RATING SYSTEM

Linear Queue

Elo Rating System

PROBLEM STATEMENT

To create a Rating System using compare and select technique with the help of Queue Data Structures.

DETAILED PROBLEM STATEMENT

From mobile phones to dentist services, it's rare to blindly make a purchase decision without reading through several online rating. In 2016, 90% of shoppers look at least one online rating before deciding to visiting a business.

Thus, we provide a much better and efficient way of providing the rating from a set of products. We intend to ask the user providing them with only 2 options at a time rather that asking to select one among many. This rating from a 2v2 among a list on n objects is known as Elo Rating System. The Elo rating system is a method for calculating the relative skill levels of players in zero-sum games such as chess. It is named after its creator Arpad Elo, a Hungarian-American physics professor.

We intend to use the Queue data structures to create memory for the objects and enqueue all the new objects they want to add simultaneously. We use the queue for the reason that Queues are flexible, requiring no communications programming. The programmer does not need any knowledge of inter-process communication. Data queues allow computers to handle multiple tasks. The queue can remain active when there are no entries, ready to process data entries when necessary.

DATA STRUCTURES

QUEUE DATA STRUCTURE

Queue Data Structure has been used in this following program because Data queues are a fast method of inter-process communication. Data queues free up jobs from performing some work, which can lead to a better response time and an overall improvement in system performance. Data queues serve as the fastest form of asynchronous communication between two different tasks, since there is less overhead than with database files and data areas. Queues are flexible, requiring no communications programming. The programmer does not need any knowledge of inter-process communication. Data queues allow computers to handle multiple tasks. The queue can remain active when there are no entries, ready to process data entries when necessary. Some jobs have performance restraints and cannot handle all the entries, so the data entries are spread out across multiple jobs. For example, only one customer service representative can help a customer at a time, so the queue can spread customer service requests among the representatives, for quicker processing.

OTHER DATA STRUCTURES

Data Structures which could be helpful for this project would be

1. Linked List: - Will help to link the object and the rating of that particular object rather than creating a new queue for that.
2. Doubly Linked List: - Will help to create the link between object, objects previous rating and objects current rating. Also, as dynamic initialization is there, there will be no wastage of memory.

FUNCTIONS

PROBABILITY

Probability function finds the probability of one option being selected by the user.
P1: Probability of winning of player with rating2
P2: Probability of winning of player with rating1.
P1 = (1.0 / (1.0 + pow(10, ((rating1 – rating2) / 400))));
P2 = (1.0 / (1.0 + pow(10, ((rating2 – rating1) / 400))));
P1 + P2 = 1.

ELORATING

The rating of an option selected by the user is updated using the formula given below :-
rating1 = rating1 + K*(Actual Score – Expected score);

INSQ

INSQ function is used to Insert the objects provided by the user.

DELQ

DELQ function is used to Delete the objects provided by the user.

ALGORITHM

INSQ

Step 1: If REAR >= SIZE – 1 then
    Write “Queue is Overflow”
Step 2: REAR = REAR + 1
Step 3: QUEUE [REAR] = X
Step 4: If FRONT = -1 then
    FRONT = 0.
Step 5: End

DELQ

Step 1: If FRONT = -1 then
    Write “Queue is Underflow”
Step 2: Return QUEUE [FRONT]
Step 3: If FRONT = REAR then
    FRONT = 0
    REAR = 0
    Else
    FRONT = FRONT + 1
Step 4: End

PROBABILITY

Step 1: Initialize P1, P2
Step 2: P1 = (1.0 / (1.0 + pow(10, ((rating1 – rating2) / 400))));
     P2 = (1.0 / (1.0 + pow(10, ((rating2 – rating1) / 400))));
Step 3: End

ELORATING

Step 1: Initialize Ra, Rb, K, Pa, Pb
Step 2: Check if d==1, if No, go to step
Step 3: Set Ra = Ra + K * (1 - Pa)
     Set Rb = Rb + K * (0 - Pb)
Step 4: Set Ra = Ra + K * (0 - Pa)
     Set Rb = Rb + K * (1 - Pb)
Step 5: End

