**BST**

Explain in your own words why BST is more efficient O(log n) time for insertion/search tasks than linear data structures (arrays/lists) that take O(n) time.

**QuickSort vs MergeSort**

Explain why quicksort is O(n^2), while mergesort is O(log n).

When can quicksort be an efficient over mergesort.

**MVC Sqlite**

You will be asked to implement the following MVC application.

You are provided with a csv file ‘TourneySeeds.csv’ - (taken from [kaggle.com](https://www.kaggle.com/c/march-machine-learning-mania-2017/data) data itself is not accessible without an account). This file has all the NCAA men’s basketball tournament participating teams from 1985-2017. Three columns are given, Year, seed information, team id. We will not be needing the last piece of data. Generally the yearly tournament has 64 = 2^6 teams divided into 4 brackets. Each bracket has 16 teams ranked from 1 to 16. The brackets are also ranked, so the 16th seed in the lowest bracket is considered the 64th seed. The tournament format consists of 4 rounds to decide the winner of each bracket, and then 2 more rounds to decide the champion. A few years ago, a 65th team was added, necessitating a preliminary game between the 64th and 6th seeds. Finally, more recently the tournament expanded to 68 teams, so the 16th seed plays a “play-in” game with the extra team to determine who will be the actual 16th seed of each bracket. This explains the format of the second column in the csv file, so for example in 2017 “Z11a” indicates region “Z” (they are always called, W, X, Y, Z), seed 11, team b (corresponding to team “Z11b” against which it will play the play-in game).

In your models, create two tables, one called ‘team’, the other called ‘game’. The ‘team’ table should have 4 columns: id, year, bracket, seed, shared. The latter 3 corresponding to the data in the second column of the file. The ‘game’ table will have at least 5 columns: id, team1, team2, round, result. Team1 and team2 columns should be foreign keys referencing ‘team’ table.

Create a seed file that seeds the ‘team’ table with the csv file.

The app should have the following functionality:

models.py that connects to the db

seed file with seed data from csv

views.py and controller.py that allow the adding and deleting of teams, and the ability to update the ‘game’ table with game results.

Added bonus – construct the functionality for the ‘games’ table in such a way that only realistic games can be added. i.e. a game in the ‘zeroth’ round can only take place between the appropriate “play-in” teams. First round games take place between, seed 1 an 16, seed 2 and 15 (within the bracket). Round 2 games take place between winners of round 1 games (highest remaining in bracket vs lowest etc.). A team can’t play twice in the same round, can’t skip a round and so forth.