ОТЧЕТ

по лабораторной работе № 6

«Анализ алгоритмов сортировки»

по дисциплине

«ОСНОВЫ ТЕОРИИ АЛГОРИТМОВ»

Выполнил студент группы ИС/б-22о

Горбенко К.Н.

Проверил:

Карлусов В.Ю.

* 1. ЦЕЛЬ РАБОТЫ

Научиться оценивать сложность и количество операций для алгоритмов сортировки.

* 1. ВАРИАНТ ЗАДАНИЯ

1. Для алгоритмов сортировки вставками, прямым выбором, пузырьком, слиянием, Шелла, пирамидальной сортировки найти оценку для количества шагов и количества требуемой памяти.
2. Создать структуру и реализовать алгоритм сортировки согласно варианту задания. Критерий сортировки – по возрастанию стоимости.
3. Реализовать более эффективные алгоритмы согласно варианту задания.
4. Сравнить производительность различных алгоритмов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | Простейший алгоритм 1 | Простейший алгоритм 2 | Быстрый алгоритм 3 | Структура и критерий сортировки |
| 17 | Метод «Пузырька» | Сортировка вставками | «Быстрая» сортировка | Структура «Товар», содержит название, стоимость, и код, сортировка по возрастанию стоимости |

* 1. СРАВНЕНИЕ АЛГОРИТМОВ СОРТИРОВКИ
     1. Сортировка вставками

Сложность алгоритма составляет .

Требуемое количество памяти - .

* + 1. Сортировка прямым выбором

Сложность алгоритма составляет .

Требуемое количество памяти - .

* + 1. Сортировка пузырьком

Сложность алгоритма составляет .

Требуемое количество памяти - .

* + 1. Сортировка слиянием

Сложность алгоритма составляет .

Требуемое количество памяти - .

* + 1. Быстрая сортировка

Сложность алгоритма составляет .

Требуемое количество памяти - .

* + 1. Сортировка Шелла

Сложность алгоритма составляет .

Требуемое количество памяти - .

* + 1. Пирамидальная сортировка

Сложность алгоритма составляет .

Требуемое количество памяти - .

* 1. ТЕКСТЫ ПРОГРАММ

static void Main(string[] args)

{

var random = new Random();

var array = new Goods[20].Select(x => GoodsGenerator.GetGoodsItem(random)).ToArray();

foreach (var item in array)

Console.Write($"{item} ");

var timerStart = DateTime.Now;

array = SortingAlgorithms.InsertionSort(array);

var timerEnd = DateTime.Now;

Console.WriteLine();

foreach (var item in array)

Console.Write($"{item} ");

Console.WriteLine($"\n{timerEnd - timerStart}");

Console.ReadKey();

}

public static class SortingAlgorithms

{

public static Goods[] InsertionSort(Goods[] sequence)

{

var N = sequence.Length;

for (int i = 0; i < N; i++)

{

Goods x = sequence[i];

int j;

for (j = i - 1; j >= 0 && sequence[j].CompareTo(x) > 0; j--)

sequence[j + 1] = sequence[j];

sequence[j + 1] = x;

}

return sequence;

}

public static Goods[] BubleSort(Goods[] sequence)

{

var N = sequence.Length;

for (int i = 0; i < N - 1; i++)

for (int j = i + 1; j < N; j++)

if (sequence[i].CompareTo(sequence[j]) > 0)

{

var temp = sequence[i];

sequence[i] = sequence[j];

sequence[j] = temp;

}

return sequence;

}

public static void QuickSort(ref Goods[] arr, int first, int last)

{

Goods p = arr[(last - first) / 2 + first];

int i = first, j = last;

while (i <= j)

{

while (arr[i].CompareTo(p) < 0) ++i;

while (arr[j].CompareTo(p) > 0) --j;

if (i <= j)

{

Goods temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

++i; --j;

}

}

if (j > first) QuickSort(ref arr, first, j);

if (i < last) QuickSort(ref arr, i, last);

}

}

public struct Goods : IComparable

{

public string Name { get; set; }

public int Price { get; set; }

public int ID { get; set; }

public Goods(string name, int price, int id)

{

Name = name;

Price = price;

ID = id;

}

public int CompareTo(object obj)

{

var otherGoods = (Goods)obj;

if (this.Price == otherGoods.Price) return 0;

if (this.Price > otherGoods.Price) return 1;

return -1;

}

public override string ToString()

=> Price.ToString();

}

public static class GoodsGenerator

{

public static Goods GetGoodsItem(Random random)

=> new Goods { Name = "Name",

Price = random.Next(100),

ID = random.Next(50) };

}

* 1. РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММ

Результат работы программы, выполняющей сортировку методом вставки приведен на рисунке № 1:

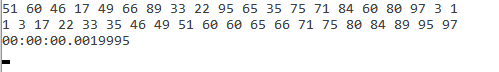


Рис. 1 – Результат сортировки методом вставки

Результат работы программы, выполняющей сортировку методом пузырька приведен на рисунке № 2:



Рис. 2 – Результат сортировки методом пузырька

Результат работы программы, выполняющей сортировку методом быстрой сортировки, приведен на рисунке № 3:



Рис. 3 – Результат сортировки методом быстрой сортировки

* 1. ВЫВОД

В ходе лабораторной работы были оценены сложности и оценки памяти следующих алгоритмов сортировки: вставки, прямым обменом, пузырьком, слиянием, сортировки Шелла, пирамидальной сортировки, быстрой сортировки.

Исходя из оценок сложности и требуемого количества памяти, наиболее эффективным алгоритмом оказался алгоритм пирамидальной сортировки.

Из алгоритмов, оцененных практически, наиболее эффективным оказался алгоритм быстрой сортировки, что подтверждает теоретическую оценку. Однако, для небольшого объема входных данных разница во времени выполнения составляет менее миллисекунды, что позволяет с одинаковой эффективностью использовать данные виды алгоритмов.