Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

Лабораторная работа №1

“Алгоритмы и структуры данных”

Тема:”Алгоритмы поиска”

Проверил: Угнуряну Валерий

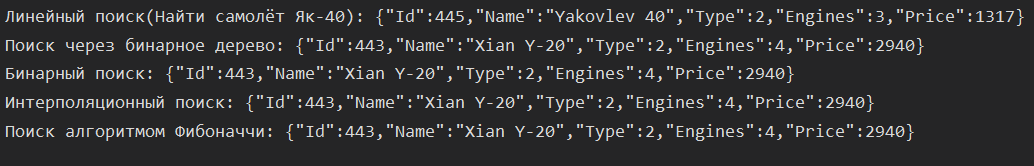
Выполнил: Чобану Артём

Группа: i1902

Кишинев 2021

**Задание:**  
Реализовать следующие алгоритмы поиска:

1. Линейный
2. Алгоритм поиска с помощью бинарного дерева
3. Двоичный
4. Интерполяционный
5. Фобиначчи

Результат работы программы:

Работа была выполнена на языке C# 9.0

Исходный код можно также получить по ссылке:

<https://github.com/ArtiomCiobanu/Algorithms_Labs>

1. **Алгоритм линейного поиска. Код программы:**

public static T FindWhere<T>(this IEnumerable<T> collection, Predicate<T> condition)  
 where T : class  
{  
 foreach (var item in collection)  
 {  
 if (condition(item))  
 {  
 return item;  
 }  
 }  
  
 return null;  
}  
  
public static Aircraft FindByName(this IEnumerable<Aircraft> aircrafts, string name)  
 => aircrafts.FindWhere(a => a.Name == name);

Использование:

//Линейный поиск  
//Console.WriteLine(aircraftById);  
Console.WriteLine($"Линейный поиск(Найти самолёт Як-40): {yak40}");

1. **Поиск с использованием бинарного дерева:**

Класс BinaryTree:

|  |  |
| --- | --- |
| public class BinaryTree {  private readonly BinaryTreeNode \_rootNode;  public BinaryTree(Aircraft rootAircraft)  {  \_rootNode = new BinaryTreeNode(rootAircraft);  }  public void InsertNode(BinaryTreeNode node) {  var currentNode = \_rootNode;  while (true)  {  if (currentNode.Values  .Select(v => v.Id)  .Intersect(node.Values.Select(v => v.Id))  .Any())  {  return;  }  if (node.Index > currentNode.Index)  {  if (currentNode.Right == null)  {  currentNode.Right = node;  return;  }  currentNode = currentNode.Right;  }  else if (node.Index < currentNode.Index)  {  if (currentNode.Left == null)  {  currentNode.Left = node;  return;  }  currentNode = currentNode.Left;  }  else  { currentNode.AddRange(node.Values);  break;  }  } } | public void InsertValue(Aircraft value)  {  InsertNode(new BinaryTreeNode(value));  }   public IEnumerable<Aircraft> Find(int index)  {  var currentNode = \_rootNode;   while (currentNode.Index != index)  {  if (index > currentNode.Index)  {  if (currentNode.Right != null)  {  currentNode = currentNode.Right;  }  }  else if (index < currentNode.Index)  {  if (currentNode.Left != null)  {  currentNode = currentNode.Left;  }  }  }   return currentNode.Values;  } } |

Класс BinaryTreeNode:

|  |  |
| --- | --- |
| public class BinaryTreeNode {  public int Index { get; }  public IList<Aircraft> Values { get; }  public BinaryTreeNode Left { get; set;}  public BinaryTreeNode Right { get;set;}   public BinaryTreeNode(Aircraft value)  {  Index = value.Price;  Values = new List<Aircraft>  {  value  };  }  public void AddValue(Aircraft value) {  if (value.Price == Index)  {  Values.Add(value);  }  else  {  throw new ArgumentException("Value has incorrect index");  } } | public void AddRange(IEnumerable<Aircraft> values)  {  foreach (var value in values)  {  AddValue(value);  }  }   public override string ToString()  => Values.Aggregate("",  (current, value) =>  current + value + "\n");  } } |

Использование:

/Поиск через бинарное дерево  
var tree = new BinaryTree(aircrafts.First());  
foreach (var aircraft in aircrafts)  
{  
 tree.InsertValue(aircraft);  
}  
  
var result = tree.Find(2940).ToArray();  
  
Console.WriteLine($"Поиск через бинарное дерево: {result.FirstOrDefault()}");

1. **Двоичный поиск**

public static Aircraft BinarySearchPrice(this Aircraft[] collection, int price)  
{  
 int leftIndex = 0;  
 int rightIndex = collection.Length - 1;  
  
 while (collection[leftIndex].Price != price)  
 {  
 var middle = (int) Math.Round((leftIndex + rightIndex) / 2.0);  
  
 if (collection[middle].Price > price)  
 {  
 rightIndex = middle;  
 }  
 else  
 {  
 leftIndex = middle;  
 }  
 }  
  
 return collection[leftIndex];  
}

Использование:

var binarySearchAircraft = sortedAircrafts.BinarySearchPrice(2940);

1. **Интерполяционный:**

public static Aircraft InterpolationSearch(

this Aircraft[] collection, int price)  
{  
 int leftIndex = 0;  
 int rightIndex = collection.Length - 1;  
  
 double leftValue = collection[leftIndex].Price;  
 double rightValue = collection[rightIndex].Price;  
  
 while (leftIndex < rightIndex &&  
 leftValue <= price && price <= rightValue)  
 {  
 int m = (int) Math.Ceiling(leftIndex +  
 (price - leftValue) \* (rightIndex - leftIndex) /  
 (rightValue - leftValue));  
  
 if (price > collection[m].Price)  
 {  
 leftIndex = m + 1;  
 }  
 else if (price < collection[m].Price)  
 {  
 rightIndex = m + 1;  
 }  
 else  
 {  
 return collection[m];  
 }  
  
 leftValue = collection[leftIndex].Price;  
 rightValue = collection[rightIndex].Price;  
 }  
  
 return collection[leftIndex];  
}

Использование:

var interpolationAircraft = sortedAircrafts.InterpolationSearch(2940);

1. **Фобиначчи:**
2. public static Aircraft FibonacciSearch(this Aircraft[] collection, int price)  
   {  
    int length = collection.Length;  
     
    int eliminated = -1;  
     
    int left = 0;  
    int middle = 1;  
    int right = middle;  
     
    while (right < length)  
    {  
    left = middle;  
    middle = right;  
    right = left + middle;  
    }  
     
    while (middle > 1 && eliminated + left < length)  
    {  
    int index = eliminated + left;  
     
    if (collection[index].Price < price)  
    {  
    right = middle;  
    middle = left;  
    left = right - middle;  
    eliminated = index;  
    }  
    else if (collection[index].Price > price)  
    {  
    right = left;  
    middle -= left;  
    left = right - middle;  
    }  
    else  
    {  
    return collection[index];  
    }  
    }  
     
    return null;  
   }

Использование:

var fibonacciAircraft = sortedAircrafts.FibonacciSearch(2940);

Алгоритмы 3-5 можно использовать с отсортированными коллекциями.