CS 61B

Final Discussion Fun times :)

1 Design

Yay! You got an interview with your dream company (or maybe you just like solving fun problems)! For each of the following scenarios:
- Determine which **data structures** (doesn't have to be strictly Java) + **algorithms** give the best performance
- The worst-case runtime for any operations listed.

(a) Chalisee says she has a list of N names of crops, where each entry in the list represents an acre of farmland in the Central Valley. Find the number of acres grown for each crop.

(b) For the scenario above say we want to query if a given crop is in the database of N crops. Optimize for both constructing the solution and matching a query

(c) Pearbnb is a trusted community marketplace for people to list, discover, and order unique produce and plants around the world. Chalisee wants to start developing auto-complete for search on Pearbnb's website. When a user types in the first K characters of a query, she wants the website to say how many products have the same K character prefix. Assume that no products have a name longer than M and there are N distinct products. Optimize for both constructing the solution and matching a query.

(d) One of the things that Pearbnb does is optimize the profits for farmers. Pearbnb uses a database of N `Order`s. Each `Order` represents an order from a customer for a specific product and has the following: the customer's name, the `Date` the order was made, the `Date` requested for the delivery, the name of the product ordered, the quantity of the product ordered, and the price per unit for the product. Chalisee, a champion for Big Data, wants to run analytics on Pearbnb's database and query for `Order`s requested to be delivered within a certain range of dates for a certain product. Optimize for both constructing the solution and matching a query.

(e) Pearbnb runs a subsidiary company, ImPearfect Produce, that handles it's deliveries to customers in urban areas. ImPearfect Produce likes to optimize its deliveries and also promote fairness, but it only allows each of its trucks to carry one type of product at a time. Therefore, ImPearfect Produce has the policy to send a truck carrying the product of the earliest uncompleted order, while trying to fulfill as many orders as possible for that product (also in the order of earliest uncompleted order). ImPearfect Produce must maintain some collection of N `Orders` that optimizes adding new orders and figuring out what products to deliver on its next truck.

2 Reductions

Taken from Spring 2017 Final.

Main idea: Problem P reduces to problem Q if we can use Q as a "black box" algorithm to solve P. Formally, if any subroutine for task Q can be used to solve P, we say P reduces to Q.

(a) Describe an algorithm to find a **maximum** spanning tree. Your algorithm must use Kruskal's as a block box without any modifications.

(b) Suppose you want to find the SPT of a graph, but where you redefine the total cost of a path as follows. Let `cost(List<Edge>)` be the sum of the weights of the edges, plus the number of edges. In other words, we want to run Dijkstra's taking into account not just the weights of the edges, but also the number of edges. Describe an algorithm to find this shortest paths tree. Your algorithm must use Dijkstra's as a "black box". Your answer should be brief.

(c) Can the problem of finding shortest paths on a graph with a negative edge weight be reduced to Dijkstra's?

3 Now for the real fun

(a) Find the n^{th} last element in a singly linked list. (n = 1 means find the last element)

(b) An animal shelter, which holds only dogs and cats, operates on a strictly "first in, first out" basis. People must adopt either the "oldest" (based on arrival time) of all animals at the shelter, or they can select whether they would prefer a dog or a cat (and will receive the oldest animal of that type). They cannot select which specific animal they would like. Create the data structures to maintain this system and implement the operations: `enqueue`, `dequeueAny`, `dequeueDog`, and `dequeueCat`.

(c) You have a large array with most of the elements as zero. Use a more space-efficient data structure, SparseArray, that implements the same interface:

- `init(arr, size)`: initialize with the original large array and size.
- `set(i, val)`: updates index at at i with val
- `get(i)`: gets the value at index i

(d) Design an efficient data structure that finds the median from a stream of numbers. In particular, the data structure implements the following interface:

- `void addNum(int num)`: Add an integer number from the data stream to the data structure.
- `double findMedian()`: Return the median of all elements so far.