# Cox Inc

* We have a class that models navigation items for our client websites. There is a root level item which is not displayed, but contains a list of child items. These child items may have child items, which may also have child items, to an arbitrary level of depth.

Write a method in C# or Java that takes in the root level NavigationItem and prints out the navigation structure in a format that a non-developer could understand and use to help them troubleshoot issues with a site. You can assume that all Lists of children are non-null and have length of zero when they are empty.

public class NavigationItem {public string Url;public string Label;public List<NavigationItem> Children;}

* Web APi
  + Status code

# John Dear

* 1. Tree traversal
  2. abstract class and interface
  3. size of an integer and a double
  4. concept of inheritance
  5. DSA and OOPS and some OS topics
  6. What is polymorphism, inheritance, abstraction.
  7. Tell us more about your projects.
  8. Start pattern questions
  9. Coding and DB questions
  10. A hard level leetcode DP problem, something like coin change but more complicated
  11. what is structure
  12. making range replacements on a string.- Not deer
  13. Find common sub strings (longest) between two string- not dear

## 1.1 Behavior

# Leetcode

1. basic tricks such as two-pointers and bit manipulation from CTCI or similar books.
2. For example, the number of ways for a task indicates DP, string transformation with dictionary indicates BFS / DFS / Trie, looking for duplicate or unique elements indicates hashing or bit manipulation, parsing indicates the use of stack
3. <https://medium.com/leetcode-patterns/leetcode-pattern-1-bfs-dfs-25-of-the-problems-part-1-519450a84353>
4. <https://medium.com/leetcode-patterns/leetcode-pattern-2-dfs-bfs-25-of-the-problems-part-2-a5b269597f52>
5. From medium leet code question number

Array

For array problems, math will play an important role here. For simple questions, basic math and data structure can solve the problems.

**Subcategories:**

sum of K numbers of elements in the list = Target, return either the index or the elements(might need to avoid repetition). (2/3/4 sums)

Partition a list into K equal part. (DP)

Maximum subarray 718,

Duplicates 217, 26, 27, 219, 287, 442

**Two pointers:** If you see in the problem that you can do comparison and it is always one type of satisfactory element is in ahead of the other, this could be resolved by two pointers(t1). Or one starts from the beginning, one from the end and going to the middle(t2). e.g. 122. Best Time to Buy and Sell Stock II (multiple transaction, accumulate sectional profit), 26, 27, 283. Move Zeroes, 121. Best Time to Buy and Sell Stock (keep track of min\_price and max\_profit, single pass, one transaction), 88. Merge Sorted Array, 167. Two Sum II — Input array is sorted(t2), 11. Container With Most Water (t2, move with greedy programming)

