Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

Практическая работа №3
по дисциплине «Структуры и организация данных»
на тему «Оценка эффективности алгоритмов»

Выполнил:
Студент Альков В.С.
Группа И407Б

Преподаватель:
Полухин А.Л.

Санкт-Петербург
2021 г.

Задача 1.

Уровень сложности – повышенный. Провести сравнение указанных алгоритмов сортировки массивов, содержащих *N1*, *N2*, *N3* и *N4* элементов, по указанному в вариативной части критерию. Каждую функцию сортировки вызывать трижды: для сортировки неупорядоченного массива, упорядоченного массива и массива, упорядоченного в обратном порядке. При работе каждого алгоритма сортировки выполнить подсчет основных (производимых над элементами массива) и вспомогательных (всех остальных) операций, указанных в вариативной части задания (сравнений или присваиваний), а также зафиксировать время работы алгоритма. Сортируемая последовательность для всех методов должна быть одинаковой (считывать необходимое количество элементов из прилагаемого файла *test_numbers.txt*). Оценить время работы и эффективность алгоритмов сортировки по заданному критерию и объему требуемой дополнительно памяти.

Дополнительно провести анализ того, как наличие повторяющихся ключей во входной последовательности влияет на трудоемкость каждого из рассматриваемых алгоритмов сортировки.

Порядок: по возрастанию элементов. Методы: выбора, пузырька, пирамидальная сортировка, быстрая сортировка. N1=10000, N2=30000, N3=70000, N4=100000.

Критерий – количество присваиваний.

1. Алгоритм сортировки методом выбора:

на каждом шаге ищется минимальный элемент ещё не отсортированной части массива и ставится на место первого элемента этой части массива.

Трудоемкость сортировки методом выбора по количеству присваиваний:

Случай	**Ситуация, соответст-вующая случаю**	**Обоснование**	**Ожидаемое число присваиваний элементов массива**	**Асимптотическая оценка сложности по количеству присваиваний**	**Ожидаемое число вспомогат. присваиваний**
наилучший	массив упорядочен	Элементы не требуют перестановок, min не будет переприсваиваться	3(n-1) N1: 29997 N2: 89997 N3: 209997 N4: 299997	Ω(n)	3n + n(n-1)/2 N1: 50025000 N2: 450075000 N3: 2450175000 N4: 5000250000
наихудший	массив упорядочен в обратном порядке	Кол-во вспомогательных присваиваний будет больше, чем везде, так как min обновляется чаще чем в других случаях	3(n-1) N1: 29997 N2: 89997 N3: 209997 N4: 299997	Ω(n)	3n + n *(n -1)/2 + n * n /4 N1: 75025000 N2: 675075000 N3: 3675175000 N4: 7500250000

средний	неупорядо-ченный массив	Среднее кол-во дополнительных присваиваний в силу неизвестности	3(n-1) N1: 29997 N2: 89997 N3: 209997 N4: 299997	Ω(n)	3n + n * (n -1)/2 + n * n /8N1: 62525000N2: 562575000N3: 3062675000N4: 6250250000

Пространственная сложность – O(1) (две переменных цикла и две вспомогательных переменных).

Сортировка выбором является неустойчивой.

При повторениях ключей количество присваиваний переменной цикла с целью поиска минимального элемента останется неизменным, но кол-во присваиваний минимального элемента уменьшиться, так как сравнение строгое, а кол-во уникальных ключей уменьшилось.

2. Алгоритм сортировки пузырьком:

Алгоритм заключается в повторяющихся проходах по массиву, во время прохода сравниваются соседние элементы, и если первый больше второго, то элементы меняются местами. За один проход как минимум один элемент встает на свое место.

Трудоемкость пузырьковой сортировки по количеству присваиваний:

Случай	**Ситуация, соответствующая случаю**	**Обоснование**	**Ожидаемое число присваиваний элементов массива**	**Асимптотическая оценка сложности по количеству присваиваний**	**Ожидаемое число вспомогат. присваиваний**
наилучший	массив упорядочен	Элементы не требуют перестановок, так как первый элемент всегда меньше второго, и условие 1ый>2ого не выполняется	0 N1: 0 N2: 0 N3: 0 N4: 0	Ω(1)	n*(n + 1) /2 + n N1: 50015000 N2: 450045000 N3: 2450105000 N4: 5000150000
наихудший	массив упорядочен в обратном порядке	Будет произведено максимальное кол-во присваиваний, так как условие 1ый>2ого всегда выполняется	3 * n * (n -1) / 2 N1: 149985000 N2: 1349955000 N3: 7349895000 N4:14999850000	$O(n^2)$	n*(n + 1) /2 + n N1: 50015000 N2: 450045000 N3: 2450105000 N4:5000150000
средний	массив неупорядочен	Среднее кол-во дополнительных присваиваний в силу неизвестности	3 * n * (n -1) / 4 N1: 74992500 N2: 674977500 N3: 3674947500 N4: 7499925000	$O(n^2)$	n*(n + 1) /2 + n N1: 50015000 N2: 450045000 N3: 2450105000 N4: 5000150000

Пространственная сложность – O(1) (две переменных цикла и одна вспомогательная переменная).

Пузырьковая сортировка является устойчивой.

Повторяющиеся значения ключей уменьшают трудоемкость алгоритма. Условие меньше будет выполнятся реже, так как некоторые соседние элементы будут равны, следовательно, уменьшится кол-во перестановок.

3. Алгоритм пирамидальной сортировки:

Метод сортировки сравнением, основанный на такой структуре данных как двоичная куча. Она похожа на сортировку выбором, где мы сначала ищем максимальный элемент и помещаем его в конец. Далее мы повторяем ту же операцию для оставшихся элементов.

Трудоемкость пирамидальной сортировки по количеству присваиваний:

Случай	**Ситуация, соответствующая случаю**	**Обоснование**	**Ожидаемое число присваиваний элементов массива**	**Асимптотическая оценка сложности по количеству присваиваний**	**Ожидаемое число вспомогат. присваиваний**
наилучший наихудший средний	Нельзя точно определить, когда производительность будет наилучшей, а когда наихудшей	Пирамида строится заново в независимости от степени сортированности массива	~3*n*logn	O(n*logn)	<5*n*logn

Пространственная сложность – O (1) (6 вспомогательных переменных).

Пирамидальная сортировка является неустойчивой.

Повторяющиеся значения ключей не влияют на трудоемкость алгоритма.

Алгоритм быстрой сортировки:

Поиск опорного элемента, перестановка остальных элементов на большие и меньшие, и рекурсивный запуск от получившихся подмассивов.

Трудоемкость быстрой сортировки по количеству присваиваний:

Случай	**Ситуация, соответст-вующая случаю**	**Обоснование**	**Ожидаемое число присваиваний элементов массива**	**Асимптотическая оценка сложности по количеству присваиваний**	**Ожидаемое число вспомогат. присваиваний**
наилучший	Когда при каждом выборе опорного элемента остальные числа разбиваются поровну с точностью до	максимальная глубина рекурсии, при которой размеры обрабатываемых подмассивов достигнут 1, составитlog(n)	<3*n N1: <30000 N2: <90000 N3: <210000 N4: <300000	O(n)	n*logn N1: 140000 N2: 450000 N3: 1190000 N4: 1700000

	единицы				
наихудший	Когда при каждом выборе опорного элемента остальные числа перемещаются в одну сторону от опорного	Требуется n - 1 операция разделения, и для каждого разделения нужно будет просмотреть все оставшиеся элементы	$3*n^2$ N1: 300000000 N2: 2700000000 N3: 14700000000 N4: 30000000000	$O(n^2)$	n^2 N1: 100000000 N2: 900000000 N3: 4900000000 N4: 10000000000
средний	При выборе опорного элемента разбивается примерно одинаково, т.е. в среднем 75% в одну, 25% в другую	Оценивается вероятностно	3*n*logn N1: 30000*14 N2: 90000*15 N3: 210000*16 N4: 300000*17	O(n*logn)	2*n*logn N1: 20000*14 N2: 60000*15 N3: 140000*16 N4: 200000*17

Пространственная сложность – O(logn)

Быстрая сортировка является неустойчивой.

