Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 17

По дисциплине «Основы алгоритмизации и программирования»

На тему «Анализ алгоритмов сортировок»

Выполнила:

Студентка 1 курса 2 группы

Глухова Д.В.

Преподаватель: асс. Андронова М.В.

2023, Минск

4. В соответствии со своим вариантом написать программу для сортировок массивов указанными в таблице методами. Исходные массивы заполняются случайными числами.

Определить зависимость времени выполнения алгоритмов от количества элементов для каждого из алгоритмов. Выполнить моделирование для массивов размером 1000, 2000, 3000, 4000, 5000 (в зависимости от быстродействия компьютера размеры массивов можно увеличивать).

Произвести сравнение эффективности алгоритмов (построить график в приложении Excel).

**Вариант 2.**

Шейкерная сортировка, пирамидальная сортировка, сортировка слиянием

|  |
| --- |
| Код программы |
| #include <stdlib.h>  #include <iostream>  using namespace std;  void swapEl(int\* arr, int i) // функция для свапа элементов  {  int buff;  buff = arr[i];  arr[i] = arr[i - 1];  arr[i - 1] = buff;  }  void Shaker(int\* arr, int size)  {  int leftMark = 1; // левая граница  int rightMark = size - 1; // правая граница  while (leftMark <= rightMark) // пока не пройдем  {  for (int i = rightMark; i >= leftMark; i--)  if (arr[i - 1] > arr[i]) swapEl(arr, i); // свапаем  leftMark++; // смещаем границу  for (int i = leftMark; i <= rightMark; i++)  if (arr[i - 1] > arr[i]) swapEl(arr, i); // тоже свапаем  rightMark--; // смещаем границу  }  }  void heapify(int arr[], int n, int root)// функция heapify для нагромождения дерева  {  int largest = root; // корень - самый большой элемент  int l = 2 \* root + 1; // слева = 2\*корень + 1  int r = 2 \* root + 2; // справа = 2\*корень + 2  if (l < n && arr[l] > arr[largest])// Если левый дочерний элемент больше корневого  largest = l;  if (r < n && arr[r] > arr[largest])// Если правый дочерний элемент больше, чем самый большой на данный момент  largest = r;  if (largest != root)// Если самый большой не является корневым  {  swap(arr[root], arr[largest]);//поменять местами корневой и самый большой  heapify(arr, n, largest);// Рекурсивное нагромождение поддерева  }  }  void heapSort(int arr[], int n)// реализация сортировки кучи  {  for (int i = n / 2 - 1; i >= 0; i--)// создать кучу  heapify(arr, n, i);  for (int i = n - 1; i >= 0; i--)// извлечение элементов из кучи один за другим  {  swap(arr[0], arr[i]);// Переместить текущий корень в конец  heapify(arr, i, 0);// снова вызовите max heapify для уменьшенной кучи  }  }  void insOrd(int m[], int sm, int em, int e)  {  int buf;  int i = sm;  while (i <= em && m[i] < e) // пока не конец  {  if (i - 1 >= sm)  m[i - 1] = m[i];  i++;  } m[i - 1] = e;  }  int\* merge(int m[], int sm, int cm, int em)  {  for (int i = 0; i <= sm; i++)  {  if (m[i] > m[cm + 1])  {  int buf = m[i];  m[i] = m[cm + 1];  insOrd(m, cm + 1, em, buf);  }  }  return m;  }  int\* sortMerge(int m[], int lm, int sm = 0)//рекурсивная процедура сортировки  {  if (lm > 1)  {  sortMerge(m, lm / 2, sm);//сортировка левой части  sortMerge(m, lm - lm / 2, sm + lm / 2); //сортировка правой части  merge(m, sm, sm + lm / 2 - 1, sm + lm - 1);//слияние двух частей  };  return m;  }  int GetRandKey(int reg = 0)  {  if (reg > 0) srand((unsigned)time(NULL));  int rmin = 0;  int rmax = 100000;  return (int)(((double)rand() / (double)RAND\_MAX) \* (rmax - rmin) + rmin);  }  int main()  {  setlocale(LC\_CTYPE, "rus");  const int N = 50000; int k[N], f[N];  clock\_t t1, t2;  GetRandKey(1);  for (int i = 0; i < N; i++) f[i] = GetRandKey(0);  for (int n = 10000; n <= N; n += 10000)  {  cout << "n = " << n << endl;  cout << "Шейкер: ";  memcpy(k, f, n \* sizeof(int));  t1 = clock();  Shaker(k, n);  t2 = clock();  cout << "Прошло " << t2 - t1 << " тактов времени" << endl;  cout << "Слиянием: ";  memcpy(k, f, n \* sizeof(int));  t1 = clock();  sortMerge(k, n);  t2 = clock();  cout << "Прошло " << t2 - t1 << " тактов времени" << endl;  cout << "Пирамидальная: ";  memcpy(k, f, n \* sizeof(int));  t1 = clock();  heapSort(k, n);  t2 = clock();  cout << "Прошло " << t2 - t1 << " тактов времени" << endl << endl;  }  return 0;  } |
| Результат в консоли |
|  |

Дополнительные задания.