**Hashset:** 268. Missing Number

**DP:**39. Combination Sum, 416. Partition Equal Subset Sum, 698. Partition to K Equal Sum Subsets

**LCS:**718. Maximum Length of Repeated Subarray

# Aveva

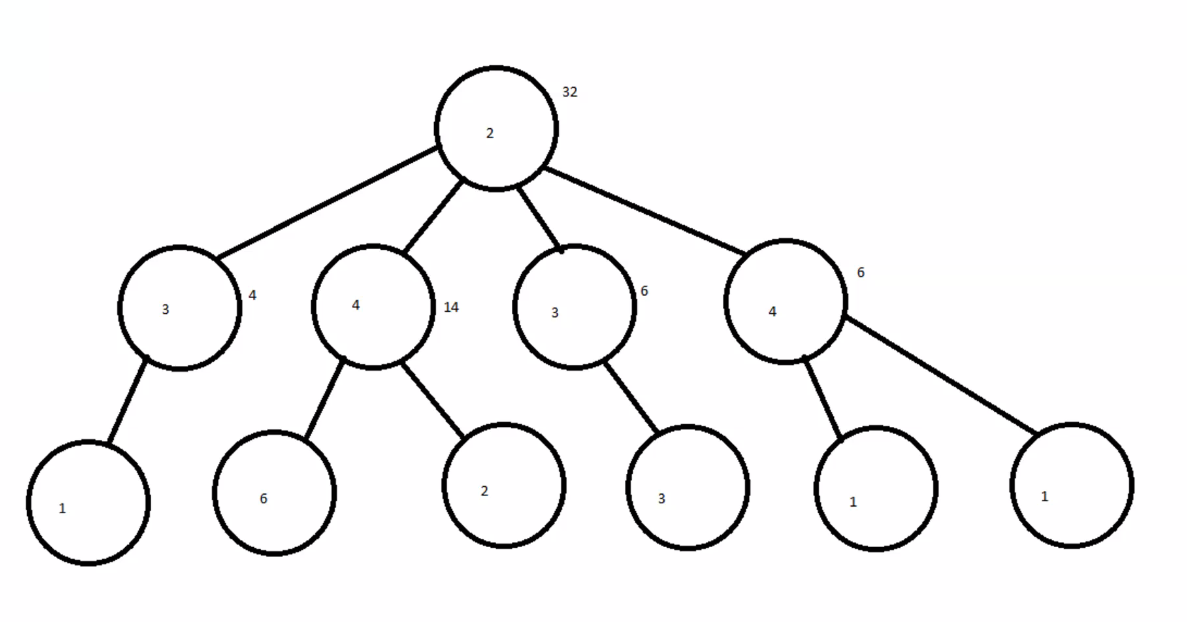
* C#
  + Delegate & Event
    - Who is dependent on what – like Delegate is dependent on event or event?
    - How to invoke Delegate
    - What exactly event is and how it works
  + Async & await
  + Design pattern – repository & singleton
  + What is dependency injection
  + Inversion of control
    - Transient
    - Scoped
    - Singleton
    - What other tool u have used for inversion control?
* WEB API
  + **CORS**
    - Accessibility
    - Why this is needed
    - Who control it?
  + **HTTP status code –** what all code u sends
    - 400 – client
    - 500
    - 200
  + WEB API **Versioning**
    - Disadvantage of url versioning
  + ***Authentication***
  + Authorization
  + Protect web API
    - Bear token – claim
  + Resiliency –
    - retry
  + Partition tolerant
  + How to roll back across microservice – like TP sent to SP and SP could not make then how.
* Web site faster
  + Bundling
  + Minification
  + CDN service
  + Paging
  + Resolution
  + Ajax calling
* Cloud
  + Access Azure blob
  + Docker & container
  + Kubernetes AKS
* Data Structure
  + Binary Tree
  + Graph
  + Sorted Data structure

# Wipro

* Dependency injection
* AWS
  + lambda uses
    - How you connect to AWS SQS or AWS service
      * Setup profile and use
      * Use AWS Access Key and Secrete key
  + FAN out and FAN IN
  + How token generated and used
  + Application lifecycle – Step by Step process
* .NET Core and .NET 5/6 difference
* How you protect your web api
  + CORS
* Serverless Application

# Charles Swab – 07 April 2022

* Sum of problem



* How you develop web api and deploy – tell me who process
* Docker container – container repository
* Dependency injection

# 6 Capgimini

* SQL
  + Table Valued function
  + View vs temp table
  + Procesure set
  + Scope Identity Vs 22identity vs
* C#
  + Dependency injection
  + Routing and where you right
  + EBvroment variable where you right
* Web Api
  + Authentication
  + CORS
* Angular
  + How u debug angular application

# CCC intelligent

* Integer >> Now power of 4
* Integer >> Send {{}} curly braces
* ECS cluster – All microservice on same cluster

# Wayfair from karate

1. -1-Find-Contiguous-URL-History
2. <https://leetcode.com/discuss/interview-question/1257969/wayfair-karat-interview-20-mins-scenario-based-system-design>
3. Find-Purchased-Users-data

Update:  
Leetcode has taken down the original coding questions posts that I had added. :(  
Adding the questions here in the comments.

## Find Purchased Users data:

import [java.io](http://java.io/).;  
import java.util.;

/\*  
The people who buy ads on our network don't have enough data about how ads are working for their business. They've asked us to find out which ads produce the most purchases on their website.

