Chat GPT Check and Analysis

# Introduction

This report focuses on optimizing ChatGPT using queries in the Arabic language through different formats and tones.

# Problem: Social Media Feed Optimization

In the domain of social media feed optimization, our focus centers on enhancing user experience through effective content curation. Our ongoing efforts involve the development of advanced algorithms aimed at analyzing and optimizing the arrangement of posts within a user's feed. The primary objective is to intelligently assess the relevance and interest level of each post for individual users. Through the implementation of sophisticated search and sorting algorithms that surpass traditional methods, we strive to create a dynamic feed optimization system. This system adapts to user preferences and behaviors, improving the performance of content recommendation and sorting mechanisms. The end goal is to present users with content that aligns closely with their interests, leading to a more engaging and personalized social media experience.

# Linear Search

Linear search is implemented in personalized content recommendation for its efficiency with sorted lists, early termination capability, simplicity of implementation, and suitability for small datasets. It allows for quick identification of the first relevant post, making it a pragmatic choice when the dataset is not extremely large and the simplicity of the algorithm is advantageous. However, for larger datasets or more complex ranking criteria, other advanced algorithms may be more appropriate.

# Quick Sort

Content sorting with Quicksort is implemented for its efficiency, especially in scenarios where sorting large datasets is common. Quicksort's advantages include its in-place sorting capability, adaptability to partially sorted data, divide-and-conquer strategy, and low overhead. This makes it well-suited for applications like personalized content recommendation, where quick and resource-efficient sorting of dynamic and frequently updated content is essential. While it has a worst-case time complexity of O(n^2), practical implementations, especially with randomized pivot selection, often perform well in real-world scenarios.

# Binary Search

Binary Search is implemented for sponsored content due to its efficiency with sorted data. It offers quick and consistent search operations with a time complexity of O(log n), making it ideal for identifying relevant sponsored content in large datasets. The algorithm's ability to reduce search time and maintain performance regardless of dataset size, along with its resource efficiency, makes it a suitable choice for optimizing the retrieval of sponsored content in scenarios where the data is sorted based on relevance, cost, or other criteria.

# Methodology

ChatGPT was asked to follow through with the code and update it as necessary to optimize its performance and correct errors along the way all while making the code simpler and more accurate. Moreover, it was done in both Arabic and English to test the difference between the chatbot’s performance in both language and its response to spelling mistakes in Arabic and English.

# Queries

The popular chatbot was asked to provide a code for our problem (Social Media Feed Optimization) and make edits according to the following queries:

A- Write simple code in C# for these Algorithm:

1. Content Sorting with Quicksort
2. Linear Search for Personalized Content
3. Binary Search for Sponsored Content

B- Optimize the code for performance.

C- Make the code simpler.

D-اكتب برنامج بلغة لهذه الخورزميات: C#

1. تصنيف المحتوى باستخدام خوارزمية التصنيف السريع
2. تصنيف المحتوى الشخصي باستخدام خوارزمية البحث الخطي
3. تصنيف المحتوى الدعائي باستخدام خوارزمية البحث الثنائي

E- عدل على الكود بالشكل الأمثل للأداء

F-أعد كتابة الكود بشكل مبسط أكثر

# Performance Measures Used

1. Time complexity
2. Code runtime
3. Code simplicity
4. Response Time

# English Queries

1. Quicksort
2. gimme code for this algo in tis prolem Content Sorting with Quicksort in social feed optimizing c#:

using System;

class SocialContent

{

public int Timestamp { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(int timestamp /\*, other properties \*/)

{

Timestamp = timestamp;

// Initialize other properties

}

}

class Program

{

static void Main()

{

// Sample social content array

SocialContent[] contentArray = {

new SocialContent(1637869200 /\*timestamp1\*/),

new SocialContent(1637872800 /\*timestamp2\*/),

new SocialContent(1637876400 /\*timestamp3\*/),

// Add more content with timestamps

};

// Sort the content array based on timestamps

Quicksort(contentArray, 0, contentArray.Length - 1);

// Display sorted content

Console.WriteLine("Sorted Content Based on Timestamps:");

foreach (var content in contentArray)

{

Console.WriteLine($"Timestamp: {content.Timestamp}");

// Display other properties as needed

}

}

// Quicksort algorithm implementation

static void Quicksort(SocialContent[] arr, int low, int high)

