimenet
5 3 1 4 0 1 3 0 5 2 1 3 1 3 4 0	1 4 1

Korlátok

$1 \leq N \leq 200\,000$, $1 \leq Q \leq 200\,000$, $1 \leq a_i \leq 10^9$
$0 \leq l_i < r_i \leq N$, $0 \leq k_i < r_i - l_i$
Időlimit: 5 másodperc
Memórialimit: 1024 MB

Intervallumon k-adik legkisebb elem - Ötlet

Átalakítás: Rendezzük az (a_i, i) párokat, így megkapjuk az indexeket az értékek szerint növekvő sorrendben. Erre a sorozatra építsünk merge sort tree -t. (Ebben a fában tulajdonképpen az indexek a nagyságrendi sorrendből kiindulva a saját helyükre rendeződnek).

- A lekérdezés arra módosul, hogy a sorozatban k-adik olyan elem kell, ami az (l, r) intervallumban van.
- A fa segítségével ezt meg tudjuk válaszolni: nézzük meg, hogy a sorozat bal felében hány ilyen van $\rightarrow$ kiderül, hogy melyik felében kell keresni.
- Egy-egy csúcsban tárolt rendezett sorozatban két bináris kereséssel meghatározható, hogy hány elem esik az (l, r) intervallumba.

Intervallumon k-adik legkisebb elem - Példa

Legyen $n = 7$, $a = (2, 3, 1, 6, 5, 3, 5)$ és a lekérdezés: $l = 2$, $r = 6$, $k = 2$, ekkor az $(1, 6, 5, 3)$ részben keressük a harmadik legkisebb elemet, tehát a válasz $a_4 = 5$ lesz.

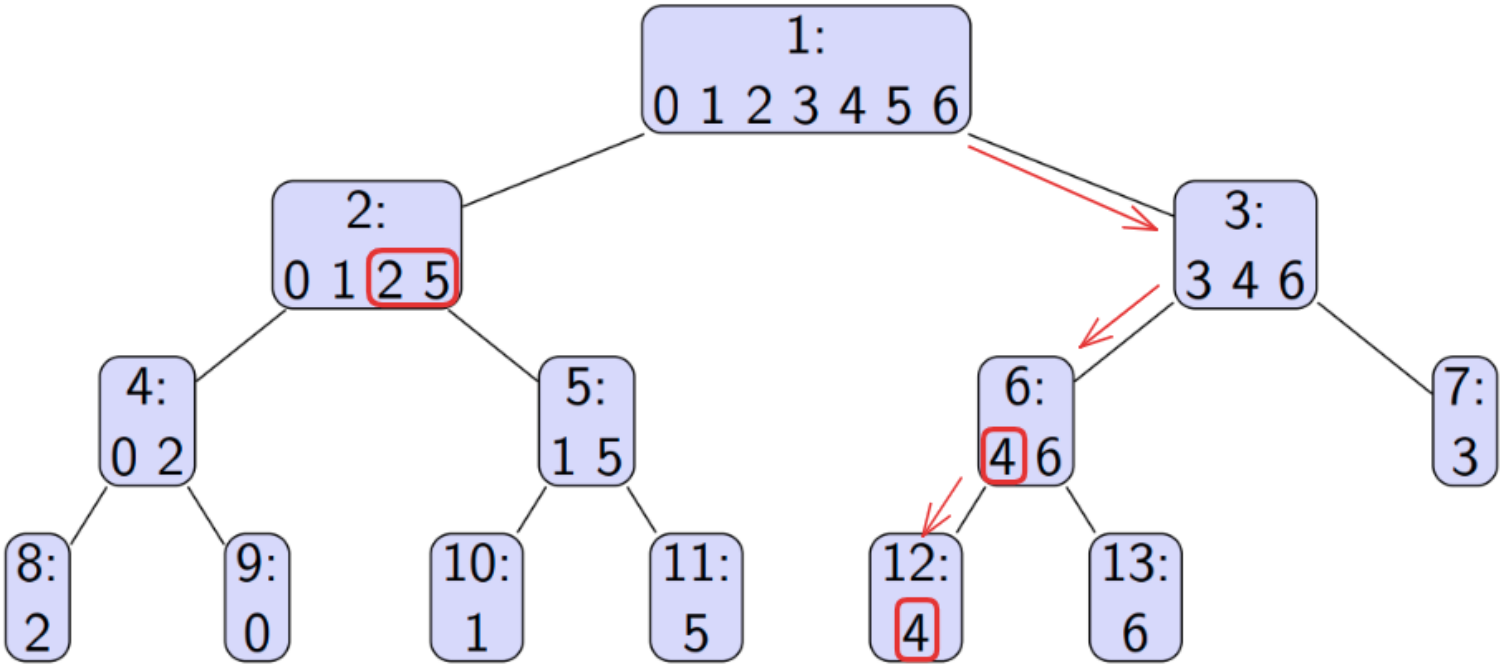
a_i	2	3	1	6	5	3	5
i	0	1	2	3	4	5	6

Rendezve:

a_i	1	2	3	3	5	5	6
i	**2**	0	1	**5**	**4**	6	**3**

A legalsó indexsorozatban a kérdés úgy alakul át, hogy melyik a k-adik elem, ami a $[2, 6)$ intervallumban van. Erre fogjuk alkalmazni a merge sort tree-t.

Intervallumon k-adik legkisebb elem - Példa



A gyökér balgyerekében csak két elem van, ami a $[2, 6)$ intervallumban van, mi a harmadikat keressük, ezért jobbra lépünk, ahol már az első elemet keressük majd ebben az intervallumban.