При увеличении количества повторов ключей, производительность ухудшится, т.к. каждый элемент равный выбранному будет присутствовать либо справа, либо слева, а значит точно не улучшит время работы, а только ухудшит.

Текст программы:

```
#include <iostream>
#include <math.h>
#include <fstream>
#include <locale.h>
#include <chrono>
#include <cstring>
#define N1 10000
#define N2 30000
#define N3 70000
#define N4 100000

using namespace std;
using namespace chrono;
/*p1 - кол-во освновых присваиванй, p2 - кол-во вспомогательных, p3 - память*/
void StraightSelection (int *a, int n, unsigned long long *p1, unsigned long long *p2, int *p3);
void BubbleSort (int *a, int n, unsigned long long *p1, unsigned long long *p2, int *p3);
void HeapSort (int *a, int n, unsigned long long *p1, unsigned long long *p2, int *p3);
int sift (int *a, int L, int R);
void QuickSort (int *a, int n, unsigned long long *p1, unsigned long long *p2, int *p3);
```

```
      int* readArr(int n, char *filename);
      void reverse(int* a, int n);
      /*массив указателей на ф-ии, чтобы удобно было сортировки в цикле*/
      void (*f[4])(int *a, int n, unsigned long long *p1, unsigned long long
*p2, int *p3) = {StraightSelection, BubbleSort, HeapSort, QuickSort};

      void sort(int sortNum, char* filename)
      {
          /*заносим размеры массивом в массив для удобства вызова сортировок в
цикле*/
          int n[4] = {N1, N2, N3, N4};
          unsigned long long p1, p2;
          int p3, **a;
          char  c1[4][30]  =  {"Метод  выбора",  "Пузырек",  "Пирамидальная
сортировка", "Быстрая сортировка"};
          char  c2[3][30]  =  {"Неупорядоченный",  "Упорядоченный",  "В  обратном
порядке"};
          high_resolution_clock time = high_resolution_clock();
          high_resolution_clock::time_point t1;
          high_resolution_clock::time_point t2;
          duration<double> time_span;
          a = new int*[4];
          for(int i=0; i<4; i++)
              a[i] = readArr(n[i], filename);
          /*цикл от 0 до 3, потому что нужно отсортировать каждый массив 3 раза:
неупорядоченный, упорядоченный, в обратном порядке*/
          for(int i=0; i<3; i++)
          {
              /*цикл от 0 до 4, потому что нужно отсортировать 4 массива, которые
состоят соответстенно из N1, N2, N3, N4 элементов*/
              for(int j =0; j<4; j++)
              {
                  /*i,  равная  двум  сигнализирует,  что  массивы  уже  были
отсортированы дважды, и настал черед сортировать массивы в обратном порядке*/
                  if(i==2)
                      reverse(a[j], n[j]);
                  /*обнуляем  счетчики  основных,  вспомогательных  операций,  и
памяти*/
                  p1 = p2 = 0;
                  p3 = 0;
                  /*запоминаем время*/
                  t1 = time.now();
                  f[sortNum](a[j], n[j], &p1, &p2, &p3);
                  t2 = time.now();
                  time_span = duration_cast<duration<double>>(t2 - t1);
                  printf("%-20s|%6d|%-30s|%-11llu     +     %-11llu     =     %-
11llu|%10d|%10lf|\n",   c2[i],   n[j],   c1[sortNum],   p1,   p2,   p1+p2,   p3,
time_span.count());
              }
              printf("-------------------------------------------------------------
-----------------------------------------------------------------\n");
          }
          /*очищаем память*/
          for(int i =0; i<4; i++)
              delete [] a[i];
          delete [] a;
      }



      int main(int argc, char* files[])
      {
          char filename[80];
```

```
        setlocale(LC_ALL, "rus");
        if(argc<2)
        {
            cout << "Имя файла:";
            cin.getline(filename, 80);
        }
        else
            strcpy(filename,files[1]);
        printf("%-20s|%-6s|%-30s|%-33s|%-10s|%-10s|\n",    "Массив",    "N",
"Сортировка", "Кол-во основных присв. + вспомогательных", "Память", "Время");
        printf("-------------------------------------------------------------
-------------------------------------------------------------\n");
        for(int i=0; i<4; i++)
            sort(i, filename);
    }

    /*ф-ия чтения массива из файла, с выделением памяти*/
    int* readArr(int n, char *filename)
    {
        std::fstream f(filename);
        if(!f.is_open())
        {
            std::cout<<"Не удалось открыть файл";
            return 0;
        };
        int *a = new int[n];
        for(int i=0; i<n; i++)
            f>>a[i];
        f.close();
        return a;
    }

    /*ф-ия записи элементов массива в обратном порядке*/
    void reverse(int* a, int n)
    {
        int tmp;
        for(int i=0; i<n/2; i++)
        {
            tmp = a[i];
            a[i] = a[n-i-1];
            a[n-i-1] = tmp;
        }
    }


    void StraightSelection (int *a, int n, unsigned long long *p1, unsigned
long long *p2, int *p3)
    {
          int i, j, tmp, *min;
        *p3 += 3*sizeof(int)+sizeof(int*);
          for (i=0, *p2 += 1; i<n-1; i++, *p2+=1)
          {
                min = a+i;
                *p2+=1;
                for (j=i+1, *p2 += 1; j<n; j++, *p2+=1)
                      if (a[j]<*min)
                {
                    min = a+j;
                    *p2+=1;
                };
            tmp = *min;
            *min = a[i];
            a[i] = tmp;
```

```
            *p1 += 3;
          }
      }


      void BubbleSort (int *a, int n, unsigned long long *p1, unsigned long long
*p2, int *p3)
      {
            int i, j, x;
          *p3 += 3*sizeof(int);
            for (i=1, *p2 += 1; i<n; i++, *p2+=1)
            {

                  for (j=n-1, *p2+=1; j>=i; j--, *p2+=1)
              {
                  if (a[j-1]>a[j])
                        {
                              x = a[j-1];
                              a[j-1] = a[j];
                              a[j] = x;
                              *p1+=3;
                        }
              }
          }
      }

      void sift (int *a, int L, int R, unsigned long long *p1, unsigned long
long *p2, int *p3)
      {
            int i = L, j = 2*L+1, x = a[L];
          *p1+=1;
          *p2+=2;
            if (j<R && a[j+1]>a[j])
              j++,*p2+=1;
            while (j<=R && a[j]>x)
            {
                  a[i] = a[j];
                  i = j;
                  j = 2*j + 1;
                  *p1+=1;
                  *p2+=2;
                  if (j<R && a[j+1]>a[j])
                  j++,*p2+=1;
            }
            a[i] = x;
            *p1+=1;
      }


      void HeapSort (int *a, int n, unsigned long long *p1, unsigned long long
*p2, int *p3)
      {
            int L = n/2, R=n-1, x;
          *p3 += 6*sizeof(int);
            *p2 += 2;
            while (L>0)
          {
              sift (a, --L, R, p1, p2, p3);
              *p2+=4;
          }
            while (R>0)
            {
                  x = a[0];
```

```
                a[0] = a[R];
                a[R] = x;
              sift (a, L, --R, p1, p2, p3);
              *p2+=4;
                *p1+=3;
            }
      }


      void QuickSort (int *a, int n,  unsigned long long *p1, unsigned long long
*p2, int *p3)
      {
            int x, w, i, j;
          *p3+=5*sizeof(int) + sizeof(int*);
          x = a[n/2];
            i=0; j=n-1;
            *p1+=1;
            *p2+=2;
            do
            {
                  while (a[i]<x) i++, *p2+=1;
                  while (x<a[j]) j--, *p2+=1;
                  if (i<=j)
                  {
                        w = a[i]; a[i] = a[j]; a[j] = w;
                        i++; j--;
                        *p1+=3;
                        *p2+=2;
                  }
            }
            while (i<j);
            if (j>0)
                  QuickSort (a, j+1, p1, p2, p3), *p2+=2;
            if (i<n-1)
                  QuickSort (a+i, n-i, p1, p2, p3), *p2+=2;
      }
```