{

if (low < high)

{

int pivotIndex = Partition(arr, low, high);

Quicksort(arr, low, pivotIndex - 1);

Quicksort(arr, pivotIndex + 1, high);

}

}

static int Partition(SocialContent[] arr, int low, int high)

{

int pivot = arr[high].Timestamp;

int i = low - 1;

for (int j = low; j < high; j++)

{

if (arr[j].Timestamp <= pivot)

{

i++;

Swap(arr, i, j);

}

}

Swap(arr, i + 1, high);

return i + 1;

}

static void Swap(SocialContent[] arr, int i, int j)

{

SocialContent temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

* time complexity: O(n^2)
* response time: medium

1. Provide a code in C# for content sorting in social media feed optimization using the quicksort algorithm:

using System;

using System.Collections.Generic;

class SocialContent

{

public int Timestamp { get; set; }

public string Content { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(int timestamp, string content /\*, other properties \*/)

{

Timestamp = timestamp;

Content = content;

// Initialize other properties

}

}

class SocialMediaFeed

{

// Quicksort algorithm for sorting SocialContent based on Timestamps

public static void Quicksort(List<SocialContent> contentList, int low, int high)

{

if (low < high)

{

int pivotIndex = Partition(contentList, low, high);

Quicksort(contentList, low, pivotIndex - 1);

Quicksort(contentList, pivotIndex + 1, high);

}

}

// Partition method for Quicksort

public static int Partition(List<SocialContent> contentList, int low, int high)

{

int pivot = contentList[high].Timestamp;

int i = low - 1;

for (int j = low; j < high; j++)

{

if (contentList[j].Timestamp <= pivot)

{

i++;

Swap(contentList, i, j);

}

}

Swap(contentList, i + 1, high);

return i + 1;

}

// Helper method to swap elements in the list

public static void Swap(List<SocialContent> contentList, int i, int j)

{

SocialContent temp = contentList[i];

contentList[i] = contentList[j];

contentList[j] = temp;

}

}

class Program

{

static void Main()

{

// Sample social content list

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent(1637869200, "Content1" /\*timestamp1, content1\*/),

new SocialContent(1637872800, "Content2" /\*timestamp2, content2\*/),

new SocialContent(1637876400, "Content3" /\*timestamp3, content3\*/),

// Add more content with timestamps and content

};

// Sort the content list based on timestamps

SocialMediaFeed.Quicksort(contentList, 0, contentList.Count - 1);

// Display sorted content

Console.WriteLine("Sorted Content Based on Timestamps:");

foreach (var content in contentList)

{

Console.WriteLine($"Timestamp: {content.Timestamp}, Content: {content.Content}");

// Display other properties as needed

}

}

}

* time complexity: O(n^2)
* response time: faster

1. Optimize code for performance and simplicity:

using System;

using System.Collections.Generic;

class SocialContent

{

public int Timestamp { get; set; }

public string Content { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(int timestamp, string content /\*, other properties \*/)

{

Timestamp = timestamp;

Content = content;

// Initialize other properties

}

}

class Program

{

static void Main()

{

// Sample social content list

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent(1637869200, "Content1" /\*timestamp1, content1\*/),

new SocialContent(1637872800, "Content2" /\*timestamp2, content2\*/),

new SocialContent(1637876400, "Content3" /\*timestamp3, content3\*/),

// Add more content with timestamps and content

};

// Sort the content list based on timestamps

contentList.Sort((x, y) => x.Timestamp.CompareTo(y.Timestamp));

// Display sorted content

Console.WriteLine("Sorted Content Based on Timestamps:");

foreach (var content in contentList)

{

Console.WriteLine($"Timestamp: {content.Timestamp}, Content: {content.Content}");

// Display other properties as needed

}

}

}

* time complexity: O(n log n)
* response time: fastest

1. Linear Search:
   * + 1. gimme code for this algo in tis prolem personalized contetn with linear serch in social feed optimizing c#:

using System;

using System.Collections.Generic;

class SocialContent

{

public string Content { get; set; }

public List<string> Tags { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(string content, List<string> tags /\*, other properties \*/)

{

Content = content;

Tags = tags;

// Initialize other properties

}

}

class SocialMediaFeed

{

// Method to perform a linear search for personalized content

public static List<SocialContent> FindPersonalizedContent(List<SocialContent> contentList, string keyword)