Intervallumon k-adik legkisebb elem – kód

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 200000;
vector<int> tree[MAXN * 4];
vector<int> a;

// Build a Merge Sort Tree
void buildTree(int node, int start, int end) {
    if (start == end) { tree[node].push_back(a[start]); return; }
    int mid = (start + end) / 2;
    buildTree(2 * node, start, mid);
    buildTree(2 * node + 1, mid + 1, end);
    merge(tree[2 * node].begin(), tree[2 * node].end(),
        tree[2 * node + 1].begin(), tree[2 * node + 1].end(),
        back_inserter(tree[node]));
}

int queryKth(int node, int start, int end, int l, int r, int k) {
    if (start == end) return tree[node][0];
    int mid = (start + end) / 2;
    // finds the last index in the segment which is <= r
    int last = upper_bound(tree[2 * node].begin(), tree[2 * node].end(), r)
        - tree[2 * node].begin();
    // finds the first index in the segment which is >= l
    int first = lower_bound(tree[2 * node].begin(), tree[2 * node].end(), l)
        - tree[2 * node].begin();
    int m = last - first;
    if (m >= k) {
        return queryKth(2 * node, start, mid, l, r, k);
    } else {
        return queryKth(2 * node + 1, mid + 1, end, l, r, k - m);
    }
}
```

Intervallumon k-adik legkisebb elem – kód

```
1
2   int main() {
3       ios::sync_with_stdio(false);
4       cin.tie(nullptr);
5       int n, q;
6       cin >> n >> q;
7       vector<int> x(n);
8       vector<pair<int, int>> p(n);
9       for (int i = 0; i < n; i++) {
10          cin >> x[i];
11          p[i] = {x[i], i};
12      }
13      sort(p.begin(), p.end());
14      a.resize(n);
15      for (int i = 0; i < n; i++) a[i] = p[i].second;
16      buildTree(1, 0, n - 1);
17      while (q--) {
18          int l, r, k;
19          cin >> l >> r >> k;
20          cout << x[queryKth(1, 0, n - 1, l, r - 1, k + 1)] << "\n";
21      }
22  }
```

4. feladat: Hold the Line

https://codeforces.com/gym/102452/problem/H

Feladat: Tartsunk nyilván egy $h_1, h_2, \ldots, h_N$ tömböt, kezdetben minden elem 0. M lekérdezés van az alábbi két típusból:

1. lecserélünk egy nullát pozitív értékre, vagy
2. egy (l, r) intervallumon lekérdezzük adott H-ra a $\min |h_i - H|$ értéket $(l \leq i \leq r)$

Korlátok

$1 \leq T \leq 10^5$, $\sum N \leq 5 \cdot 10^5$, $\sum M \leq 10^6$ $1 \leq h_i, H \leq 10^9$ **Időlimit**: 4.5 s, **Memória**: 512 MB

Hold the Line - Megoldási ötlet

- Használjunk olyan merge sort tree-t, ahol minden csomópont egy **rendezett halmazt** (`set<int>`) tárol.
- **Frissítés:** a h_i értéket minden olyan csomópont halmazába beszúrjuk, amely lefedi az i-edik indexet. $\Rightarrow O(\log^2 N)$
- **Lekérdezés:** az (l, r) intervallumot a szokásos módon bejárjuk, és minden releváns halmazban megkeressük a H-hoz legközelebbi elemet a `lower_bound` segítségével.
  $\Rightarrow O(\log^2 N)$
- A válasz az összes vizsgált halmaz legkisebb különbsége.

Komplexitás: $O(N + M \log^2 N)$ összesen.

Megjegyzés: Van $O((N + M) \log(N + M))$ komplexitású megoldás is, amit itt nem tárgyalunk.

Hold the Line - kód

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   const int INF = (int)1e9 + 10;
5   struct merge_sort_tree {
6       const int n;
7       vector<set<int>> tree;
8       merge_sort_tree(int n_) : n(1<<(32 - __builtin_clz(n_))), tree(n*2) {}
9
10      void update(int i, int c) {
11          for(i += n; i > 0; i >>= 1) tree[i].insert(c);
12      }
13
14      int query(const set<int>& s, int x) {
15          int res = INF;
16          auto it = s.lower_bound(x);
17          if(it != s.end()) res = min(res, *it - x);
18          if(it != s.begin()) res = min(res, x - *prev(it));
19          return res;
20      }
21
22      int query(int l, int r, int h) {
23          int res = INF;
24          for(l += n, r += n; l < r; l >>= 1, r >>= 1) {
25              if(l&1) res = min(res, query(tree[l++], h));
26              if(r&1) res = min(res, query(tree[--r], h));
27          }
28          return res;
29      }
30  };
```

Hold the Line - kód

```
1   void solve() {
2       int n, q;
3       cin>>n>>q;
4
5       merge_sort_tree tree(n);
6
7       while(q--) {
8           int t;
9           cin>>t;
10          if(t == 0) {
11              int x, h;
12              cin>>x>>h;
13              tree.update(x - 1, h);
14          } else {
15              int l, r, h;
16              cin>>l>>r>>h;
17              int ans = tree.query(l-1, r, h);
18              cout << (ans == INF ? -1 : ans) << '\n';
19          }
20      }
21  }
22
23  int main(){
24      ios_base::sync_with_stdio(false);
25      cin.tie(0);
26
27      int t;
28      cin>>t;
29      while(t--) {
30          solve();
31      }
32  }
```