Given m lists each of average size n such that each list is sorted and there are no duplicates available in each list. Find the top k most frequently occurring elements in these lists.
Each list is sorted and there are no duplicates in a particular list.
[1, 2, 4, 6]
[2 , 5 , 6, 8, 10]
[1, 3, 11]
You can not use O(n*m) space;

Asked in round 1:

Manage resume version:

There are two function given:

```
Void update( string key, string value ) {
        //complete this
        // each time, a key and corresponding value will be given. If key is already present in resume, then update the value of key otherwise add key and corresponding value in resume.
        //each time we update something, version value get increased by 1.
}

map<string,string> get(int version) {
        //complete this
        // it is required to return all the latest update occurred till the given “version”.
}
```

Example:

Consider like initially resume is blank.

update(“name”, “alpha”) —> version 1

get(1) —-> [(“name”, “alpha”)]

Update (“name”, “suresh”) —->version 2

get(2) —---> [(“name”, “suresh”)]

update(“education”, “btech”) —----> version 3

get(3) —------> [(“name”, “suresh”), (“education”, “btech”)] // (returning all the latest update happens till now)

update(“cgpa”, “8”) —---->version 4

get(4) —---> [(“name”, “suresh”), (“education”, “btech”), (“cgpa”, “8”)]

get(2) —---> [(“name”, “suresh”)]

Approach:

map< string, vector<pair<int,string>>>mp;

1st string: to store key
Int: to store version number
2nd string: to store value
Each time, u get any update regarding any key, then push_back the value with version in the vector.
For get function:
Traverse each key of map, and find the version equal to or smaller than the required version (this can be done efficiently by using binary search on version as version will always be sorted) and then return the value correspondingly.