{

List<SocialContent> personalizedContent = new List<SocialContent>();

foreach (var content in contentList)

{

// Assuming the search is based on tags or keywords within the content

if (content.Tags.Contains(keyword))

{

personalizedContent.Add(content);

}

}

return personalizedContent;

}

}

class Program

{

static void Main()

{

// Sample social content list

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent("Content1", new List<string> { "technology", "coding", "AI" }),

new SocialContent("Content2", new List<string> { "science", "research", "discovery" }),

new SocialContent("Content3", new List<string> { "coding", "programming", "software" }),

// Add more content with tags

};

// User's personalized keyword

string userKeyword = "coding";

// Find personalized content based on user's keyword

List<SocialContent> personalizedContent = SocialMediaFeed.FindPersonalizedContent(contentList, userKeyword);

// Display personalized content

Console.WriteLine($"Personalized content for keyword '{userKeyword}':");

foreach (var content in personalizedContent)

{

Console.WriteLine($"Content: {content.Content}");

// Display other properties as needed

}

}

}

-response time: slow to medium

-time complexity: O(n)

* + - 1. Provide a code in C# for personalized content in social media feed optimization using the linear search algorithm:

using System;

using System.Collections.Generic;

class SocialContent

{

public string Content { get; set; }

public List<string> Tags { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(string content, List<string> tags /\*, other properties \*/)

{

Content = content;

Tags = tags;

// Initialize other properties

}

}

class SocialMediaFeed

{

// Method to perform a linear search for personalized content

public static List<SocialContent> FindPersonalizedContent(List<SocialContent> contentList, string userInterest)

{

List<SocialContent> personalizedContent = new List<SocialContent>();

foreach (var content in contentList)

{

// Assuming the search is based on tags or keywords within the content

if (content.Tags.Contains(userInterest))

{

personalizedContent.Add(content);

}

}

return personalizedContent;

}

}

class Program

{

static void Main()

{

// Sample social content list

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent("Content1", new List<string> { "technology", "coding", "AI" }),

new SocialContent("Content2", new List<string> { "science", "research", "discovery" }),

new SocialContent("Content3", new List<string> { "coding", "programming", "software" }),

// Add more content with tags

};

// User's personalized interest

string userInterest = "coding";

// Find personalized content based on user's interest

List<SocialContent> personalizedContent = SocialMediaFeed.FindPersonalizedContent(contentList, userInterest);

// Display personalized content

Console.WriteLine($"Personalized content for interest '{userInterest}':");

foreach (var content in personalizedContent)

{

Console.WriteLine($"Content: {content.Content}");

// Display other properties as needed

}

}

}

-response time: slow to medium

-time complexity: O(n)

* + - 1. Optimize code for performance and simplicity:

using System;

using System.Collections.Generic;

using System.Linq;

class SocialContent

{

public string Content { get; set; }

public List<string> Tags { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(string content, List<string> tags /\*, other properties \*/)

{

Content = content;

Tags = tags;

// Initialize other properties

}

}

class Program

{

static void Main()

{

// Sample social content list

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent("Content1", new List<string> { "technology", "coding", "AI" }),

new SocialContent("Content2", new List<string> { "science", "research", "discovery" }),

new SocialContent("Content3", new List<string> { "coding", "programming", "software" }),

// Add more content with tags

};

// User's personalized interest

string userInterest = "coding";

// Find personalized content based on user's interest

var personalizedContent = contentList.Where(content => content.Tags.Contains(userInterest)).ToList();

// Display personalized content

Console.WriteLine($"Personalized content for interest '{userInterest}':");

foreach (var content in personalizedContent)

{

Console.WriteLine($"Content: {content.Content}");

// Display other properties as needed

}

}

}

-response time: fast

-time complexity: O(n)

1. Binary Search for Sponsored Content:

1. gimme code for this algo in tis prolem sponsore contetn with binary serch in social feed optimizing c#:

using System;

using System.Collections.Generic;

class SocialContent

{

public string Content { get; set; }

public bool IsSponsored { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(string content, bool isSponsored /\*, other properties \*/)

{

Content = content;

IsSponsored = isSponsored;

// Initialize other properties

}

}

class SocialMediaFeed

{

// Method to perform binary search for sponsored content

public static int FindSponsoredContent(List<SocialContent> contentList)

