NATIONAL INSTITUTE OF TECHNOLOGY DELHI

CSB-252

Design and Analysis of Algorithms Lab (CSB-252)

(Department of computer science and engineering)

Submitted by: Nishant Raj

Submitted to: (Prof.) Dr. Rishav

Roll no.: 211210042

Approved:

Lab 06: KNN Algorithm

In this assignment students will understand the implementation of KNN algorithm and predict the category of the unknown movie.

Write complete program and algorithm for KNN analyse best, worst and average case time complexity.

Following steps need to be accomplished for KNN algorithm:

1. Algorithm

2. Program (preferred language is C, but students are allowed to use any other

programming language. Furthermore, there should not be any in-built function)

3. Predict the category of a movie given by user. (e.g. Action – 50, comedy- 50, result-

Action/Comedy?).

4. Execution time needs to be printed on the console

5. Number of bytes used during the execution also needs to be printed

ALGORITHM

We are here finding distance using distance formula and corresponding type of movie and pushing back this set into a vector. We have initially given input of 4 movies and now we can judge the type of other movies.

1. We are judging minimum distance.
2. We are judging maximum distance.
3. We are judging distance from centroid as well.

And take majority!!!

For this we have created 2 classes movie and closeness.

Here is the code!

INPUT

```
#include<iostream>

#include<cmath>

#include<vector>

using namespace std;


class movie{

    public:
```

```
    int action;
    int comedy;
    char category;

    movie(int action,int comedy , char category){
        this-> action= action;
        this-> comedy= comedy;
        this-> category = category;
    }
};
class closeness{
    public:
    float distance;
    char category;
    closeness(int action , int comedy, movie mknown){
        distance = sqrt(pow(action - mknown.action ,2) + pow(comedy - mknown.comedy, 2));
        category =  mknown.category;
    }
};
int find_centroid(vector<movie> v , char m_type , char return_type){
    int count;
    pair<int, int> p;
    p.first=0;
    p.second =0;
    for(int i=0; i< v.size(); i++){
        if(v[i].category== m_type){
            p.first += v[i].action;
            p.second += v[i].comedy;
            count++;
        }
    }
    if(return_type=='x'){
        return p.first/ count;
    }
    else{
        return p.second/ count;
    }
}
int main(){
    vector<movie> movies;
    vector<closeness> distance;
    movie m1(100, 5, 'A');
    movie m2(20, 90, 'C');
    movie m3(95, 15, 'A');
    movie m4(25, 110, 'C');
    movies.push_back(m1);
    movies.push_back(m2);
```

```
movies.push_back(m3);
movies.push_back(m4);
int n=1;
do{
if(n==1){
    int action;
    int comedy;
    cout<<"Enter Action"<<endl;
    cin>> action;
     cout<<"Enter Comedy"<<endl;
    cin>> comedy;
    for (int i=0 ; i< movies.size(); i++){
        closeness c(action , comedy , movies[i]);
        distance.push_back(c);
    }
    int min= INT_MAX, max =INT_MIN; char cat_min , cat_max;
    for (int i=0; i< distance.size(); i++){
        if(distance[i].distance< min){
            min = distance[i].distance;
            cat_min = distance[i].category;
        }
        if(distance[i].distance> max){
            max = distance[i].distance;
            cat_max = distance[i].category;
        }
    }
    if (cat_min != cat_max){
        char category = cat_min;
        cout<<category<<endl;
        movie m(action, comedy , category);
        movies.push_back(m);
    }
    else{
        int centroid_c_x= find_centroid(movies , 'C', 'x');
        int centroid_c_y= find_centroid(movies , 'C', 'y');
        int centroid_a_x= find_centroid(movies , 'A', 'x');
        int centroid_a_y= find_centroid(movies , 'A', 'y');

        int distance1 = sqrt(pow(action - centroid_c_x ,2) + pow(comedy - centroid_c_x, 2));
        int distance2 = sqrt(pow(action - centroid_a_x ,2) + pow(comedy - centroid_a_x, 2));
        if(distance1< distance2){
            cout<< "C"<<endl;
            movie m(action , comedy, 'C');
            movies.push_back(m);
        }
        else{
            cout<< "A"<<endl;
```

```
            movie m(action , comedy, 'A');
            movies.push_back(m);
          }
        }
        cout<<"DO YOU WANT TO ADD MORE DATA (1/0)"<<endl;
        cin>> n;
    }
    else
    return 0;
    }while(n==1);
    return 0;
}
```

OUTPUT

```
 PS C:\Users\User\OneDrive\Desktop\Sem-4 Material\Algorithms> cd "c:\Users\User\OneDrive\Desktop\Sem-4 Material\Algorithms\" ; if ($?) { g++ knn.cpp -o knn } ; if ($?) { .\knn }
Enter Action
70
Enter Comedy
30
A
DO YOU WANT TO ADD MORE DATA (1/0)
1
Enter Action
30
Enter Comedy
70
C
DO YOU WANT TO ADD MORE DATA (1/0)
1
Enter Action
50
Enter Comedy
50
C
DO YOU WANT TO ADD MORE DATA (1/0)
0
```

- Time Complexity of the above program is O(n)
- Space Complexity is O(1)