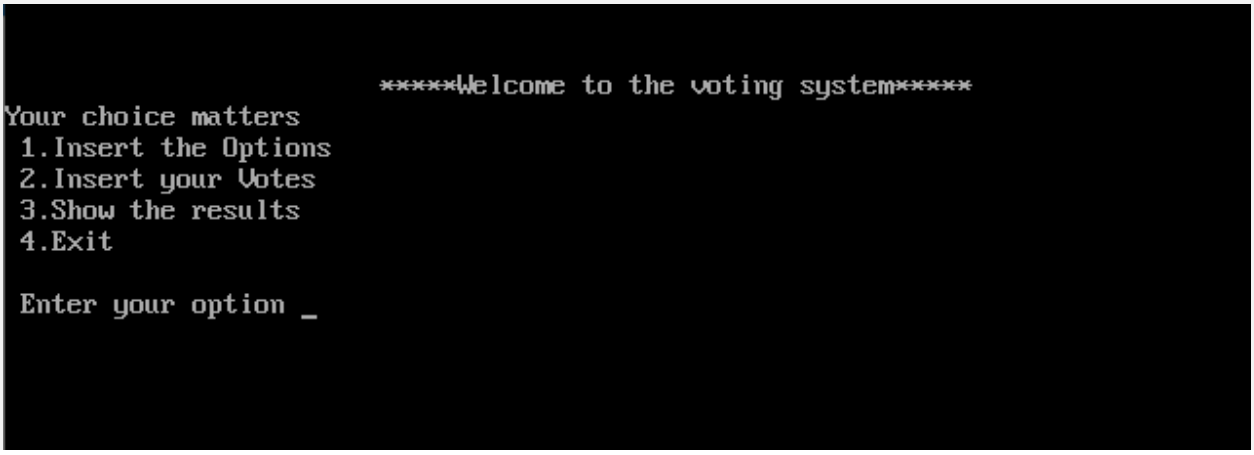
PROGRAM

```
#include<stdio.H>
#include<conio.H>
#include<math.H>
#include<string.h>
#include<stdlib.h>
#define max 20
int q=0;
int nu=0;
int d;
float Ra[100];
float Probability(int rating1, int rating2)
{   float g= rating1-rating2;
    float a=g/400;
    float b=pow(10,a);
    float c=1+b;
    float e=1.00/c;
    return e;
}
void EloRating(float Ra, float Rb, int K, int d)
{

    float Pb = Probability(Ra, Rb);
    float Pa = Probability(Rb, Ra);
    if (d == 1) {
       Ra = Ra + K * (1 - Pa);
       Rb = Rb + K * (0 - Pb);
    }
    else {
       Ra = Ra + K * (0 - Pa);
       Rb = Rb + K * (1 - Pb);
    }

    printf("Updated Ratings:-\n");
    printf("Ra = %f",Ra);
    printf("\nRb = %f",Rb);
}

int insq(char queue[max][80], int *rear, char data[80])
{
       if(*rear == max -1)
       return(-1);
       else
       {
       *rear = *rear + 1;
       strcpy(queue[*rear], data);
       return(1);
       }
}
int delq(char queue[max][80], int *front, int *rear, char data[80])
{
if(*front == *rear)
return(-1);
else
{
(*front)++;
```

```
strcpy(data, queue[*front]);
return(1);
}
}
char *peek(char queue[max][80], int *nu, char data[80])
{
if (*nu==-1)
printf("Wrong");
else
strcpy(data, queue[*nu]);
return queue[*nu];
}

int main()
{
       int op1;
       clrscr();
do
{
       clrscr();
       printf("\n\n\n\t\t\t*****Welcome to the voting system***** \nYour
choice matters");
       printf("\n 1.Insert the Options");
       printf("\n 2.Insert your Votes");
       printf("\n 3.Show the results");
       printf("\n 4.Exit");
       printf("\n\n Enter your option ");
       scanf("%d",&op1);
       switch(op1)
       {
       case 1:
              clrscr();
              char queue[max][80], data[80];
              int front, rear, reply;
              int ch;
              front = rear = -1; //... Initialize a Queue
              do
              {
              printf("------------------------------\n");
              printf("\tMenu");
              printf("\n------------------------------");
              printf("\n 1. Insert Your Option In The List");
              printf("\n 2. Delete Your Option From The Lisy");
              printf("\n.3. Enter To Read Your Particular Option");
              printf("\n 4. Exit");
              printf("\n------------------------------n");
              printf("\nChoose operation ");
              scanf("%d", &ch);
              switch(ch)
              {
              case 1:
               printf("\nEnter Your Option  ");
               scanf("%s",data);
               nu=nu+1;
               reply = insq(queue,&rear,data);
               if(reply == -1 )
                printf("\nOOPS! The List is Full \n");
                else
```

```
               printf("\n '%s' is inserted in the List.\n\n",data);
                break;
              case 2:
                reply = delq(queue, &front, &rear, data);
                if( reply == -1 )
                printf("\nThe List is Empty \n");
                else
                {
                printf("\nDeleted Option from the List is  %s\n", data);
                nu=nu-1;
                }
                printf("\n%d",nu);
                break;
              case 3:  printf("Enter the no");
                int no;
                scanf("%d",&nu);
                char var[90];
                *var=*peek(queue,&nu,data);
