binarySearch

```cpp
//
// Created by yjs on 2022/4/21.
//

#include <bits/stdc++.h>

using namespace std;


class Search {
public:

    int binarySearch(const vector<int> nums, const int &target) {
        int left = 0, right = nums.size() - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid - 1;
            }
        }
        return -1;
    }


    int lowerSearch(const vector<int> nums, const int &target) {
        // [a,b)
        int left = 0, right = nums.size();
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                right = mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid;
            }
        }
        return left;
    }


    int upperSearch(const vector<int> nums, const int &target) {
        // (a,b]
        int left = 0, right = nums.size();
        while (left < right) {
            int mid = left + (right - left) / 2;
```

```
            if (nums[mid] == target) {
                left = mid + 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else if (nums[mid] > target) {
                right = mid;
            }
        }
        return left - 1;
    }


};


int main() {

    vector<int> nums{1, 25, 69, 25, 17, 17, 14};
    sort(nums.begin(), nums.end());
    for_each(nums.begin(), nums.end(), [](auto c) { cout << c << " "; });
    cout << endl;
    Search search1;

    cout << search1.binarySearch(nums, 17) << endl;
    cout << search1.lowerSearch(nums, 17) << endl;
    cout << search1.upperSearch(nums, 17) << endl;


    return 0;
}
```

KMP

```
#include <bits/stdc++.h>

using namespace std;



class KMP1{

private:
    string  pat;
    vector<int> next;
public:

    KMP1(const string & pat):pat(pat){
        // init next
        int j=0,k=-1;
        next.resize(pat.length());
        next[0]=-1;
        while (j<pat.length()-1){
```

```
            if(k==-1 || pat[j]==pat[k]){
                j++;
                k++;
                next[j]=k;
            }else{
                k=next[k];
            }


        } // end init next
    }


    int search(const string & txt ){
        int i=0,j=0;
        while (i<txt.length() && j<pat.length()){

            if(j==-1 || txt[i]==pat[j]){
                i++;
                j++;


            }else{
                j=next[j];
            }


        }
        if(j>=pat.length()){
            return i-pat.length();
        }else{
            return -1;
        }




    }




};


class KMP2{

private:
    string  pat;
    vector<int> arcnext;
public:

    KMP2(const string & pat):pat(pat){
        // init next
```

```
        int j=0,k=-1;
        arcnext.resize(pat.length());
        arcnext[0]=-1;
        while (j<pat.length()-1){

            if(k==-1 || pat[j]==pat[k]){
                j++;
                k++;
                if(pat[j]==pat[k]){
                    arcnext[j]=arcnext[k];
                }else{
                    arcnext[j]=k;
                }
            }else{
                k=arcnext[k];
            }
        } // end init next
    }


    int search(const string & txt ){
        int i=0,j=0;
        while (i<txt.length() && j<pat.length()){

            if(j==-1 || txt[i]==pat[j]){
                i++;
                j++;


            }else{
                j=arcnext[j];
            }


        }
        if(j>=pat.length()){
            return i-pat.length();
        }else{
            return -1;
        }




    }




};


int main(){
```

```
    string res{"abaabc"};
    KMP1 * kmp=new KMP1(res);
    int pos=kmp->search("abaabaabcab");
    cout <<pos<<endl;



    return 0;
}
```