Результаты работы программы:

```
Имя файла:test_numbers.txt
Массив              |N     |Сортировка                  |Кол-во основых присв. + вспомогательных|Память    |Время
-------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Метод выбора                |29997       + 50103589    = 50133586   |        20|  0,184895
Неупорядоченный     | 30000|Метод выбора                |89997       + 450339394   = 450429391  |        20|  1,641913
Неупорядоченный     | 70000|Метод выбора                |209997      + 2450849916  = 2451059913 |        20|  8,874496
Неупорядоченный     |100000|Метод выбора                |299997      + 5001249981  = 5001549978 |        20| 26,414737
-------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Метод выбора                |29997       + 50024998    = 50054995   |        20|  0,347676
Упорядоченный       | 30000|Метод выбора                |89997       + 450074998   = 450164995  |        20|  4,140805
Упорядоченный       | 70000|Метод выбора                |209997      + 2450174998  = 2450384995 |        20| 16,240465
Упорядоченный       |100000|Метод выбора                |299997      + 5000249998  = 5000549995 |        20| 27,401037
-------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Метод выбора                |29997       + 75024998    = 75054995   |        20|  0,402995
В обратном порядке  | 30000|Метод выбора                |89997       + 675074998   = 675164995  |        20|  4,043925
В обратном порядке  | 70000|Метод выбора                |209997      + 3675174998  = 3675384995 |        20| 18,987783
В обратном порядке  |100000|Метод выбора                |299997      + 7500249998  = 7500549995 |        20| 30,986364
-------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пузырек                     |74085783    + 50014999    = 124100782  |        12|  0,633835
Неупорядоченный     | 30000|Пузырек                     |677364852   + 450044999   = 1127409851 |        12|  6,303233
Неупорядоченный     | 70000|Пузырек                     |3644710836  + 2450104999  = 6094815835 |        12| 42,495278
Неупорядоченный     |100000|Пузырек                     |7362556977  + 5000149999  = 12362706976|        12| 68,908123
-------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пузырек                     |0           + 50014999    = 50014999   |        12|  0,231773
Упорядоченный       | 30000|Пузырек                     |0           + 450044999   = 450044999  |        12|  2,136692
Упорядоченный       | 70000|Пузырек                     |0           + 2450104999  = 2450104999 |        12| 11,600992
Упорядоченный       |100000|Пузырек                     |0           + 5000149999  = 5000149999 |        12| 23,822035
-------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пузырек                     |149985000   + 50014999    = 199999999  |        12|  0,563791
В обратном порядке  | 30000|Пузырек                     |1349955000  + 450044999   = 1799999999 |        12|  5,096563
В обратном порядке  | 70000|Пузырек                     |7349895000  + 2450104999  = 9799999999 |        12| 27,988682
В обратном порядке  |100000|Пузырек                     |14999850000 + 5000149999  = 19999999999|        12| 57,011809
-------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пирамидальная сортировка    |174293      + 375588      = 549881     |        24|  0,000000
Неупорядоченный     | 30000|Пирамидальная сортировка    |569782      + 1244880     = 1814662    |        24|  0,015636
Неупорядоченный     | 70000|Пирамидальная сортировка    |1414598     + 3118440     = 4533038    |        24|  0,031250
Неупорядоченный     |100000|Пирамидальная сортировка    |2075517     + 4587647     = 6663164    |        24|  0,062517
-------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пирамидальная сортировка    |181952      + 393898      = 575850     |        24|  0,000000
Упорядоченный       | 30000|Пирамидальная сортировка    |590096      + 1295286     = 1885382    |        24|  0,010081
Упорядоченный       | 70000|Пирамидальная сортировка    |1464298     + 3241768     = 4706066    |        24|  0,020123
Упорядоченный       |100000|Пирамидальная сортировка    |2150850     + 4771757     = 6922607    |        24|  0,025638
-------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пирамидальная сортировка    |166692      + 354573      = 521265     |        24|  0,000000
В обратном порядке  | 30000|Пирамидальная сортировка    |549208      + 1187719     = 1736927    |        24|  0,015627
В обратном порядке  | 70000|Пирамидальная сортировка    |1362678     + 2971316     = 4333994    |        24|  0,015625
В обратном порядке  |100000|Пирамидальная сортировка    |1997430     + 4376726     = 6374156    |        24|  0,031249
-------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Быстрая сортировка          |114655      + 221224      = 335879     |    293356|  0,000000
Неупорядоченный     | 30000|Быстрая сортировка          |380049      + 695225      = 1075274    |    872844|  0,015625
Неупорядоченный     | 70000|Быстрая сортировка          |934420      + 1913128     = 2847548    |   2056600|  0,015627
Неупорядоченный     |100000|Быстрая сортировка          |1378439     + 2641877     = 4020316    |   2928800|  0,031256
-------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Быстрая сортировка          |23616       + 149053      = 172669     |    165312|  0,000000
Упорядоченный       | 30000|Быстрая сортировка          |65532       + 485544      = 551076     |    458724|  0,000000
Упорядоченный       | 70000|Быстрая сортировка          |148928      + 1212336     = 1361264    |   1042496|  0,000000
Упорядоченный       |100000|Быстрая сортировка          |262140      + 1862154     = 2124294    |   1834980|  0,015624
-------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Быстрая сортировка          |38617       + 149070      = 187687     |    165340|  0,000000
В обратном порядке  | 30000|Быстрая сортировка          |110529      + 485556      = 596085     |    458724|  0,015624
В обратном порядке  | 70000|Быстрая сортировка          |253929      + 1212356     = 1466285    |   1042524|  0,000000
В обратном порядке  |100000|Быстрая сортировка          |412137      + 1862168     = 2274305    |   1834980|  0,015633
-------------------------------------------------------------------------------------------------------------------


Process returned 0 (0x0)   execution time : 394.586 s
Press any key to continue.
```