{

int left = 0;

int right = contentList.Count - 1;

while (left <= right)

{

int mid = left + (right - left) / 2;

// Check if the middle content is sponsored

if (contentList[mid].IsSponsored)

{

// Check if it's the first occurrence of sponsored content

if (mid == 0 || !contentList[mid - 1].IsSponsored)

{

return mid; // Found sponsored content

}

else

{

// Move left to find the first occurrence of sponsored content

right = mid - 1;

}

}

else

{

// Move right to search in the right half

left = mid + 1;

}

}

return -1; // Sponsored content not found

}

}

class Program

{

static void Main()

{

// Sample social content list (assuming it's sorted by sponsorship status)

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent("Content1", false),

new SocialContent("Content2", false),

new SocialContent("Content3", true),

new SocialContent("Content4", true),

new SocialContent("Content5", true),

new SocialContent("Content6", false),

// Add more content with sorted sponsorship status

};

// Find the index of the first sponsored content

int sponsoredIndex = SocialMediaFeed.FindSponsoredContent(contentList);

// Display the index of sponsored content

if (sponsoredIndex != -1)

{

Console.WriteLine($"Index of first sponsored content: {sponsoredIndex}");

Console.WriteLine($"Sponsored content: {contentList[sponsoredIndex].Content}");

}

else

{

Console.WriteLine("Sponsored content not found.");

}

}

}

-response time: slow

-time complexity: O(log n)

2. Provide a code in C# for sponsored content in social media feed optimization using the binarysearch algorithm:

using System;

using System.Collections.Generic;

class SocialContent

{

public string Content { get; set; }

public bool IsSponsored { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(string content, bool isSponsored /\*, other properties \*/)

{

Content = content;

IsSponsored = isSponsored;

// Initialize other properties

}

}

class Program

{

static void Main()

{

// Sample social content list (assuming it's sorted by sponsorship status)

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent("Content1", false),

new SocialContent("Content2", false),

new SocialContent("Content3", true),

new SocialContent("Content4", true),

new SocialContent("Content5", true),

new SocialContent("Content6", false),

// Add more content with sorted sponsorship status

};

// Perform binary search to find the first sponsored content

int sponsoredIndex = contentList.FindIndex(c => c.IsSponsored);

// Display the index of sponsored content

if (sponsoredIndex != -1)

{

Console.WriteLine($"Index of first sponsored content: {sponsoredIndex}");

Console.WriteLine($"Sponsored content: {contentList[sponsoredIndex].Content}");

}

else

{

Console.WriteLine("Sponsored content not found.");

}

}

}

-response time: medium

-time complexity: O(n)

3. Optimize the code for simplicity and performance:

using System;

using System.Collections.Generic;

class SocialContent

{

public string Content { get; set; }

public bool IsSponsored { get; set; }

// Other properties related to the social content

// Constructor

public SocialContent(string content, bool isSponsored /\*, other properties \*/)

{

Content = content;

IsSponsored = isSponsored;

// Initialize other properties

}

}

class Program

{

static void Main()

{

// Sample social content list (assuming it's sorted by sponsorship status)

List<SocialContent> contentList = new List<SocialContent>

{

new SocialContent("Content1", false),

new SocialContent("Content2", false),

new SocialContent("Content3", true),

new SocialContent("Content4", true),

new SocialContent("Content5", true),

new SocialContent("Content6", false),

// Add more content with sorted sponsorship status

};

// Perform binary search to find the first sponsored content

int sponsoredIndex = contentList.BinarySearch(new SocialContent("", true), Comparer<SocialContent>.Create((x, y) => x.IsSponsored.CompareTo(y.IsSponsored)));

// Adjust index if the element is not found

if (sponsoredIndex < 0)

{

sponsoredIndex = ~sponsoredIndex; // Get the bitwise complement to find the insertion point

}

// Display the index of sponsored content

if (sponsoredIndex < contentList.Count && contentList[sponsoredIndex].IsSponsored)

{

Console.WriteLine($"Index of first sponsored content: {sponsoredIndex}");

Console.WriteLine($"Sponsored content: {contentList[sponsoredIndex].Content}");

}

else

{

Console.WriteLine("Sponsored content not found.");

}

}

}

-response time: fast

-time complexity: O(log n)

# Conclusions for English Queries

In the English, we notice that the more specific a query gets, the faster the response time. Moreover, generally speaking, the more specific a query is the higher time complexity we get. This points to the fact that ChatGPT is very proficient in English as it was the language it was first trained on. To add to that, ChatGPT is very sensitive to the dialect and tone used. Essentially that means if we use a more specific and formal tone, it would do a better job at understanding what we need and achieving it. Moreover, it achieves it at a much better response and provides a fully functioning code that is not complicated and in fact optimized for performance.

