b'

Google Interview Experience (Off-Campus)

- Difficulty Level :\nHard
- Last Updated :\n04 Jun, 2019

LinkedIn resume filtering:\xc2\xa0I received a mail from the recruiter of Google asking about my resume. After a few days, he called me to ask for some details. He started asking about my profile, my language of preference for coding, etc. He also asked me whether I held any job offers at that moment. Later on, he started asking me general technical question such as:

1. Runtime complexity of sort/search algorithms
2. Hashing and tree conditional complexities
3. Balanced tree examples
4. Estimate 2^24 without a calculator

Telephonic Rounds at Hangouts:

The interviewer greeted me and asked me a stereotypical question, **Tell me about yourself.**

The main question was: Given candidates standing for an election, you have to create an interface which contains the following function:

1. voteCandidate (candidateID);

2. getTopK (k);

The first function is used to cast a vote to the candidate represented by the candidateID.

The second function is used to find the top K candidates at any moment during the course of an election.

Solution: Use maps to store the count of votes of each candidate and max heap to retrieve top k elements. Time complexity O(n + klogn).

Refer to this link for the second function:

After the telephonic interview, I was invited for onsite interview. I had 4 rounds.

Round 1:

Q: A graph has N vertices numbered from 1 to N. We have two lists. One list M consisted of edges between vertices. The other list K consists of restricted paths. We have to add edges one by one from M and check whether the addition of the particular edge leads to a path between the restricted edges given in K. If it creates a path, we have to discard the edge.

Example: N = 4; K = {(1, 4)}; M = {(1, 2), (2, 3), (3, 4)}. Here, addition of edge (3, 4) will create a path between 1 and 4. Hence we discard edge (3, 4)

A: Use connected components.

Round 2:

Q1. We have an interface named Logger which contains two functions:

startReq(req Id, start time)

finishReq(req Id, end time)

Logger gets a large number of requests with the start time. These requests are sent to startReq function. After the request finishes, we invoke the function finishReq. We should be able to print the output containing the finished requests sorted by start time.

\xc2\xa0

Request ID	Start Time	End Time
A	0	25
B	4	18
C	2	20
D	7	10

We need output as:

A 0 25

C 2 20

B 4 18

D 7 10

A: Use map to store the request end time. Use the queue for keeping the requests sorted.

Q2. Given various subsequences of an array, print the overall array:

Example: [1, 3, 5], [1, 3, 9], [9, 5]

Array : [1, 3, 9, 5]

A: Use DAG to represent the subsequences. Perform topological sort to get the array.

Round 3:

Q1: Implement the version control map system which takes the snapshot of the versions of data. Implement the following functions:

put(key, value) -> puts the value again the key in the latest version of the map

get(key) -> get the value of the key for the latest version of the data

snapshot() -> take a snapshot and increment the version

getValVersion(version id, key) -> return value of the key of the particular version

A: I made use of maps of vectors like this map<int, vector<pair<int, int>> versionMap;

Q2: Given a stream of integers, a value k and a value w, consider the integers in the window w and chop off greater k and smaller k elements from the window w. From the remaining elements, compute the average.

A: make use of Min and Max heaps.

Round 4:

Q1: Whether a string A can be transformed to a string B

Echo {1, 2, 3}{a, b} convert into

1a, 1b, 2a, 2b, 3a, 3b

PS: there can be any number of curly braces.

A: Make Use of queue to get the resultant strings.

My Personal Notes\n*arrow_drop_up*

Add your personal notes her

Save

'