- Why HashMap

HashMap are efficient for locating a value based on a key and inserting and deleting values based on a key.

Hashmap is good to store a collection of pairs. The entries are not ordered. For every entry strores there will be a hashed index. Searching by hashed index is efficient. Runtime efficiency for search is constant O(1). The Container splits its data to lot of "buckets" that contains only elements with same hashcode of key. Thus the main advantage of using HashMap is speed

Another advantage is t it permits null value

For search 1&2, to find receiver/dialler base on dialler’s number. A dialler and receiver number can make a Key, Value pair stored in HasMap.
Key can be dialler or receiver, value can be an array of numbers.

For search 4&5, to find max and min switch, we can store switch ID with its frequency as key value entry stored in HashMap. We can iterate through map to find max/min value, so as to find max/min connection switch.

- Why AVL Tree

AVL tree is a self-balancing binary search tree. AVL trees are intended for in-memory use, where random access is relatively cheap. It has efficient range search. For every insert and deletion, it will restructure itself to maintain balance.

The advantage of using AVL Tree is it maintains O(log n) in average and worst case. Another advantage is AVL tree is memory efficient, it does not reserve more than it needs.

For search 3, to find fault in records, every record must go through as there is no order when there will be a fault. However if there is a time slot, Callrecords can be sorted by timestamp. Thus searching records by timestamp is efficient with O(log n).

For search 6 with time condition, the ordering of timestamp can help quicken search. Furthermore we can implement a TreeMap. Make timestamp as key and CallRecord as value. The searching runtime efficiency is close to constant O(log n)