Tech Interview Prep

Lecture 6

Christian Yongwhan Lim

Thursday, September 21, 2023

Overview

- Q&A
- Roll Call
- TechPrep AI Solution Ideas
- Greedy Algorithm and Invariant
- Graph
- Important Reminders

Q&A

- Is there a coding question that you found very hard and tried very hard to understand the solution, but eventually ended up memorising the question and the solution?

- Why did you choose to work as a software engineer?

- Do you need PhD to be a great engineer?

Q&A

- How do you manage your time and flow when you are solving an interview problem?

- How do you best prepare for behavior interview? Can we go through some classic questions?

- Do quant jobs (ex: Jane Street, Hudson River Trading, Citadel, ...) ask the same questions as MAANG (Meta, Apple, Amazon, Netflix, and Google)?

Q&A

- Why do you choose to be an adjunct faculty?

- Will this course talk about things that we need to do beyond problem-solving to succeed in a tech interview?

- What is the difference between the interview process for an internship and for an early industry hire with just a few years of experience?

Roll Call

- While waiting for roll call to finish, use https://bit.ly/2023-naq to sign up for **2023 ICPC North America Qualifier**.
- You do not need any prior background to compete! The deadline is coming up soon!

- **September 30** from **2pm ET** to 7pm ET, at most likely **Uris Hall**.

TechPrep AI Solution Ideas

- Valid Anagram
- Distinct Adjacent
- K Collinear Line

Valid Anagram

-

Valid Anagram

- sort

Distinct Adjacent

-

Distinct Adjacent

- dynamic programming

K-Collinear Line

-

K-Collinear Line

- (triple) for loops

Greedy Algorithm and Invariant

- "A greedy algorithm is an algorithm that computes a solution in steps; at each step the algorithm makes a decision that is locally optimum, and it never changes that decision."

Greedy Algorithm: Question #1 out of 2

- Balanced strings are those that have an equal quantity of 'L' and 'R' characters.
- Given a **balanced** string s (so, to be explicit, it only contains 'L' and 'R' only), split it in the maximum amount of balanced strings.
- Return the maximum amount of split balanced strings.

Greedy Algorithm: Answer #1 out of 2

- Time: O(n)
- Space: O(1)

```
int balancedStringSplit(string s) {
    int ret=0, cur=0;
    for (char ch : s) {
        if(ch=='R') cur++;
        else cur--;
        if(!cur) ret++;
    }
    return ret;
}
```

Greedy Algorithm: Question #2 out of 2

- A string S of lowercase English letters is given.
- We want to partition this string into as many parts as possible so that each letter appears in at most one part, and return a list of integers representing the size of these parts.

- Time: O(n log n)
- Space: O(1)

Greedy Algorithm: Answer #2 out of 2

```
vector<int> partitionLabels(string S) {
    vector<int> ret;
    int prev=-1, mx=0, n=S.size();
    map<char,int> mp;
    for (int i=0; i<n; i++)
        mp[S[i]]=i;
    for (int i=0; i<n; i++) {
        mx=max(mp[S[i]],mx);
        if(i==mx) ret.push_back(i-prev), prev=i;
    }
    return ret;
}
```

Graph

- "Informally, a graph is a set of vertices and edges connecting them. Formally, a directed graph is a set V of vertices and set E of edges."
- **Undirected graph** is a case where edge is bidirectional.
- **Directed acyclic graph** (DAG) is where edge is directional but the graph does not contain a cycle.
- Adjacency list (Sparse) vs Adjacency matrix (Dense)
- **Tree** is a connected graph without cycle.
- **Forest** is a graph without cycle.

Graph: Question #1 out of 3

- In a town, there are N people labelled from 1 to N.
- There is a rumor that one of these people is secretly the town judge.
- If the town judge exists, then:
  - The town judge trusts nobody.
  - Everybody (except for the town judge) trusts the town judge.
  - There is exactly one person that satisfies properties 1 and 2.
- You are given trust, an array of pairs trust[i] = [a, b] representing that the person labelled a trusts the person labelled b.
- If the town judge exists and can be identified, return the label of the town judge. Otherwise, return -1.

- Time: $O(N^2)$
- Space: $O(N^2)$

Graph: Answer #1 out of 3

```
class Solution {
public:
    int findJudge(int N, vector<vector<int>>& v) {
        vector<vector<bool>>
trust(N,vector<bool>(N,false));
        for (auto x : v)
            trust[x[0]-1][x[1]-1]=true;
        // …
    }
};
```

- Time: $O(N^2)$
- Space: $O(N^2)$

Graph: Answer #1 out of 3

```
map<int,int> ct;
for (int i=0; i<N; i++) {
    int cur=0;
    for (int j=0; j<N; j++) {
        if(i==j) continue;
        if(trust[j][i]) cur++;
    }
    if(cur==N-1) ct[i]++;
}
```