Incorrect query test

1. اكتب بلغة لهذه الخورزمي: C#

تصنيف المحتوى باستخدام خوارزمية من ابل التصنيف السريع

using System;  
using System.Collections.Generic;  
using System.IO;  
using System.Linq;  
using Facebook.FastText;  
  
class Program  
{  
 static void Main()  
 {  
 // تحديد مسار ملف التدريب  
 string trainingDataPath = "path/to/training\_data.txt";  
  
 // تحديد مسار ملف النموذج المدرب  
 string modelPath = "path/to/model.bin";  
  
 // تدريب النموذج  
 TrainModel(trainingDataPath, modelPath);  
  
 // تحميل النموذج المدرب  
 var classifier = new FastTextClassifier(modelPath);  
  
 // مثال على تصنيف نص  
 string textToClassify = "هذا نص للتصنيف";  
 string predictedLabel = ClassifyText(classifier, textToClassify);  
  
 // عرض النتيجة  
 Console.WriteLine($"النص: {textToClassify}");  
 Console.WriteLine($"التصنيف المتوقع: {predictedLabel}");  
 }  
  
 static void TrainModel(string trainingDataPath, string modelPath)  
 {  
 // تحديد خيارات التدريب  
 var trainOptions = new FastTextOptions  
 {  
 Loss = LossName.HierarchicalSoftmax,  
 LearningRate = 0.1,  
 Epoch = 25,  
 WordNgrams = 2  
 };  
  
 // إعداد معلم النموذج  
 var trainer = new FastTextTrainer(trainOptions);  
  
 // تدريب النموذج  
 trainer.Train(trainingDataPath, modelPath);  
 }  
  
 static string ClassifyText(FastTextClassifier classifier, string text)  
 {  
 // تصنيف النص  
 var result = classifier.Predict(text);  
  
 // الحصول على التصنيف الأعلى  
 var topPrediction = result.Predictions.OrderByDescending(p => p.Probability).First();  
  
 return topPrediction.Label;  
 }  
}

time complexity : O(E \* N \* log(V))

2.

-اكتب برنامج بلغة لهذه الخورزميات: C#

تصنيف المحتوى الشخص باخدام خوارزمي البحث الخطي

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى وتصنيفه  
 string[] content = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 Console.WriteLine("الرجاء إدخال المحتوى الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الخطي  
 int index = LinearSearch(content, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{content[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى.");  
 }  
 }  
  
 // خوارزمية البحث الخطي  
 static int LinearSearch(string[] array, string target)  
 {  
 for (int i = 0; i < array.Length; i++)  
 {  
 if (array[i].Equals(target, StringComparison.OrdinalIgnoreCase))  
 {  
 return i; // تم العثور على المحتوى  
 }  
 }  
 return -1; // المحتوى غير موجود  
 }  
}

time complexity : O(n)

3. D-اكتب برنامج بلغة لهذه الخورزميات: C#

تصنيف المحوى الدعائ باستخدام خوارزمة في البحث الثنائي

Ans:

هذا هو برنامج بلغة C# يستخدم خوارزمية البحث الثنائي لتصنيف المحتوى الدعائي:

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الدعائي وتصنيفه  
 string[] advertisingContent = { "إعلان1", "إعلان2", "إعلان3", "إعلان4", "إعلان5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 // يجب أن تكون المصفوفة مرتبة لاستخدام البحث الثنائي  
 Array.Sort(advertisingContent);  
  
 Console.WriteLine("الرجاء إدخال المحتوى الدعائي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الثنائي  
 int index = BinarySearch(advertisingContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{advertisingContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الدعائي.");  
 }  
 }  
  
