Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

по «Алгоритмам и структурам данных»

Структуры данных

Выполнил:

Студент группы P3233

Нгуен Нгок Дык

Преподаватели:

Косяков М.С.

Санкт-Петербург

2022

Code: https://github.com/ndwannafly/ITMO_ALGO

Задача № **I. Машинки**

Task reformulation:

- There are N numbers
- A set can contain maximum K numbers
- Initially, set is empty
- Q queries, each query is a number.
- If set has already contained this number, continue next query
- If set doesn't contain this number.

  + If size of the set exceeds K, then remove any number from the set and insert this number into the set

  + If size of the set doesn't exceed K, insert this number into the set

Constraint:

- $1 \leq K, N \leq 100\ 000$
- $1 \leq P \leq 500\ 000$

Keyword: Greedy, Set

Solution:

- Use a set to store all the numbers
- If set has already contained this number, continue next query
- If set doesn't contain this number

  + If size of the set doesn't exceed K, insert this number into the set

  + If size of the set exceeds K, remove the number from the set that the closest of its next occurrence to the right is furthest.
- L[i] - closest of i's next occurrence to the left
- R[i] - closest of i's next occurrence to the right
- The set contains pairs of R[i] and i , sorted by ascending order of R[i].
- Update the set when remove or insert new number into the set . O(logk)

Complexity:

- Operations: O(plogk).

- Space: O(n)

Задача № **J. Гоблины и очереди**

Task reformulation:

- Maintain a queue
- N requests, each request has 3 types:

  + i : push i to the end of the queue

  * i : push i to the middle of the queue

  - : remove the begin of the queue. Print out the id of it

Constraint:

- $N \leq 100\ 000$

Keyword: maintain 2 Deque

Solution:

- Left deque manages from begin -> mid
- Right deque manages from last -> mid + 1
- Left size always greater than Right size
- + request : push back to Right
- - request : print out front of Left then pop it out
- * request : push to front of the Right
- If Left deque has less elements than Right deque, remove Left back and push it to Right front. O(1)

Complexity:

- Operations: O(n)
- Spaces: O(n)

Задача № **K. Менержер памяти-1**

Task reformulation:

- An array of N consecutive memory cell, numbered from 1 to N.
- M requests which are allocated request or deallocated request
- Allocate X : Allocate a block size X which the cell before the first cell of this block is not free.
- Deallocate X: Deallocate x-th request

Constraint:

- $1 \leq N \leq 2^{31} - 1$
- $1 \leq M \leq 10^5$

Keyword: set or heap, implementation.

Solution:

- Maintain a set or a heap stores all the free segments (size and starting position), sort them by ascending order of their sizes
- Each allocated request:

  + Get the longest free segment from the set.

  + Allocate and narrow the free segment.

- Each deallocated request:

  There are 4 cases:

  + There is **no free segment** nearby to the **left** and also **no free segment** nearby to the **right**

  + There **is** free segment nearby to the **left** and also free segment nearby to the **right**

  + There is **only** free segment nearby to the **left**

  + There is **only** free segment nearby to the **right**

  In each case, update new segment and insert into the set.

Note:

- This task is more about implementation. Also demand good skill at debugging.
- I maintain two tracing arrays: trace_left[], trace_right[] in order to trace the free segment when we know the start or the end of the segment.

Complexity:

- **Operations** : O(M * log N)

- **Spaces** : O(M)

Задача № **L. Минимум на отрезке**

Task reformulation:

- We have an array of N elements and number K.
- Start from 1 move to the end of the array.
- Find the minimum element of each segment length K.

Constraint:

- $1 \leq N \leq 150\ 000$
- $1 \leq K \leq 10000$
- $K \leq N$

Keyword: Segment Tree or RMQ

My comment:

- Classical problem.
- Finding minimum of range, we can solve by Segment Tree or RMQ (Range Minimum Query).
- My solution is Segment Tree because I prefer it.
- RMQ is more optimal but ok let's back to it if we learn about LCA later.

Solution:

- https://en.wikipedia.org/wiki/Segment_tree
- Nothing special if we understand the segment tree.
- Update the node to the segment tree O(log n)
- Iterate through the array and get the minimum of each segment O(log n).

Note:

- There are more interesting techniques about Segment Tree for example:
- Segment Tree Lazy Propagation
- Segment Tree Persistent
- Binary Search in Segment tree

Complexity:

- Operation**:** O(nlogn)
- Spacing**:** O(4*n)