Результаты указанных сортировок с повторяющимися элементами по 10 штук

```
Имя файла:10.txt
Массив              |N     |Сортировка                    |Кол-во основых присв. + вспомогательных|Память    |Время     |
-------------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Метод выбора                  |29997       + 50100030    = 50130027   |         20|  0,184900|
Неупорядоченный     | 30000|Метод выбора                  |89997       + 450323064   = 450413061  |         20|  1,619789|
Неупорядоченный     | 70000|Метод выбора                  |209997      + 2450771531  = 2450981528 |         20| 11,234374|
Неупорядоченный     |100000|Метод выбора                  |299997      + 5001108854  = 5001408851 |         20| 33,524712|
-------------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Метод выбора                  |29997       + 50024998    = 50054995   |         20|  0,216143|
Упорядоченный       | 30000|Метод выбора                  |89997       + 450074998   = 450164995  |         20|  1,958464|
Упорядоченный       | 70000|Метод выбора                  |209997      + 2450174998  = 2450384995 |         20| 10,110841|
Упорядоченный       |100000|Метод выбора                  |299997      + 5000249998  = 5000549995 |         20| 20,358886|
-------------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Метод выбора                  |29997       + 64117351    = 64147348   |         20|  0,432284|
В обратном порядке  | 30000|Метод выбора                  |89997       + 514958789   = 515048786  |         20|  3,108079|
В обратном порядке  | 70000|Метод выбора                  |209997      + 2625663471  = 2625873468 |         20| 12,861073|
В обратном порядке  |100000|Метод выбора                  |299997      + 5250249998  = 5250549995 |         20| 21,969317|
-------------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пузырек                       |74710923    + 50014999    = 124725922  |         12|  0,570304|
Неупорядоченный     | 30000|Пузырек                       |670648536   + 450044999   = 1120693535 |         12|  5,860628|
Неупорядоченный     | 70000|Пузырек                       |3664020696  + 2450104999  = 6114125695 |         12| 30,031490|
Неупорядоченный     |100000|Пузырек                       |7494437853  + 5000149999  = 12494587852|         12| 62,536288|
-------------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пузырек                       |0           + 50014999    = 50014999   |         12|  0,253915|
Упорядоченный       | 30000|Пузырек                       |0           + 450044999   = 450044999  |         12|  2,089807|
Упорядоченный       | 70000|Пузырек                       |0           + 2450104999  = 2450104999 |         12| 11,530961|
Упорядоченный       |100000|Пузырек                       |0           + 5000149999  = 5000149999 |         12| 23,479322|
-------------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пузырек                       |149963757   + 50014999    = 199978756  |         12|  0,579437|
В обратном порядке  | 30000|Пузырек                       |1349767194  + 450044999   = 1799812193 |         12|  4,950547|
В обратном порядке  | 70000|Пузырек                       |7349135283  + 2450104999  = 9799240282 |         12| 27,944223|
В обратном порядке  |100000|Пузырек                       |14998500000 + 5000149999  = 19998649999|         12| 56,912816|
-------------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пирамидальная сортировка      |174335      + 375469      = 549804     |         24|  0,000000|
Неупорядоченный     | 30000|Пирамидальная сортировка      |569697      + 1244559     = 1814256    |         24|  0,015625|
Неупорядоченный     | 70000|Пирамидальная сортировка      |1414702     + 3118178     = 4532880    |         24|  0,031257|
Неупорядоченный     |100000|Пирамидальная сортировка      |2074938     + 4586358     = 6661296    |         24|  0,031251|
-------------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пирамидальная сортировка      |181180      + 391332      = 572512     |         24|  0,000000|
Упорядоченный       | 30000|Пирамидальная сортировка      |590416      + 1290725     = 1881141    |         24|  0,000000|
Упорядоченный       | 70000|Пирамидальная сортировка      |1464208     + 3224582     = 4688790    |         24|  0,015624|
Упорядоченный       |100000|Пирамидальная сортировка      |2148925     + 4743394     = 6892319    |         24|  0,022134|
-------------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пирамидальная сортировка      |166617      + 354404      = 521021     |         24|  0,015626|
В обратном порядке  | 30000|Пирамидальная сортировка      |548975      + 1186957     = 1735932    |         24|  0,000000|
В обратном порядке  | 70000|Пирамидальная сортировка      |1362087     + 2969381     = 4331468    |         24|  0,022135|
В обратном порядке  |100000|Пирамидальная сортировка      |1996533     + 4373730     = 6370263    |         24|  0,015626|
-------------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Быстрая сортировка            |122007      + 218027      = 340034     |     318612|  0,015627|
Неупорядоченный     | 30000|Быстрая сортировка            |423075      + 736649      = 1159724    |    1025556|  0,006507|
Неупорядоченный     | 70000|Быстрая сортировка            |1103727     + 1789908     = 2893635    |    2513616|  0,015626|
Неупорядоченный     |100000|Быстрая сортировка            |1644696     + 2652925     = 4297621    |    3662400|  0,015625|
-------------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Быстрая сортировка            |38629       + 172540      = 211169     |     259756|  0,000000|
Упорядоченный       | 30000|Быстрая сортировка            |169761      + 618223      = 787984     |    1006404|  0,015632|
Упорядоченный       | 70000|Быстрая сортировка            |516122      + 1647792     = 2163914    |    2887472|  0,000000|
Упорядоченный       |100000|Быстрая сортировка            |819996      + 2486145     = 3306141    |    4479972|  0,015628|
-------------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Быстрая сортировка            |53626       + 172540      = 226166     |     259756|  0,000000|
В обратном порядке  | 30000|Быстрая сортировка            |214754      + 618220      = 832974     |    1006376|  0,000000|
В обратном порядке  | 70000|Быстрая сортировка            |621115      + 1647778     = 2268893    |    2887444|  0,015625|
В обратном порядке  |100000|Быстрая сортировка            |969993      + 2486136     = 3456129    |    4479972|  0,015625|
-------------------------------------------------------------------------------------------------------------------------------

Process returned 0 (0x0)   execution time : 349.843 s
Press any key to continue.
```