2. Дан массив целых чисел, количество элементов которого надо ввести с клавиатуры. Найти максимальный элемент массива и его номер, при условии, что все элементы различны. Найти минимальный элемент массива.

|  |
| --- |
| Код программы |
| #include <iostream>  using namespace std;  void insertSort(int\* A, int size);  void main() {  const int max = 1000;  setlocale(LC\_ALL, "rus");  int B[max]; int maxB = 0, minB = 0, size, maxNum;  srand(time(NULL));  cout << "Введите размер массива: ";  cin >> size;  cout << "Массив В: ";  for (int i = 0; i < size; i++)  {  B[i] = -500 + rand() % 1000;  cout << B[i] << " ";  }  cout << endl;  for (int i = 0; i < size; i++)  {  if (B[i] > maxB) {  maxB = B[i];  maxNum = i + 1;  }  if (B[i] < minB)  minB = B[i];  }  cout << endl;  cout << "Максимальный элемент: " << maxB << " с индексом " << maxNum;  cout << endl;  cout << "Минимальный элемент: " << minB;  cout << endl;  insertSort(B, size);  cout << "Отсортированный массив: ";  for (int i = 0; i < size; i++)  {  cout << B[i] << " ";  }  cout << endl;  }  void insertSort(int\* A, int size)  {  int t, i, j;  for (i = 1; i < size; i++)  {  t = A[i];  for (j = i - 1; j >= 0 && A[j] > t; j--)  A[j + 1] = A[j];  A[j + 1] = t;  }  } |
| Результат в консоли |
|  |

3. Дан массив из 10 элементов. Первые 4 элемента упорядочить по возрастанию, последние 4 по убыванию.

|  |
| --- |
| Код программы |
| #include <iostream>  using namespace std;  void insertionSort(int arr[], int n) { //сортировка простой вставкой  int i, key, j;  for (i = 1; i < n; i++) {  key = arr[i]; //сохранить  j = i - 1; //уменьшение размера унупорядоченной части  while (j >= 0 && arr[j] > key) { //поиск места вставки  arr[j + 1] = arr[j]; //сдвиг элемента  j = j - 1;  }  arr[j + 1] = key; //вставка очередного на место первого большего его  }  }  void main() {  setlocale(LC\_ALL, "rus");  int A[10], B[4], maxB = 0;  srand(time(NULL));  cout << "Изначальный массив: ";  for (int i = 0; i < 10; i++)  {  A[i] = 10 + rand() % 40;  cout << A[i] << " ";  }  insertionSort(A, 4);  cout << "\nПервые 4 элемента по возрастанию: ";  for (int i = 0; i < 10; i++)  {  cout << A[i] << " ";  }  cout << endl;  for (int i = 9, k = 0; i > 5; i--, k++)  {  B[k] = A[i];  }  insertionSort(B, 4);  cout << "Последние 4 элемента по убыванию: ";  for (int i = 9, k = 0; i > 5; i--, k++)  {  A[i] = B[k];  }  for (int i = 0; i < 10; i++)  {  cout << A[i] << " ";  }  cout << endl;  } |
| Результат в консоли |
|  |

4. Дан массив из 15 целых чисел на отрезке [-5; 5]. Упорядочить массив, удалив повторяющиеся элементы.

|  |
| --- |
| Код программы |
| #include <iostream>  using namespace std;  void insertionSort(int arr[], int n)  { //сортировка простой вставкой  int i, key, j;  for (i = 1; i < n; i++) {  key = arr[i]; //сохранить  j = i - 1; //уменьшение размера унупорядоченной части  while (j >= 0 && arr[j] > key) { //поиск места вставки  arr[j + 1] = arr[j]; //сдвиг элемента  j = j - 1;  }  arr[j + 1] = key; //вставка очередного на место первого большего его  }  }  void main()  {  setlocale(LC\_ALL, "rus");  int A[15], B[15] = { -5,-4,-3,-2,-1,0,1,2,3,4,5 }, flag = 0, howmuch = 15;  srand(time(NULL));  cout << "Изначальный массив: ";  for (int i = 0; i < 15; i++)  {  A[i] = -5 + rand() % 10;  cout << A[i] << " ";  }  cout << endl;  for (int i = 0; i < 15; i++)  {  for (int k = 0; k < 15; k++)  {  if (B[i] == A[k]) {  flag++;  if (flag > 1) {  A[k] = 6;  howmuch--;  }  }  }  flag = 0;  }  cout << "\nМассив с удаленными элементами: ";  for (int i = 0; i < 15; i++)  {  if (A[i] != 6)  cout << A[i] << " ";  }  cout << endl;  insertionSort(A, 15);  cout << "\nОтсортированный массив: ";  for (int i = 0; i < howmuch; i++)  {  cout << A[i] << " ";  }  } |
| Результат в консоли |
|  |