**Linear sort Algorithm to sort reviews and top users**

Linear sorting algorithm that uses **count sort** in this application as a technique for each user who logs in to keep a count of movies watched and number of movies reviewed by each user and the resultant data is presented in the application.

Each interior node performs the comparison of 2 elements and leaves represent the order of all the elements passed in the array as a list of movies.

The algorithm in turn constructs a decision tree to hold the n leaf values where each order of values corresponds to one path from the initial node to the leaf node.

Count sort using a technique called transform and conquer where each input is added to a table of count and indexes each location internally to identify and sort the values passed.

Linear sorts do not use comparison operators to retrieve information about the value of each user, hence are used efficiently here since we are only sorting digits and not strings.

The algorithm reads user reviews

1. Sortbycount holds a return value of the function that holds the array of user reviews
2. Createinput\_moviewatched is assigned to a function that is created to hold an array of input objects

**var** createinput\_moviewatched = **function** (inputModel, inputSize, inputType) {

**var** input\_moviewatcheds = [];

**for** (**var** i = 0; i < inputSize; i++) {

**if** (inputType === 'username') {

input\_moviewatcheds[i] = Math.floor(Math.random() \* 1024);

} **else** {

input\_moviewatcheds[i] = **new** inputModel();

}

}

1. Initially assign input\_moviewatched to 0

**function** inputModel (value){

**this**.name = $(moviename),

**this**.value = value

};

**var** input\_moviewatched = createinput\_moviewatched(inputModel, inputSize, inputType);

**var** user\_count = [];

**var** sortedResult = [];

**var** review\_count = 0;

1. Each frequency is held into an array based on the number of movies watched by each user.

**for** (**var** j=0; j < input\_moviewatched.length; j++) {

**if** (inputType === 'username') {

user\_count[input\_moviewatched[j]] += 1;

} **else** {

user\_count[input\_moviewatched[j].value] += 1;

}

}

1. We add the number of times the input occurs in the resultant value and finally assign it to the user and sort the values in the descending order and post it as the top users and their respective review/movie watched count.

*// Add the number of times this integer occurs (count) into the result*

**for** (**var** k=0; k < user\_count.length; k++) {

**var** count = user\_count[k];

**for** (**var** l=0; l < count; l++) {

**if** (inputType === 'username') {

sortedResult[review\_count] = k;

} **else** {

sortedResult[review\_count] = **new** inputModel(k);

}

review\_count++;

}

}

**return** sortedResult;

The return value of the function is returned to app.js and is sent to index.ejs view to display the resultant content available from the view module.