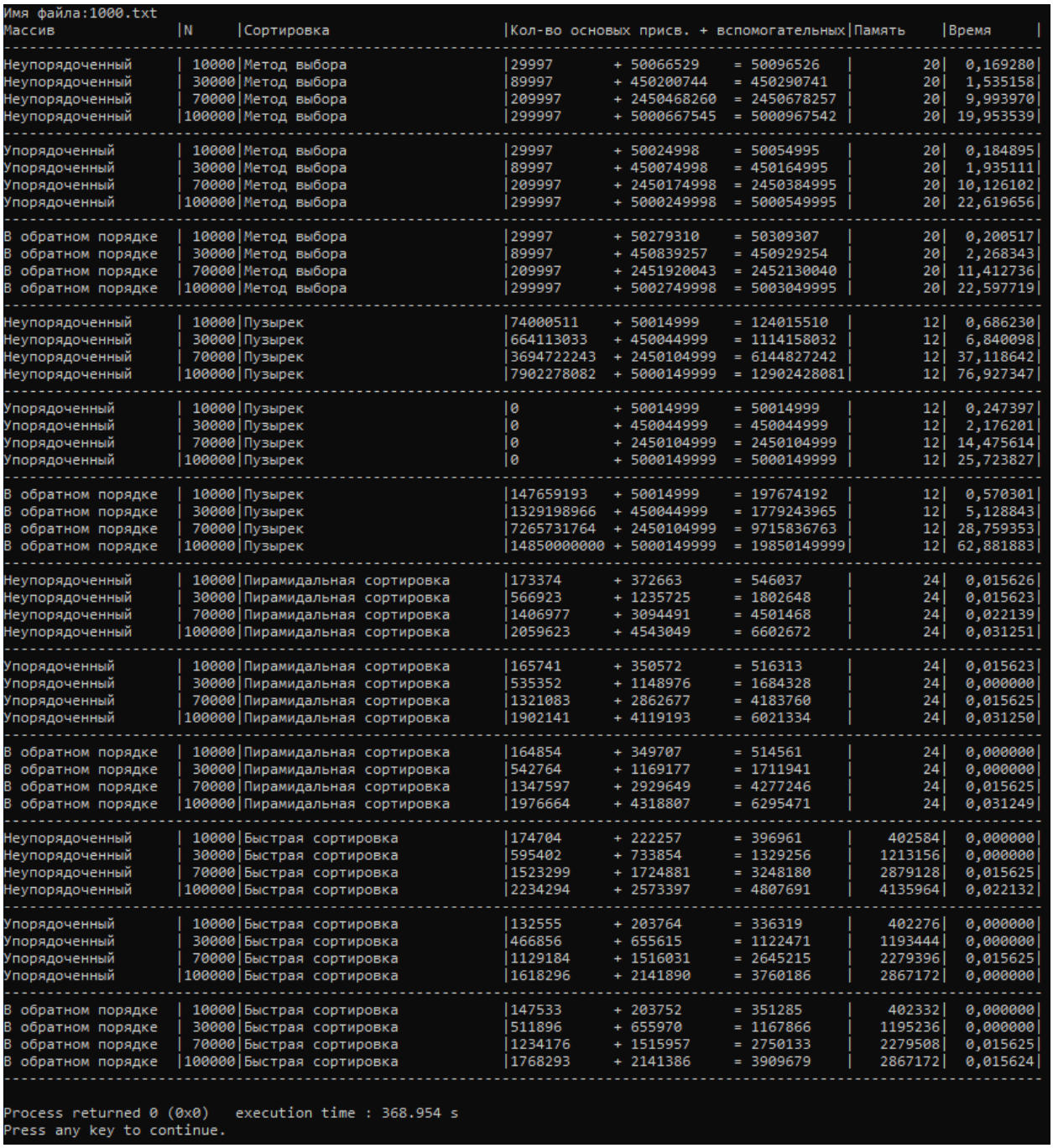
Результаты указанных сортировок с повторяющимися элементами по 100 штук

```
Имя файла:100.txt
Массив              |N     |Сортировка                  |Кол-во основых присв. + вспомогательных|Память    |Время     |
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Метод выбора                |29997       + 50087657    = 50117654    |        20|  0,162761|
Неупорядоченный     | 30000|Метод выбора                |89997       + 450269202   = 450359199   |        20|  1,488306|
Неупорядоченный     | 70000|Метод выбора                |209997      + 2450625119  = 2450835116  |        20|  8,883121|
Неупорядоченный     |100000|Метод выбора                |299997      + 5000898976  = 5001198973  |        20| 17,445985|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Метод выбора                |29997       + 50024998    = 50054995    |        20|  0,184892|
Упорядоченный       | 30000|Метод выбора                |89997       + 450074998   = 450164995   |        20|  1,588534|
Упорядоченный       | 70000|Метод выбора                |209997      + 2450174998  = 2450384995  |        20|  8,838479|
Упорядоченный       |100000|Метод выбора                |299997      + 5000249998  = 5000549995  |        20| 19,347714|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Метод выбора                |29997       + 52463867    = 52493864    |        20|  0,216141|
В обратном порядке  | 30000|Метод выбора                |89997       + 457487161   = 457577158   |        20|  1,748615|
В обратном порядке  | 70000|Метод выбора                |209997      + 2467610661  = 2467820658  |        20|  9,040772|
В обратном порядке  |100000|Метод выбора                |299997      + 5025249998  = 5025549995  |        20| 18,633431|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пузырек                     |75075309    + 50014999    = 125090308   |        12|  0,527979|
Неупорядоченный     | 30000|Пузырек                     |668525118   + 450044999   = 1118570117  |        12|  5,003866|
Неупорядоченный     | 70000|Пузырек                     |3625909536  + 2450104999  = 6076014535  |        12| 27,815917|
Неупорядоченный     |100000|Пузырек                     |7279979541  + 5000149999  = 12280129540 |        12| 60,811628|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пузырек                     |0           + 50014999    = 50014999    |        12|  0,238271|
Упорядоченный       | 30000|Пузырек                     |0           + 450044999   = 450044999   |        12|  2,038747|
Упорядоченный       | 70000|Пузырек                     |0           + 2450104999  = 2450104999  |        12| 11,992564|
Упорядоченный       |100000|Пузырек                     |0           + 5000149999  = 5000149999  |        12| 23,196826|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пузырек                     |149753136   + 50014999    = 199768135   |        12|  0,548209|
В обратном порядке  | 30000|Пузырек                     |1347887799  + 450044999   = 1797932798  |        12|  5,037384|
В обратном порядке  | 70000|Пузырек                     |7341537195  + 2450104999  = 9791642194  |        12| 29,094728|
В обратном порядке  |100000|Пузырек                     |14985000000 + 5000149999  = 19985149999 |        12| 56,050588|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пирамидальная сортировка    |174138      + 375075      = 549213      |        24|  0,015624|
Неупорядоченный     | 30000|Пирамидальная сортировка    |569420      + 1243545     = 1812965     |        24|  0,006509|
Неупорядоченный     | 70000|Пирамидальная сортировка    |1414569     + 3116934     = 4531503     |        24|  0,015626|
Неупорядоченный     |100000|Пирамидальная сортировка    |2075496     + 4585988     = 6661484     |        24|  0,046874|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пирамидальная сортировка    |176608      + 378068      = 554676      |        24|  0,000000|
Упорядоченный       | 30000|Пирамидальная сортировка    |556793      + 1203662     = 1760455     |        24|  0,000000|
Упорядоченный       | 70000|Пирамидальная сортировка    |1352639     + 2942120     = 4294759     |        24|  0,031251|
Упорядоченный       |100000|Пирамидальная сортировка    |1938738     + 4213253     = 6151991     |        24|  0,031258|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пирамидальная сортировка    |166296      + 353375      = 519671      |        24|  0,000000|
В обратном порядке  | 30000|Пирамидальная сортировка    |546914      + 1180749     = 1727663     |        24|  0,015625|
В обратном порядке  | 70000|Пирамидальная сортировка    |1356777     + 2954526     = 4311303     |        24|  0,015632|
В обратном порядке  |100000|Пирамидальная сортировка    |1990648     + 4356104     = 6346752     |        24|  0,031251|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Быстрая сортировка          |144861      + 222360      = 367221      |    371448|  0,015625|
Неупорядоченный     | 30000|Быстрая сортировка          |508986      + 732880      = 1241866     |   1156260|  0,015633|
Неупорядоченный     | 70000|Быстрая сортировка          |1308914     + 1766278     = 3075192     |   2716532|  0,015626|
Неупорядоченный     |100000|Быстрая сортировка          |1945470     + 2575012     = 4520482     |   3892728|  0,015629|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Быстрая сортировка          |81768       + 199617      = 281385      |    377664|  0,000000|
Упорядоченный       | 30000|Быстрая сортировка          |320408      + 661797      = 982205      |   1216628|  0,000000|
Упорядоченный       | 70000|Быстрая сортировка          |843567      + 1612148     = 2455715     |   2705892|  0,000000|
Упорядоченный       |100000|Быстрая сортировка          |1246996     + 2309538     = 3556534     |   3583972|  0,015626|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Быстрая сортировка          |96772       + 199694      = 296466      |    378112|  0,000000|
В обратном порядке  | 30000|Быстрая сортировка          |365296      + 661758      = 1027054     |   1216180|  0,000000|
В обратном порядке  | 70000|Быстрая сортировка          |948375      + 1612023     = 2560398     |   2705892|  0,000000|
В обратном порядке  |100000|Быстрая сортировка          |1396993     + 2309482     = 3706475     |   3583972|  0,015626|
-------------------------------------------------------------------------------------------------------------------------

Process returned 0 (0x0)   execution time : 315.790 s
Press any key to continue.
```

Результаты указанных сортировок с повторяющимися элементами по 500 штук

```
Имя файла:500.txt
Массив              |N     |Сортировка                  |Кол-во основых присв. + вспомогательных|Память    |Время     |
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Метод выбора                |29997      + 50072504    = 50102501    |        20|  0,169271|
Неупорядоченный     | 30000|Метод выбора                |89997      + 450220387   = 450310384   |        20|  1,472669|
Неупорядоченный     | 70000|Метод выбора                |209997     + 2450513899  = 2450723896  |        20|  8,372599|
Неупорядоченный     |100000|Метод выбора                |299997     + 5000741664  = 5001041661  |        20| 17,680202|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Метод выбора                |29997      + 50024998    = 50054995    |        20|  0,178383|
Упорядоченный       | 30000|Метод выбора                |89997      + 450074998   = 450164995   |        20|  1,588777|
Упорядоченный       | 70000|Метод выбора                |209997     + 2450174998  = 2450384995  |        20|  8,893279|
Упорядоченный       |100000|Метод выбора                |299997     + 5000249998  = 5000549995  |        20| 17,921289|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Метод выбора                |29997      + 50520082    = 50550079    |        20|  0,184897|
В обратном порядке  | 30000|Метод выбора                |89997      + 451540800   = 451630797   |        20|  1,682333|
В обратном порядке  | 70000|Метод выбора                |209997     + 2453649389  = 2453859386  |        20|  9,620774|
В обратном порядке  |100000|Метод выбора                |299997     + 5005249998  = 5005549995  |        20| 26,359237|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пузырек                     |74560809   + 50014999    = 124575808   |        12|  0,886744|
Неупорядоченный     | 30000|Пузырек                     |670863567  + 450044999   = 1120908566  |        12|  9,294519|
Неупорядоченный     | 70000|Пузырек                     |3692372253 + 2450104999  = 6142477252  |        12| 31,459954|
Неупорядоченный     |100000|Пузырек                     |7402556997 + 5000149999  = 12402706996 |        12| 68,804653|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пузырек                     |0          + 50014999    = 50014999    |        12|  0,253906|
Упорядоченный       | 30000|Пузырек                     |0          + 450044999   = 450044999   |        12|  2,074231|
Упорядоченный       | 70000|Пузырек                     |0          + 2450104999  = 2450104999  |        12| 12,736680|
Упорядоченный       |100000|Пузырек                     |0          + 5000149999  = 5000149999  |        12| 24,432751|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пузырек                     |148822641  + 50014999    = 198837640   |        12|  0,601557|
В обратном порядке  | 30000|Пузырек                     |1339568334 + 450044999   = 1789613333  |        12|  5,213596|
В обратном порядке  | 70000|Пузырек                     |7307817600 + 2450104999  = 9757922599  |        12| 33,890200|
В обратном порядке  |100000|Пузырек                     |14925000000 + 5000149999 = 19925149999 |        12| 56,668858|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пирамидальная сортировка    |173582     + 373366      = 546948      |        24|  0,000000|
Неупорядоченный     | 30000|Пирамидальная сортировка    |567563     + 1237956     = 1805519     |        24|  0,015625|
Неупорядоченный     | 70000|Пирамидальная сортировка    |1410487    + 3104683     = 4515170     |        24|  0,031257|
Неупорядоченный     |100000|Пирамидальная сортировка    |2069825    + 4569491     = 6639316     |        24|  0,031251|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пирамидальная сортировка    |167431     + 354767      = 522198      |        24|  0,015633|
Упорядоченный       | 30000|Пирамидальная сортировка    |537881     + 1155510     = 1693391     |        24|  0,000000|
Упорядоченный       | 70000|Пирамидальная сортировка    |1326862    + 2877823     = 4204685     |        24|  0,015625|
Упорядоченный       |100000|Пирамидальная сортировка    |1910165    + 4141219     = 6051384     |        24|  0,022133|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пирамидальная сортировка    |165236     + 350439      = 515675      |        24|  0,000000|
В обратном порядке  | 30000|Пирамидальная сортировка    |543801     + 1171984     = 1715785     |        24|  0,015624|
В обратном порядке  | 70000|Пирамидальная сортировка    |1351484    + 2940005     = 4291489     |        24|  0,022134|
В обратном порядке  |100000|Пирамидальная сортировка    |1983138    + 4335891     = 6319029     |        24|  0,031251|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Быстрая сортировка          |165087     + 225196      = 390283      |    393960|  0,000000|
Неупорядоченный     | 30000|Быстрая сортировка          |574150     + 700535      = 1274685     |   1205260|  0,015624|
Неупорядоченный     | 70000|Быстрая сортировка          |1457191    + 1721988     = 3179179     |   2833852|  0,015634|
Неупорядоченный     |100000|Быстрая сортировка          |2159258    + 2499251     = 4658509     |   4063052|  0,031254|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Быстрая сортировка          |116647     + 202471      = 319118      |    393988|  0,000000|
Упорядоченный       | 30000|Быстрая сортировка          |421898     + 655914      = 1077812     |   1188068|  0,015632|
Упорядоченный       | 70000|Быстрая сортировка          |1023764    + 1515746     = 2539510     |   2279396|  0,000000|
Упорядоченный       |100000|Быстрая сортировка          |1468596    + 2142540     = 3611136     |   2867172|  0,015629|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Быстрая сортировка          |131636     + 202532      = 334168      |    394436|  0,000000|
В обратном порядке  | 30000|Быстрая сортировка          |466578     + 656051      = 1122629     |   1189860|  0,000000|
В обратном порядке  | 70000|Быстрая сортировка          |1128700    + 1516045     = 2644745     |   2280292|  0,000000|
В обратном порядке  |100000|Быстрая сортировка          |1618593    + 2142286     = 3760879     |   2867172|  0,015625|
-------------------------------------------------------------------------------------------------------------------------

Process returned 0 (0x0)   execution time : 345.040 s
Press any key to continue.
```