Our client provided us with a list of user IDs of customers who bought something on a landing page after clicking one of their ads:

Each user completed 1 purchase.  
completed\_purchase\_user\_ids = [  
"3123122444","234111110", "8321125440", "99911063"]

And our ops team provided us with some raw log data from our ad server showing every time a user clicked on one of our ads:

ad\_clicks = [  
#"IP\_Address,Time,Ad\_Text",  
"122.121.0.1,2016-11-03 11:41:19,Buy wool coats for your pets",  
"96.3.199.11,2016-10-15 20:18:31,2017 Pet Mittens",  
"122.121.0.250,2016-11-01 06:13:13,The Best Hollywood Coats",  
"82.1.106.8,2016-11-12 23:05:14,Buy wool coats for your pets",  
"92.130.6.144,2017-01-01 03:18:55,Buy wool coats for your pets",  
"122.121.0.155,2017-01-01 03:18:55,Buy wool coats for your pets",  
"92.130.6.145,2017-01-01 03:18:55,2017 Pet Mittens",  
]

//2017 Pet Mittens [3123122444, 96.3.199.11]

The client also sent over the IP addresses of all their users.

all\_user\_ips = [  
#"User\_ID,IP\_Address",  
"2339985511,122.121.0.155",  
"234111110,122.121.0.1",  
"3123122444,92.130.6.145",  
"39471289472,2001:0db8:ac10:fe01:0000:0000:0000:0000",  
"8321125440,82.1.106.8",  
"99911063,92.130.6.144"  
]

Write a function to parse this data, determine how many times each ad was clicked, then return the ad text, that ad's number of clicks, and how many of those ad clicks were from users who made a purchase.

Expected output:

1 of 2 2017 Pet Mittens  
0 of 1 The Best Hollywood Coats  
3 of 4 Buy wool coats for your pets

purchases: number of purchase IDs  
clicks: number of ad clicks  
ips: number of IP addresses  
\*/  
public class WayFairInterview1 {

public static List findPurchasedUsersData(String[] completedPurchaseUsers, String[] adClicks,  
String[] allUserIps) {  
List result = new ArrayList();  
Map<String, List> map = new HashMap<String, List>();  
Map<String, String> ipUserMap = new HashMap<String, String>();  
Set usersWhoMadePurchases = new HashSet();  
for (String user : completedPurchaseUsers) {  
usersWhoMadePurchases.add(user);  
}

for (String ip : allUserIps) {

String[] splits = ip.split(",");

ipUserMap.put(splits[1], splits[0]);

}

for (String adClick : adClicks) {

String[] splits = adClick.split(",");

String iPAddress = splits[0];

String userID = ipUserMap.getOrDefault(iPAddress, "");

String adText = splits[2];

List<String> inner = map.getOrDefault(adText, new ArrayList<String>());

inner.add(userID);

map.put(adText, inner);

}

for (String key : map.keySet()) {

int totalClicks = map.get(key).size();

List<String> usersWhoClicked = map.get(key);

int purchasedCount = getPurchasedCount(usersWhoClicked, usersWhoMadePurchases);

String outputString = purchasedCount + " of " + totalClicks + " " + key;

result.add(outputString);

}

// System.out.println(map);

return result;

}

public static int getPurchasedCount(List usersWhoClicked, Set usersWhoMadePurchases) {  
int count = 0;  
for (String user : usersWhoClicked) {  
if (usersWhoMadePurchases.contains(user)) {  
count++;  
}  
}  
return count;  
}

public static void main(String[] argv) {  
String[] completedPurchaseUsers = { "3123122444", "234111110", "8321125440", "99911063" };

// "IP Address, timestamp, Ad text"

String[] adClicks = { "122.121.0.1,2016-11-03 11:41:19,Buy wool coats for your pets",

"96.3.199.11,2016-10-15 20:18:31,2017 Pet Mittens",

"122.121.0.250,2016-11-01 06:13:13,The Best Hollywood Coats",

"82.1.106.8,2016-11-12 23:05:14,Buy wool coats for your pets",

"92.130.6.144,2017-01-01 03:18:55,Buy wool coats for your pets",

"122.121.0.155,2017-01-01 03:18:55,Buy wool coats for your pets",

"92.130.6.145,2017-01-01 03:18:55,2017 Pet Mittens" };

// "User ID, IP address"

String[] allUserIps = { "2339985511,122.121.0.155", "234111110,122.121.0.1", "3123122444,92.130.6.145",

"39471289472,2001:0db8:ac10:fe01:0000:0000:0000:0000", "8321125440,82.1.106.8",

"99911063,92.130.6.144" };

System.out.println(findPurchasedUsersData(completedPurchaseUsers, adClicks, allUserIps));

}  
}

Kindly upvote if it helps! Good luck with your interview.

Update:  
Leetcode has taken down the original coding questions posts that I had added. :(  
Adding the questions here in the comments.

