Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

Алгоритмы сортировки

Вариант № 7

Преподаватель \_\_\_\_\_\_\_\_\_\_ И. В. Матковский

подпись, дата

Студент КИ20-08Б 032049025 \_\_\_\_\_\_\_\_\_\_ А. Н. Головань

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2021

**1 Цель работы**

Изучить алгоритмы сортировки.

**2 Порядок выполнения работы**

Разработать программы, демонстрирующие два заданных метода сортировки; сравнить их эффективность

**3 Задание**

|  |  |  |
| --- | --- | --- |
| *№* | *Первый метод* | *Второй метод* |
| 7 | Сортировка пузырьком | Быстрая сортировка |

**4 Код программы**

P5.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92 | import os  from random import randint  from time import time  import random  count = 1000  os.system("cls")  a = []  for i in range(count):  a.append(randint(-100,100))  def bubble(b):  a = b.copy()  for i in range(len(a)):  for j in range(i):  if a[i] < a[j]:  a[i],a[j] = a[j],a[i]  return a  def quicksort(b):  a = b.copy()  if len(a) <= 1:  return a  else:  q = random.choice(a)  s\_nums = []  m\_nums = []  e\_nums = []  for n in a:  if n < q:  s\_nums.append(n)  elif n > q:  m\_nums.append(n)  else:  e\_nums.append(n)  return quicksort(s\_nums) + e\_nums + quicksort(m\_nums)  def heapify(nums, heap\_size, root\_index):  # Индекс наибольшего элемента считается корневым индексом  largest = root\_index  left\_child = (2 \* root\_index) + 1  right\_child = (2 \* root\_index) + 2  # Если левый потомок корня — это допустимый индекс, а элемент больше,  # чем текущий наибольший, то мы обновляем наибольший элемент  if left\_child < heap\_size and nums[left\_child] > nums[largest]:  largest = left\_child  # То же самое и для правого потомка корня  if right\_child < heap\_size and nums[right\_child] > nums[largest]:  largest = right\_child  # Если наибольший элемент уже не корневой, они меняются местами  if largest != root\_index:  nums[root\_index], nums[largest] = nums[largest], nums[root\_index]  heapify(nums, heap\_size, largest)  def heap\_sort(b):  a = b.copy()  n = len(a)  # Создаём Max Heap из списка  # 2-й аргумент означает остановку алгоритма перед элементом -1, то есть  # перед первым элементом списка  # 3-й аргумент означает повторный проход по списку в обратном направлении,  # уменьшая счётчик i на 1  for i in range(n, -1, -1):  heapify(a, n, i)  # Перемещаем корень Max Heap в самый конец списка  for i in range(n - 1, 0, -1):  a[i], a[0] = a[0], a[i]  heapify(a, i, 0)    return a  start\_time = time()  sort = heap\_sort(a)  end\_time = time()  print("list sort by heap\_sort for {} sec".format(end\_time-start\_time))  start\_time = time()  sort = bubble(a)  end\_time = time()  print("list sort by bubble for {} sec".format(end\_time-start\_time))  start\_time = time()  sort = quicksort(a)  end\_time = time()  print("list sort by quicksort for {} sec".format(end\_time-start\_time)) |

**5 Теоретические оценки сложности алгоритмов**

В программе используются два алгоритма Сортировка. Первый из них – сортировка пузырьком, его сложность составляет O(n2)в среднем и худшем случае, O(n) в лучшем случае. В качестве второго алгоритма сортировки используется быстрая сортировка составит O(n\*log( n )), в среднем – O(n\*log( n )), в худшем – O(n2)

**6 Экспериментальные оценки сложности алгоритмов**

Таблица 1 - зависимость времени от объема входных данных

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | 1000 | 10000 | 100000 |
| Пузырьком | 0.027c | 2.364c | 17.265 с |
| Быстрая | 0.001 | 0.008c | 0.062c |