Результаты указанных сортировок с повторяющимися элементами по 1000 штук



```
Имя файла:1000.txt
Массив              |N     |Сортировка                  |Кол-во основных присв. + вспомогательных|Память    |Время     |
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Метод выбора                |29997      + 50066529    = 50096526    |        20|  0,169280|
Неупорядоченный     | 30000|Метод выбора                |89997      + 450200744   = 450290741   |        20|  1,535158|
Неупорядоченный     | 70000|Метод выбора                |209997     + 2450468260  = 2450678257  |        20|  9,993970|
Неупорядоченный     |100000|Метод выбора                |299997     + 5000667545  = 5000967542  |        20| 19,953539|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Метод выбора                |29997      + 50024998    = 50054995    |        20|  0,184895|
Упорядоченный       | 30000|Метод выбора                |89997      + 450074998   = 450164995   |        20|  1,935111|
Упорядоченный       | 70000|Метод выбора                |209997     + 2450174998  = 2450384995  |        20| 10,126102|
Упорядоченный       |100000|Метод выбора                |299997     + 5000249998  = 5000549995  |        20| 22,619656|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Метод выбора                |29997      + 50279310    = 50309307    |        20|  0,200517|
В обратном порядке  | 30000|Метод выбора                |89997      + 450839257   = 450929254   |        20|  2,268343|
В обратном порядке  | 70000|Метод выбора                |209997     + 2451920043  = 2452130040  |        20| 11,412736|
В обратном порядке  |100000|Метод выбора                |299997     + 5002749998  = 5003049995  |        20| 22,597719|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пузырек                     |74000511   + 50014999    = 124015510   |        12|  0,686230|
Неупорядоченный     | 30000|Пузырек                     |664113033  + 450044999   = 1114158032  |        12|  6,840098|
Неупорядоченный     | 70000|Пузырек                     |3694722243 + 2450104999  = 6144827242  |        12| 37,118642|
Неупорядоченный     |100000|Пузырек                     |7902278082 + 5000149999  = 12902428081 |        12| 76,927347|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пузырек                     |0          + 50014999    = 50014999    |        12|  0,247397|
Упорядоченный       | 30000|Пузырек                     |0          + 450044999   = 450044999   |        12|  2,176201|
Упорядоченный       | 70000|Пузырек                     |0          + 2450104999  = 2450104999  |        12| 14,475614|
Упорядоченный       |100000|Пузырек                     |0          + 5000149999  = 5000149999  |        12| 25,723827|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пузырек                     |147659193  + 50014999    = 197674192   |        12|  0,570301|
В обратном порядке  | 30000|Пузырек                     |1329198966 + 450044999   = 1779243965  |        12|  5,128843|
В обратном порядке  | 70000|Пузырек                     |7265731764 + 2450104999  = 9715836763  |        12| 28,759353|
В обратном порядке  |100000|Пузырек                     |14850000000 + 5000149999 = 19850149999 |        12| 62,881883|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Пирамидальная сортировка    |173374     + 372663      = 546037      |        24|  0,015626|
Неупорядоченный     | 30000|Пирамидальная сортировка    |566923     + 1235725     = 1802648     |        24|  0,015623|
Неупорядоченный     | 70000|Пирамидальная сортировка    |1406977    + 3094491     = 4501468     |        24|  0,022139|
Неупорядоченный     |100000|Пирамидальная сортировка    |2059623    + 4543049     = 6602672     |        24|  0,031251|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Пирамидальная сортировка    |165741     + 350572      = 516313      |        24|  0,015623|
Упорядоченный       | 30000|Пирамидальная сортировка    |535352     + 1148976     = 1684328     |        24|  0,000000|
Упорядоченный       | 70000|Пирамидальная сортировка    |1321083    + 2862677     = 4183760     |        24|  0,015625|
Упорядоченный       |100000|Пирамидальная сортировка    |1902141    + 4119193     = 6021334     |        24|  0,031250|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Пирамидальная сортировка    |164854     + 349707      = 514561      |        24|  0,000000|
В обратном порядке  | 30000|Пирамидальная сортировка    |542764     + 1169177     = 1711941     |        24|  0,000000|
В обратном порядке  | 70000|Пирамидальная сортировка    |1347597    + 2929649     = 4277246     |        24|  0,015625|
В обратном порядке  |100000|Пирамидальная сортировка    |1976664    + 4318807     = 6295471     |        24|  0,031249|
-------------------------------------------------------------------------------------------------------------------------
Неупорядоченный     | 10000|Быстрая сортировка          |174704     + 222257      = 396961      |    402584|  0,000000|
Неупорядоченный     | 30000|Быстрая сортировка          |595402     + 733854      = 1329256     |   1213156|  0,000000|
Неупорядоченный     | 70000|Быстрая сортировка          |1523299    + 1724881     = 3248180     |   2879128|  0,015625|
Неупорядоченный     |100000|Быстрая сортировка          |2234294    + 2573397     = 4807691     |   4135964|  0,022132|
-------------------------------------------------------------------------------------------------------------------------
Упорядоченный       | 10000|Быстрая сортировка          |132555     + 203764      = 336319      |    402276|  0,000000|
Упорядоченный       | 30000|Быстрая сортировка          |466856     + 655615      = 1122471     |   1193444|  0,000000|
Упорядоченный       | 70000|Быстрая сортировка          |1129184    + 1516031     = 2645215     |   2279396|  0,015625|
Упорядоченный       |100000|Быстрая сортировка          |1618296    + 2141890     = 3760186     |   2867172|  0,000000|
-------------------------------------------------------------------------------------------------------------------------
В обратном порядке  | 10000|Быстрая сортировка          |147533     + 203752      = 351285      |    402332|  0,000000|
В обратном порядке  | 30000|Быстрая сортировка          |511896     + 655970      = 1167866     |   1195236|  0,000000|
В обратном порядке  | 70000|Быстрая сортировка          |1234176    + 1515957     = 2750133     |   2279508|  0,015625|
В обратном порядке  |100000|Быстрая сортировка          |1768293    + 2141386     = 3909679     |   2867172|  0,015624|
-------------------------------------------------------------------------------------------------------------------------

Process returned 0 (0x0)   execution time : 368.954 s
Press any key to continue.
```

Анализ полученных данных:

Полученные результаты практически совпадают с расчетными, значит расчеты верные.