                printf("\n %s \n",data);
                break;
                }
                }while(ch<4);
                break;
       case 2:   for(int j=0;j<6;j++)
                {
                int d;
                int K = 30;
                srand(time(0));
                int a=1,b=1,v1=1,v2=1;
                char data1[80],data2[80];
                while(a==b && v1==v2)
                  {
                  a=rand()%nu;
                  b=rand()%nu;
                  v1=a;
                  v2=b;
                  }
                peek(queue,&a,data1);
                peek(queue,&b,data2);
                printf("\nWhich Do you Choose\n \t\t\t 1.%s or
2.%s\n",data1,data2);
                scanf("%d",&d);
                if(q==0)
                {
                     for(int i=0;i<nu;i++)
                            {
                            Ra[i]=1000;
                            }
                }
                if(d==1)
                {
                float Pb=Probability(Ra[a],Ra[b]);
                float Pa=Probability(Ra[b],Ra[a]);
                d=1;
                Ra[a]=Ra[a]+K*(1-Pa);
                Ra[b]=Ra[b]+K*(0-Pb);
                }
                else
```

```
                {
                d=0;
                float Pb=Probability(Ra[a],Ra[b]);
                float Pa=Probability(Ra[b],Ra[a]);
                Ra[a]=Ra[a]+K*(0-Pa);
                Ra[b]=Ra[b]+K*(1-Pb);
                }
                q=q+1;
                }
                getch();
                break;
        case 3:
                clrscr();
                printf("\n The Following are the ratings\n");
                for(int i=0;i<nu;i++)
                {
                char var[90];
                *var=*peek(queue,&i,data);
                printf("\n %s:- %f \n",data,Ra[i]);
                }
                getch();
                break;
        }
}while (op1<4);
getch();
return 0;
}
```



```
------------------------------
       Menu
------------------------------
 1. Insert Your Option In The List
 2. Delete Your Option From The Lisy
.3. Enter To Read Your Particular Option
 4. Exit
------------------------------n
Choose operation 1

Enter Your Option  Apples

 'Apples' is inserted in the List.

------------------------------
       Menu
------------------------------
 1. Insert Your Option In The List
 2. Delete Your Option From The Lisy
.3. Enter To Read Your Particular Option
 4. Exit
------------------------------n
Choose operation 1

Enter Your Option  Oranges
```

]]

```
 2.Insert your Votes
 3.Show the results
 4.Exit

 Enter your option 2

Which Do you Choose
                        1.Litchis or 2.Strawberries
1

Which Do you Choose
                        1.Strawberries or 2.watermelon
1

Which Do you Choose
                        1.Mangoes or 2.Apples
1

Which Do you Choose
                        1.Strawberries or 2.Oranges
1

Which Do you Choose
                        1.Grapes or 2.Mangoes
```

```
The Following are the ratings

Apples:- 985.000000

Oranges:- 985.000000

Mangoes:- 1029.352783

Grapes:- 985.647217

Litchis:- 1015.000000

Chickoos:- 984.309692

Strawberries:- 1015.647217

watermelon:- 1000.043091
```

CONCLUSION

The Program was executed and the output helps us to achieve what to choose from the objects. The use of queue data structures helps us to understand its importance.