 // خوارزمية البحث الثنائي  
 static int BinarySearch(string[] array, string target)  
 {  
 int left = 0;  
 int right = array.Length - 1;  
  
 while (left <= right)  
 {  
 int middle = (left + right) / 2;  
 int compareResult = string.Compare(array[middle], target);  
  
 if (compareResult == 0)  
 {  
 return middle; // تم العثور على المحتوى  
 }  
 else if (compareResult < 0)  
 {  
 left = middle + 1; // البحث في النصف الأيمن  
 }  
 else  
 {  
 right = middle - 1; // البحث في النصف الأيسر  
 }  
 }  
  
 return -1; // المحتوى الدعائي غير موجود  
 }  
}

time complexity : O(log n)

Correct Query :

1.اكتب برنامج بلغة لهذه الخورزميات: C#

تصنيف المحتوى باستخدام خوارزمية التصنيف السريع

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى وتصنيفه  
 string[] content = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 // تصنيف المحتوى باستخدام خوارزمية التصنيف السريع  
 QuickSort(content, ratings, 0, content.Length - 1);  
  
 Console.WriteLine("المحتوى بعد التصنيف:");  
  
 for (int i = 0; i < content.Length; i++)  
 {  
 Console.WriteLine($"المحتوى: {content[i]}, التصنيف: {ratings[i]}");  
 }  
 }  
  
 // خوارزمية التصنيف السريع  
 static void QuickSort(string[] array, int[] ratings, int low, int high)  
 {  
 if (low < high)  
 {  
 int partitionIndex = Partition(array, ratings, low, high);  
  
 QuickSort(array, ratings, low, partitionIndex - 1);  
 QuickSort(array, ratings, partitionIndex + 1, high);  
 }  
 }  
  
 static int Partition(string[] array, int[] ratings, int low, int high)  
 {  
 string pivot = array[high];  
 int i = low - 1;  
  
 for (int j = low; j < high; j++)  
 {  
 if (String.Compare(array[j], pivot) < 0)  
 {  
 i++;  
 Swap(array, i, j);  
 Swap(ratings, i, j);  
 }  
 }  
  
 Swap(array, i + 1, high);  
 Swap(ratings, i + 1, high);  
  
 return i + 1;  
 }  
  
 static void Swap<T>(T[] array, int index1, int index2)  
 {  
 T temp = array[index1];  
 array[index1] = array[index2];  
 array[index2] = temp;  
 }  
}

time complexity : O(n log n)

2.

اكتب برنامج بلغة لهذه الخورزميات: C#

تصنيف المحتوى الشخصي باستخدام خوارزمية البحث الخطي

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الشخصي وتصنيفه  
 string[] personalContent = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 Console.WriteLine("الرجاء إدخال المحتوى الشخصي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الخطي  
 int index = LinearSearch(personalContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{personalContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الشخصي.");  
 }  
 }  
  
 // خوارزمية البحث الخطي  
 static int LinearSearch(string[] array, string target)  
 {  
 for (int i = 0; i < array.Length; i++)  
 {  
 if (array[i].Equals(target, StringComparison.OrdinalIgnoreCase))  
 {  
 return i; // تم العثور على المحتوى  
 }  
 }  
 return -1; // المحتوى غير موجود  
 }  
}

time complexity : O(n)

اكتب برنامج بلغة لهذه الخورزميات: C#

تصنيف المحتوى الدعائي باستخدام خوارزمية البحث الثنائي

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الدعائي وتصنيفه  
 string[] advertisingContent = { "إعلان1", "إعلان2", "إعلان3", "إعلان4", "إعلان5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 // يجب أن تكون المصفوفة مرتبة لاستخدام البحث الثنائي  
 Array.Sort(advertisingContent);  
  
 Console.WriteLine("الرجاء إدخال المحتوى الدعائي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الثنائي  
 int index = BinarySearch(advertisingContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{advertisingContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الدعائي.");  
 }  
 }  
  
 // خوارزمية البحث الثنائي  
 static int BinarySearch(string[] array, string target)  
 {  
 int left = 0;  
 int right = array.Length - 1;  
  
 while (left <= right)  
 {  
 int middle = (left + right) / 2;  
 int compareResult = string.Compare(array[middle], target);  
  
 if (compareResult == 0)  
 {  
 return middle; // تم العثور على المحتوى  
 }  
 else if (compareResult < 0)  
 {  
 left = middle + 1; // البحث في النصف الأيمن  
 }  
 else  
 {  
 right = middle - 1; // البحث في النصف الأيسر  
 }  
 }  
  
 return -1; // المحتوى الدعائي غير موجود  
 }  
}

Time Complexity: O(log n)