## Find Contiguous URL History:

* Write a function that takes two users' browsing histories as input and
* returns the longest contiguous sequence of URLs that appears in both.
* Sample input:
* user0 = ["/start", "/green", "/blue", "/pink", "/register", "/orange",
* "/one/two"] user1 = ["/start", "/pink", "/register", "/orange", "/red", "a"]
* user2 = ["a", "/one", "/two"] user3 = ["/pink", "/orange", "/yellow",
* "/plum", "/blue", "/tan", "/red", "/amber", "/HotRodPink", "/CornflowerBlue",
* "/LightGoldenRodYellow", "/BritishRacingGreen"] user4 = ["/pink", "/orange",
* "/amber", "/BritishRacingGreen", "/plum", "/blue", "/tan", "/red",
* "/lavender", "/HotRodPink", "/CornflowerBlue", "/LightGoldenRodYellow"] user5
* = ["a"] user6 = ["/pink","/orange","/six","/plum","/seven","/tan","/red",
* "/amber"]
* Sample output:
* findContiguousHistory(user0, user1) => ["/pink", "/register", "/orange"]
* findContiguousHistory(user0, user2) => [] (empty)
* findContiguousHistory(user0, user0) => ["/start", "/green", "/blue", "/pink",
* "/register", "/orange", "/one/two"] findContiguousHistory(user2, user1) =>
* ["a"] findContiguousHistory(user5, user2) => ["a"]
* findContiguousHistory(user3, user4) => ["/plum", "/blue", "/tan", "/red"]
* findContiguousHistory(user4, user3) => ["/plum", "/blue", "/tan", "/red"]
* findContiguousHistory(user3, user6) => ["/tan", "/red", "/amber"]
* n: length of the first user's browsing history m: length of the second user's
* browsing history

\*/

import java.util.\*;

public class FindContiguousURLHistory {

public static List findContiguousHistory(String user1[], String user2[]){  
List result=new ArrayList<>();  
if(user1.length==0 || user2.length==0){  
return result;  
}  
int dp[][]=new int[user1.length+1][user2.length+1];  
int max=Integer.MIN\_VALUE;  
int endIndex=-1;  
for(int i=user1.length-1;i>=0;i--){  
for(int j=user2.length-1;j>=0;j--){  
if(user1[i].equals(user2[j])){  
dp[i][j]=dp[i+1][j+1]+1;  
if(max<dp[i][j]){  
max=dp[i][j];  
endIndex=j;  
}  
break;  
}  
}  
}  
if(max==Integer.MIN\_VALUE){  
return result;  
}  
for(int i=endIndex;i<endIndex+max;i++){  
result.add(user2[i]);  
}

return result;

}

public static void main(String[] argv) {  
String[] user0 = { "/start", "/green", "/blue", "/pink", "/register", "/orange", "/one/two" };  
String[] user1 = { "/start", "/pink", "/register", "/orange", "/red", "a" };  
String[] user2 = { "a", "/one", "/two" };  
String[] user3 = { "/pink", "/orange", "/yellow", "/plum", "/blue", "/tan", "/red", "/amber", "/HotRodPink",  
"/CornflowerBlue", "/LightGoldenRodYellow", "/BritishRacingGreen" };  
String[] user4 = { "/pink", "/orange", "/amber", "/BritishRacingGreen", "/plum", "/blue", "/tan", "/red",  
"/lavender", "/HotRodPink", "/CornflowerBlue", "/LightGoldenRodYellow" };  
String[] user5 = { "a" };  
String[] user6 = { "/pink", "/orange", "/six", "/plum", "/seven", "/tan", "/red", "/amber" };

System.out.println(findContiguousHistory(user0, user1));

System.out.println(findContiguousHistory(user0, user2));

System.out.println(findContiguousHistory(user2, user1));

System.out.println(findContiguousHistory(user5, user2));

System.out.println(findContiguousHistory(user3, user4));

System.out.println(findContiguousHistory(user4, user3));

System.out.println(findContiguousHistory(user3, user6));

}

}

Kindly upvote if it helps! Good luck with your interview.

## How did you calculate throughput? I was asked same questions.

# Karate -Walmart

* Questions focus on fundamentals of data structures and algorithms.
* Focus more on Arrays, Strings, Hashmap, Lists for this round.
* Asked me LC medium question