Heaps 2

Wednesday, 16 June 2021 7:59 PM

① Running Median [Codeforces, Alpha Prep]

↳ you will be given an array / stream of numbers.

→ [2, 14, 9, 7, 5, 6]

→ [2] → 2

→ [2,14] → [2,2] → 1

→ [2,14,9] → [1,2,9] → 2

→ [2,14,9,7] → [1,2,7,9] → 2

→ [2,14,9,7,5] → [1,2,5,7,9] → 2

→ [2,14,9,7,5,6] → [1,2,5,6,7,9] → 3

vector → [2, 1, 2, 2, 4.]

median

what is a median?

middle most element in sorted array.

↳ even numbers: ⊕ ↳ mean number has to be returned.

Properties of median

→ middle most element

→ left side numbers are smaller, and right side numbers are bigger.

Brute Force

① Form an array from the nos and sort them and return median

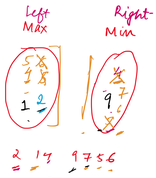
TC → $O(N^2 \log N)$

② Insertion sort ✓ we can sort in $O(N)$

TC → $O(N^2)$ ✓

Left Max

Right Min



2 14 9 7 5 6

| left | right |
|---|---|
| max heap | min heap |

(x)

① left.push(x) ✓ $\boxed{N \log N}$

② right.push(left.top())

③ left.pop()

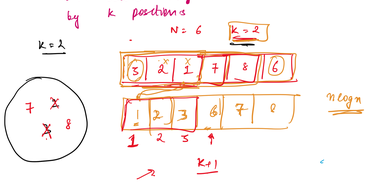
④ if (right.size() > left.size()), left.push(right.top()) right.pop()

⑤ return left.top();

② k apart sorted array → sort the array

(*) The element which will be present at $i^{th}$ position in the sorted array will at most be displaced by k positions

k = 2



N = 6 k = 2

n logn

① min heap

② Add k+1 elements to heap.

③ i = 0 j = k+1

while (j < N) {

arr[i++] = pq.top()

pq.pop()

pq.push(arr[j++])

}

while (i < N)

arr[i++] = pq.top()

pq.pop()

③ k Max pair combinations

A = { 1, 4, 2, 3 }

B = { 2, 5, 1, 6 }

k max pair sums

4+6 = 10 → 1

4+5 = 9 → 2

6+3 = 9 → 3

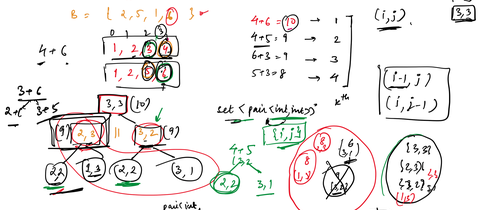
5+3 = 8 → 4

$k^{th}$

(i, j)

(i-1, j)

(i, j-1)



priority_queue < pair<int, pair<int, int>> > pq;

set < pair<int, int> > visited;

sort (A, A+n)

sort (B, B+m)

pq.push({A[n-1]+B[m-1], {n-1, m-1}})

visited.insert({n-1, m-1});

for (int a = 0; a < k; a++) {

auto x = pq.top();

pq.pop()

ans.push_back(x.first);

int i = x.second.first;

int j = x.second.second;

if (visited.find({i-1, j}) != visited.end()) {

visited.insert({i-1, j});

pq.push({A[i-1] + B[j], {i-1, j}})

}

// do same for (i, j-1)

}

return ans

}

$O(n \log n + m \log m + k(\log(k)))$

k

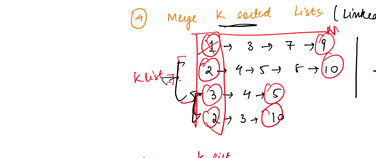
SC → O(k)

vector<int> ans;

TC →

SC →

④ Merge k sorted Lists (Linked lists)



vector < Node* > arr

k N

N* k

N 2N 3N 4N

: $N^2$

k lists

O(N)

O(2N)

O(3N)

O(4N)

: O(kN)

k list

$k^2 N$

$\log k$

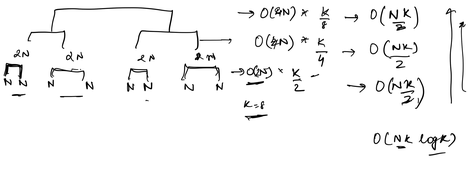
$N(k/2)$

O(N)

$O\left(\frac{Nk}{2} \log k\right) + 2N\left(\frac{k}{4}\right) + 4N\left(\frac{k}{8}\right) - - -$

$\frac{Nk}{2}$

int, Node*



→ O(2N) × k/8 → O(Nk/2)

O(4N) × k/4 → O(Nk/2)

→ O(8N) × k/2 → O(Nk/2)

k = 8

O(Nk log k)

ListNode* mergeK( vector < ListNode* > A) {

→ ListNode* dummy = new ListNode(0); // head = dummy

priority_queue < Node*, vector< Node*>, compare > pq;

for (auto x: A) pq.push(x);

→ while (!pq.empty) {

auto x = pq.top() pq.pop()

dummy → next = x;

dummy = dummy → next;

if (x → next) pq.push(x → next)

}

return head → next;

struct compare {

bool operator() ( ListNode* A, ListNode* B ) {

return A → val ≥ B → val;

}

}

(*) →

O(Nk log k)

O(k)

a a a b c

ab

k = 1

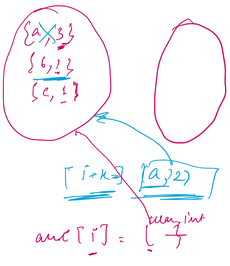
a b a c a

for (int i = 0; i < n; i++)

i+k

if( arr[i] = " " )

pq.top()



[i+k] [a, 2]

arr[i] = (char, int)