Optimization:

التصنيف السريع

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى وتصنيفه  
 string[] content = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 // تصنيف المحتوى باستخدام خوارزمية التصنيف السريع  
 QuickSort(content, ratings, 0, content.Length - 1);  
  
 Console.WriteLine("المحتوى بعد التصنيف:");  
  
 for (int i = 0; i < content.Length; i++)  
 {  
 Console.WriteLine($"المحتوى: {content[i]}, التصنيف: {ratings[i]}");  
 }  
 }  
  
 // خوارزمية التصنيف السريع  
 static void QuickSort(string[] array, int[] ratings, int low, int high)  
 {  
 if (low < high)  
 {  
 int partitionIndex = Partition(array, ratings, low, high);  
  
 if (partitionIndex > 0)  
 {  
 QuickSort(array, ratings, low, partitionIndex - 1);  
 }  
  
 QuickSort(array, ratings, partitionIndex + 1, high);  
 }  
 }  
  
 static int Partition(string[] array, int[] ratings, int low, int high)  
 {  
 string pivot = array[high];  
 int i = low - 1;  
  
 for (int j = low; j < high; j++)  
 {  
 if (String.Compare(array[j], pivot) < 0)  
 {  
 i++;  
 Swap(array, i, j);  
 Swap(ratings, i, j);  
 }  
 }  
  
 Swap(array, i + 1, high);  
 Swap(ratings, i + 1, high);  
  
 return i + 1;  
 }  
  
 static void Swap<T>(T[] array, int index1, int index2)  
 {  
 T temp = array[index1];  
 array[index1] = array[index2];  
 array[index2] = temp;  
 }  
}

2.البحث الخطي  
using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الشخصي وتصنيفه  
 string[] personalContent = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 Console.WriteLine("الرجاء إدخال المحتوى الشخصي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الخطي  
 int index = LinearSearch(personalContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{personalContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الشخصي.");  
 }  
 }  
  
 // خوارزمية البحث الخطي  
 static int LinearSearch(string[] array, string target)  
 {  
 for (int i = 0; i < array.Length; i++)  
 {  
 if (string.Equals(array[i], target, StringComparison.OrdinalIgnoreCase))  
 {  
 return i; // تم العثور على المحتوى  
 }  
 }  
 return -1; // المحتوى غير موجود  
 }  
}

Time complexity: O(n)

3.البحث الثنائي

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الدعائي وتصنيفه  
 string[] advertisingContent = { "إعلان1", "إعلان2", "إعلان3", "إعلان4", "إعلان5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 Console.WriteLine("الرجاء إدخال المحتوى الدعائي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الثنائي  
 int index = BinarySearch(advertisingContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{advertisingContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الدعائي.");  
 }  
 }  
  
 // خوارزمية البحث الثنائي  
 static int BinarySearch(string[] array, string target)  
 {  
 int left = 0;  
 int right = array.Length - 1;  
  
 while (left <= right)  
 {  
 int middle = left + (right - left) / 2; // تجنب فرص الفاغرة الكبيرة  
 int compareResult = string.Compare(array[middle], target);  
  
 if (compareResult == 0)  
 {  
 return middle; // تم العثور على المحتوى  
 }  
 else if (compareResult < 0)  
 {  
 left = middle + 1; // البحث في النصف الأيمن  
 }  
 else  
 {  
 right = middle - 1; // البحث في النصف الأيسر  
 }  
 }  
  
 return -1; // المحتوى الدعائي غير موجود  
 }  
}

Time complexity : O(log n)

Code simplicity :

Binary Search:

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الدعائي وتصنيفه  
 string[] advertisingContent = { "إعلان1", "إعلان2", "إعلان3", "إعلان4", "إعلان5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 Console.WriteLine("الرجاء إدخال المحتوى الدعائي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // تصنيف المحتوى باستخدام البحث الثنائي  
 int index = BinarySearch(advertisingContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{advertisingContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الدعائي.");  
 }  
 }  
  
