Data Stuctures: Assignment 1

Josh Wainwright
UID:1079596

1. **Wikipedia Articles** — *Hash map.*
   - Hash key is article title, hash value is the article text etc.
   - Access times want to be fast for an article that is read more times than it is edited. For every access of an article, the title is used to locate the article which can be displayed, so the average and worst case times are equal, i.e. constant.
   - When the article is edited, or a new article is added, the average case will still be constant. The worst case is when the hash map must be increased in sized and so all the articles have to be rehased, in which case there will be a significant time increase.
2. **Driving Lessons** — *Tree set.*
   - Just need to individually store the name of each customer who has had a free lesson.
   - Set allows no duplicates, so a user cannot be added more than once. If implementation allows, could check the customer's status in the set at the same time as adding them.
     - When a customer tries to book a free lesson, try to add them to the set.
     - If this fails, then they already exist so cannot have a free lesson.
     - If it is successful, then they have their lesson, but if they try to book again, their name exists in the set.
   - When checking a name, the worst case is that they are not already in the set and so all elements have to be checked, i.e. linear, same for adding.
3. **Painting Prices** — *Tree map.*
   - Want the items to be sorted, so add them using the price as the key (making the assumption that no two pictures have the same price) and any other information as the value.
   - Insertion and deletion is $O(\log(n))$ for both worst and average cases, so access is fast. The items will be sorted by the key, so to get the most expensive items for a given price, the tree would be traversed to find the item less than and closest to that price, and then the next n items below it returned.
4. **Birth Cities** — *Hash set.*
   - The data would be stored simply as the name of that city in the set.
   - To check if a city already exists in the set, the hash of the new city would be taken and that position in that array checked. If it is occupied, it is reasonable to assume that that city has already been added to the set, and so it is not added again.
   - Adding to the set is simply a case of putting the new city into the array at the hash's position so will be constant. Searching for an item is also constant for both worst and average cases since a single position need only be checked for data.