Graph: Answer #1 out of 3

- Time: $O(N^2)$
- Space: $O(N^2)$

```
for (int i=0; i<N; i++) {
    int cur=0;
    for (int j=0; j<N; j++) {
        if(i==j) continue;
        if(!trust[i][j]) cur++;
    }
    if(cur==N-1) ct[i]++;
}
```

Graph: Answer #1 out of 3

- Time: $O(N^2)$
- Space: $O(N^2)$

```
vector<int> ret;
for (int i=0; i<N; i++)
    if(ct[i]==2)
        ret.push_back(i+1);
if(ret.size()!=1)
    return -1;
return ret[0];
```

Graph: Question #2 out of 3

- You are given a map of a server center, represented as a n * m integer matrix grid, where 1 means that on that cell there is a server and 0 means that it is not a server.
- Two servers are said to communicate if they are on the same row or on the same column.
- Return the number of servers that communicate with any other server.

- Time: O(nm)
- Space: O(n+m)

Graph: Answer #2 out of 3

```
class Solution {
public:
    int countServers(vector<vector<int>>& grid) {
        int n=grid.size(), m=grid[0].size();
        map<int,int> row,col;
        for (int i=0; i<n; i++)
            for (int j=0; j<m; j++)
                if(grid[i][j])
                    row[i]++, col[j]++;
        // …
    }
};
```

- Time: O(nm)
- Space: O(n+m)

Graph: Answer #2 out of 3

```
int ret=0;
for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++) {
        if(!grid[i][j]) continue;
        if(row[i]>1||col[j]>1) ret++;
    }
}
return ret;
```

Graph: Question #3 out of 3

- Given an undirected graph, return true if and only if it is **bipartite**.
- Recall that a graph is bipartite if we can split its set of nodes into two independent subsets A and B, such that every edge in the graph has one node in A and another node in B.
- The graph is given in the following form: graph[i] is a list of indexes j for which the edge between nodes i and j exists.
- Each node is an integer between 0 and graph.length - 1.
- There are no self edges or parallel edges: graph[i] does not contain i, and it doesn't contain any element twice.

Graph: Answer #3 out of 3

- Time: O(N)
- Space: O(N)

```
bool isBipartite(vector<vector<int>>& graph) {
    vector<int> col(graph.size(),-1);
    for (int i=0; i<graph.size(); i++) {
        if(col[i]!=-1) continue;
        // …
    }
    return true;
}
```

- Time: O(N)
- Space: O(N)

Graph: Answer #3 out of 3

```
queue<int> q; q.push(i); col[i]=0;
while(!q.empty()) {
    int cur=q.front(); q.pop();
    for (const auto &nxt : graph[cur]) {
        if(col[nxt]==-1)
            col[nxt]=1-col[cur], q.push(nxt);
        if(col[cur]==col[nxt]) return false;
    }
}
```

Module II Format

- A **problem link** will be shared a day in advance.
- Each student will give a **presentation** in **10~15 minute** on their idea and solution.
- Only a **full, executable solution** is acceptable.

Module II Format

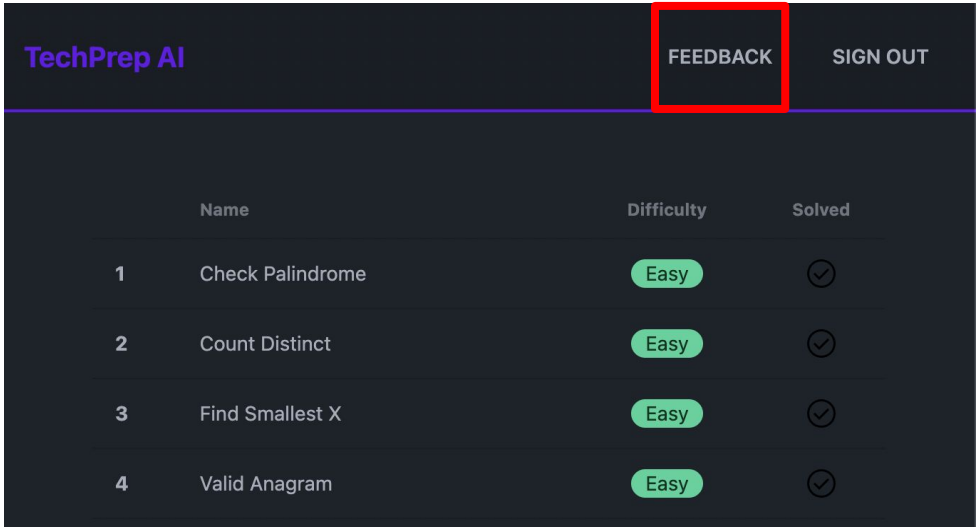
- As each student is presenting, a **real-time feedback** would be rendered, as needed.

- We will finish by presenting a **model solution**.

- Students other than the presenter should listen to the presentation and provide a score with a brief justification in a **Google form** that I share at the start of the lecture.

Important Reminders

- Make sure to do **LeetCode Weekly** and **Biweekly** on **<u>Saturday</u>**!
- For TechPrep AI Daily Challenge, please submit your feedback using:




THANK YOU