 // خوارزمية البحث الثنائي  
 static int BinarySearch(string[] array, string target)  
 {  
 int left = 0;  
 int right = array.Length - 1;  
  
 while (left <= right)  
 {  
 int middle = (left + right) / 2;  
 int compareResult = string.Compare(array[middle], target);  
  
 if (compareResult == 0)  
 {  
 return middle; // تم العثور على المحتوى  
 }  
 else if (compareResult < 0)  
 {  
 left = middle + 1; // البحث في النصف الأيمن  
 }  
 else  
 {  
 right = middle - 1; // البحث في النصف الأيسر  
 }  
 }  
  
 return -1; // المحتوى الدعائي غير موجود  
 }  
}

Time complexity : O(log n)

2-Quick sort :

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى وتصنيفه  
 string[] content = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 // تصنيف المحتوى باستخدام خوارزمية التصنيف السريع  
 QuickSort(content, ratings, 0, content.Length - 1);  
  
 Console.WriteLine("المحتوى بعد التصنيف:");  
  
 for (int i = 0; i < content.Length; i++)  
 {  
 Console.WriteLine($"المحتوى: {content[i]}, التصنيف: {ratings[i]}");  
 }  
 }  
  
 // خوارزمية التصنيف السريع  
 static void QuickSort(string[] array, int[] ratings, int low, int high)  
 {  
 if (low < high)  
 {  
 int partitionIndex = Partition(array, ratings, low, high);  
  
 QuickSort(array, ratings, low, partitionIndex - 1);  
 QuickSort(array, ratings, partitionIndex + 1, high);  
 }  
 }  
  
 static int Partition(string[] array, int[] ratings, int low, int high)  
 {  
 string pivot = array[high];  
 int i = low - 1;  
  
 for (int j = low; j < high; j++)  
 {  
 if (String.Compare(array[j], pivot) < 0)  
 {  
 i++;  
 Swap(array, i, j);  
 Swap(ratings, i, j);  
 }  
 }  
  
 Swap(array, i + 1, high);  
 Swap(ratings, i + 1, high);  
  
 return i + 1;  
 }  
  
 static void Swap<T>(T[] array, int index1, int index2)  
 {  
 T temp = array[index1];  
 array[index1] = array[index2];  
 array[index2] = temp;  
 }  
}

Time Complexity: O(n log n)

3-linear search :

using System;  
  
class Program  
{  
 static void Main()  
 {  
 // مصفوفة تحتوي على المحتوى الشخصي وتصنيفه  
 string[] personalContent = { "محتوى1", "محتوى2", "محتوى3", "محتوى4", "محتوى5" };  
 int[] ratings = { 5, 3, 1, 4, 2 };  
  
 Console.WriteLine("الرجاء إدخال المحتوى الشخصي الذي ترغب في تصنيفه:");  
 string userInput = Console.ReadLine();  
  
 // البحث الخطي  
 int index = LinearSearch(personalContent, userInput);  
  
 if (index != -1)  
 {  
 Console.WriteLine($"تم العثور على المحتوى '{personalContent[index]}' بتصنيف {ratings[index]}");  
 }  
 else  
 {  
 Console.WriteLine("لم يتم العثور على المحتوى الشخصي.");  
 }  
 }  
  
 // البحث الخطي  
 static int LinearSearch(string[] array, string target)  
 {  
 for (int i = 0; i < array.Length; i++)  
 {  
 if (string.Equals(array[i], target, StringComparison.OrdinalIgnoreCase))  
 {  
 return i; // تم العثور على المحتوى  
 }  
 }  
 return -1; // المحتوى غير موجود  
 }  
}

Time Complexity : O(n)

# Conclusions for Arabic Queries

In the case of Arabic queries, we notice the same behavioral pattern for time complexities, as the more specific a query is the better the time complexity. However, all response times for Arabic were the same, rather slow which was expected. Moreover, ChatGPT had a hard time understanding what was required when the queries were translated to Arabic and often produced unnecessarily incorrect answers. The answers tended to be imprecise.

# Outcomes for English Queries

# Outcomes for Arabic Queries

# Comparison of Outcomes of Both Languages

# 

|  |  |  |
| --- | --- | --- |
| **Performance Measure/Languages** | **English** | **Arabic** |
| Time Complexity | Lower the more optimized the code is | Lower the more optimized the code is |
| Code Runtime | Lower the more optimized the code is | Lower the more optimized the code is |
| Code Simplicity | Higher the more precise the query | Generally lower than English |
| Response Time | Lower | Generally higher than English |