По трудоемкости в лучшем и среднем случаях расположим сортировки в список от лучшей к худшей: быстрая, пирамидальная, выбором пузырьком. В худшем случае: пирамидальная, выбором, быстрая, пузырьковая.

Время работы в лучшем и среднем случаях от лучшей к худшей сортировке: быстрая, пирамидальная, выбором, пузырьковая. Что соотносится с общим кол-вом посчитанных

операций, самый медленный пузырек – у него наибольшее кол-во операций, сама быстрая – быстрая сортировка – у нее наименьшее кол-во. В худшем: пирамидальная, выбором, быстрая, пузырьком.

По пространственной сложности от лучшей к худшей: пузырек, выбором, пирамидальная, быстрая. У первых трех не зависит от кол-ва элементов. Хоть пузырек и выигрывает здесь, но скорость у него самая маленькая, с другой же стороны быстрая сортировка самая быстрая, но здесь проигрывает. Получается, самым оптимальным было бы использовать пирамидальную сортировку, она и по скорости от быстрой практически не отстает, и пространственная сложность не зависит от кол-ва элементов.

Повторяющиеся ключи не особо повлияли на трудоемкость пирамидальной сортировки. Но заметно, что для быстрой сортировки они сильно повышают трудоемкость, а именно – чем больше повторяющихся ключей, тем сильнее повышается трудоемкость. С другой стороны, для сортировки выбором наоборот – с повышением кол-ва повт. ключей понижается трудоемкость, особенно это заметно при сортировке массива, упорядоченного в обратном порядке, так как кол-во уникальных ключей понизилось, с ними же понизилось переприсваивание минимума. Также и с пузырьком – время работы понизилось, так как кол-во уникальных ключей уменьшилось, что уменьшило перестановки элементов.

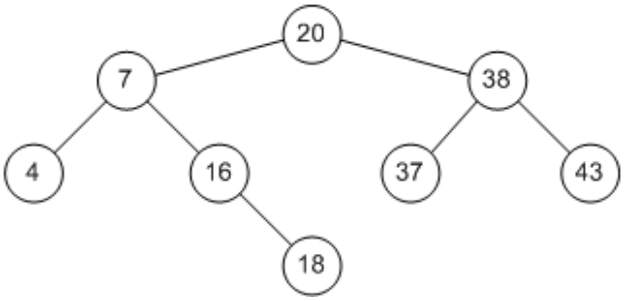
Вывод: по результатам проведенного анализа самым эффективным из рассмотренных алгоритмов по соотношению время-память является алгоритм пирамидальной сортировки

Задача 2.

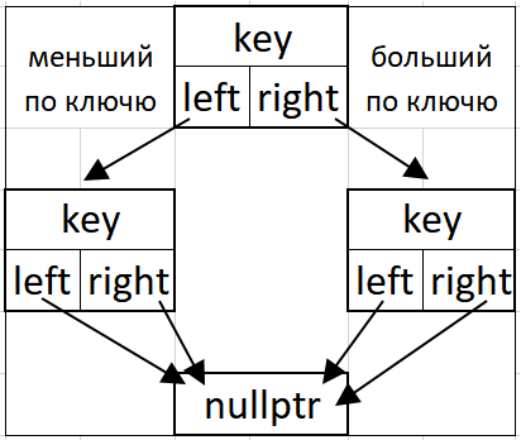
Реализовать две указанные структуры данных, заполнив их значениями из приложенного файла *test_numbers.txt*. Выполнить поиск 100 ключей в указанных структурах данных, для каждого ключа выводить сообщение о том, найден он или нет, и количество выполненных при поиске сравнений ключей, в конце программы вывести среднее количество сравнений, пришедшееся на один ключ. При формировании тестового набора включать в него как имеющиеся в файле, так и отсутствующие в нем ключи. Оценить количество требуемой памяти для реализации каждой структуры и количество сравнений при поиске.

Несбалансированное бинарное дерево поиска, декартово дерево.

Схематичное изображение несбалансированного бинарного дерева



Схематичное изображение структуры хранения, использованной в программе для программирования несбалансированного бинарного дерева:

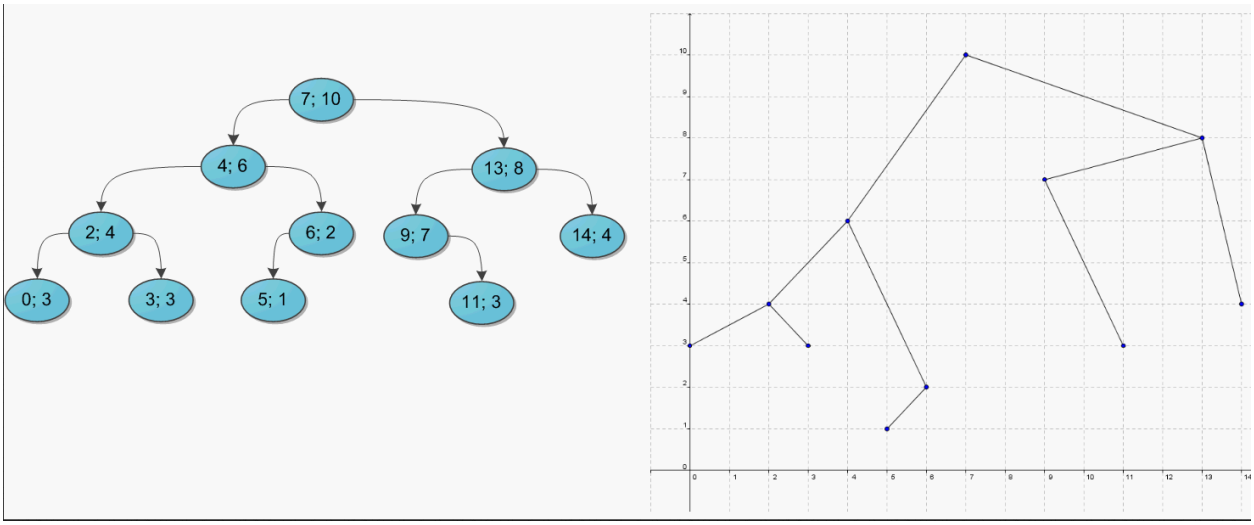


Трудоемкость поиска:

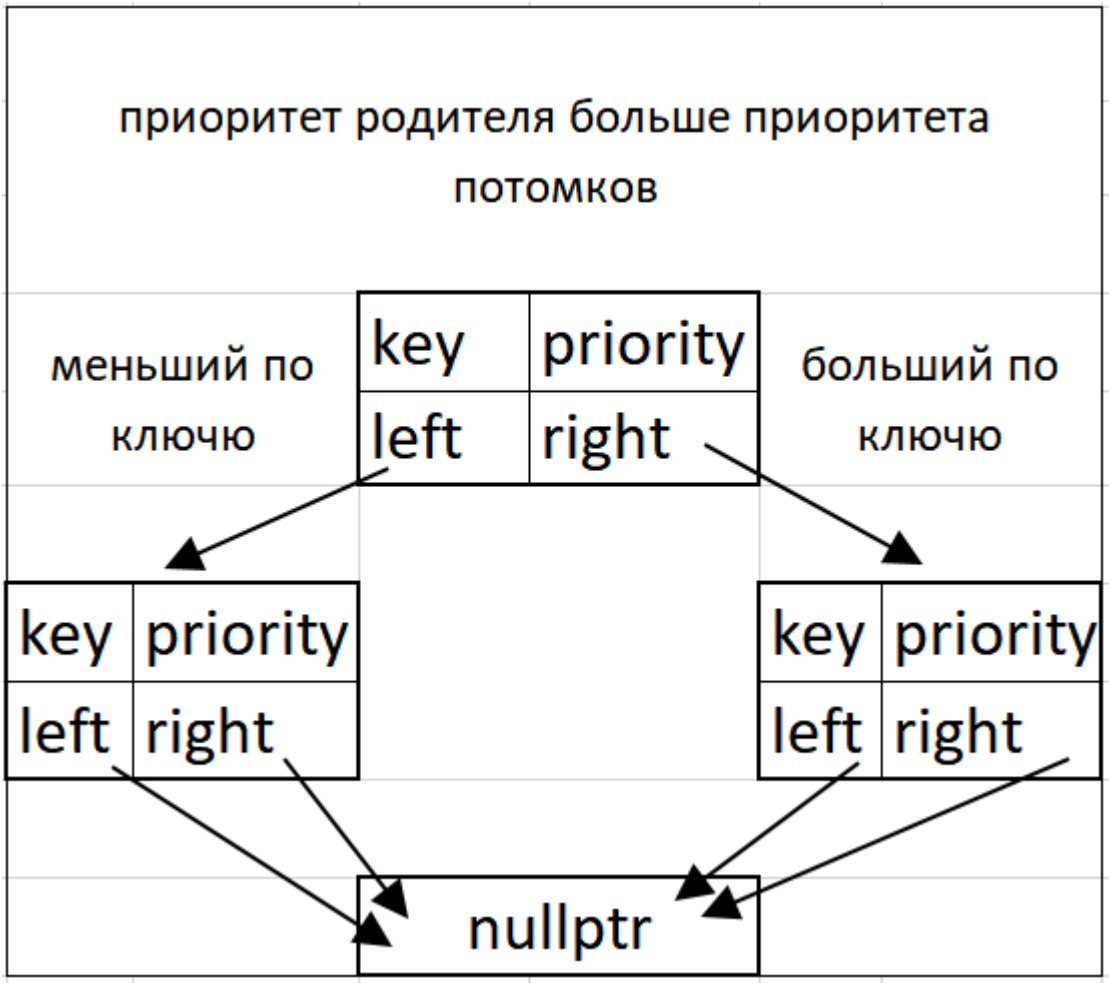
Случай	Ситуация, соответствующая случаю	Обоснование	Ожидаемое число сравнений при поиске	**Асимптотическая оценка сложности поиска**
наилучший	искомый ключ находится в корне дерева	требуется проверить один ключ – корень, 3 сравнения	3	Ω(1)
наихудший	дерево вырожденное, искомый ключ отсутствует в дереве	требуется осуществить перебор всех ключей, количество сравнений равно количеству узлов * 3	3*100000	O(N)
наиболее вероятный	дерево не вырожденное, искомый ключ присутствует в дереве	требуется осуществить перебор ключей на одной ветке дерева три сравнения на ключ умножить на среднюю высоту дерева	$3*\log_2 100000$ ~50	O(logN)

Требуемый объем памяти (sizeof(long long)+2*sizeof(pointer))*N=(8+2*8)*N=2400000 байт

Схематичное изображение декартова дерева:



Схематичное изображение структуры хранения, использованной в программе для программирования декартова дерева:



Трудоемкость поиска:

Случай	Ситуация, соответствующая случаю	Обоснование	Ожидаемое число сравнений при поиске	**Асимптотическая оценка сложности поиска**
наилучший	искомый ключ находится в корне	требуется проверить один ключ – корень, 3	3	Ω(1)

	дерева	сравнения		
наихудший	дерево вырожденное, что очень маловероятно, искомый ключ отсутствует в дереве	требуется осуществить перебор всех ключей, количество сравнений равно количеству узлов * 3	3*100000	O(N)
наиболее вероятный	дерево не вырожденное, искомый ключ присутствует в дереве	требуется осуществить перебор ключей на одной ветке дерева три сравнения на ключ умножить на среднюю высоту дерева	$3*\log_2 100000$ ~50	O(logN)

Требуемый объем памяти

(2*sizeof (long long) +2*sizeof (pointer))*N= (2*8+2*8)*N=3200000 байт

Тестовый набор ключей:

присутствующие в файле:

49117651 36186109 74050165 50286471 61592488 10346557 62363782 73922392 96961206 51839542 49379470 48048915 83518709 68538628 70853791 63959591 38047866 88542550 12588286 28805679 94545499 12496565 75617033 13476598 57165625 94041114 81313258 69935054 29955459 77543666 75995785 32559879 52984009 26623744 82494427 25717586 35369719 48441088 43168186 30346113 92194708 34457100 35880683 25382853 20213799 41315834 41613254 44911543 43595696 84840870 36296599 75491285 80897972 90453881 44447209 46540891 73749457 80548018 83169581 13463365 14523788 45019956 74151124 27984588 38523648 76196738 32283584 95409296 88718644 67020491

отсутствующие в файле:

68548750 45979104 16159476 94855968 38914691 89489880 79986422 72300925 13248050 60599938 38179949 29139706 98662397 82460363 81486164 93895457 45992239 68168305 56487257 65508395 28880652 90018761 18405904 89779491

меньше, чем в файле:

863259 2190 3821 1841 8042 4252

Текст программы:

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <random>
#define N 100000

/*шаблонная функция поиска, первый параметр должен быть BinaryNode или DecardNode*/
template<class T>
T* _search(T* root, long long int key, int* currentComprasions)
{
```

```
            T* tmp = root;
            bool flag = true;
            *currentComprasions = 0;
            while(flag)
            {
                ++*currentComprasions;
                if(tmp == nullptr)
                {
                    flag = false;
                }
                else
                    /*если искомых ключ меньше, то нужно искать в левом поддереве*/
                    if(key < tmp->key)
                    {
                        ++*currentComprasions;
                        tmp = tmp->left;
                    }
                    else
                        /*если искомых ключ больше, то нужно искать в правом
поддереве*/
                        if(key > tmp->key)
                        {
                            *currentComprasions+=2;
                            tmp = tmp->right;
                        }
                        /*в этом случае, есть вероятность, что ключ был найден*/
                        else
                        {
                            *currentComprasions+=2;
                            flag = false;
                        }
            }
            return tmp;
        }
        /*генератор больших чисел, чем rand()*/
        std::mt19937 mersenne(static_cast<unsigned int>(time(0)));

        class BinaryNode
        {
        public:
            long long int key;
            BinaryNode *left, *right;
            BinaryNode(long long int k, BinaryNode *l, BinaryNode *r)
            {
                key = k;
                left = l;
                right = r;
            }
            ~BinaryNode()
            {
                delete left;
                delete right;
            }
        };
        /*Несбалансированное бинарное дерево*/
        class BinaryTree
        {
            BinaryNode* root;
        public:
            BinaryTree()
            {
                root = nullptr;
            }
```

```
    ~BinaryTree()
    {
        delete root;
    }
    /*добавение вершины в несбалансированное бинарное дерево*/
    void addNode(long long int key)
    {
        BinaryNode* tmp = root, *tmpParent = nullptr;
        bool left;
        while(tmp)
        {
            tmpParent = tmp;
            if(key < tmp->key)
            {
                tmp = tmp->left;
                left = true;
            }
            else
            {
                tmp = tmp->right;
                left = false;
            }

        }
        tmp = new BinaryNode(key, nullptr, nullptr);
        if(!tmpParent)
            root = tmp;
        else
            if(left)
                tmpParent->left = tmp;
            else
                tmpParent->right = tmp;
    }
    BinaryNode* search(long long int key, int* currentComprasions)
    {
        return _search(root, key, currentComprasions);
    }

};

/*Декартово дерево*/
class DecardNode
{
public:
    long long int key, priority;
    DecardNode *left, *right;
    DecardNode(long long int k)
    {
        key = k;
        priority = mersenne();
        left = nullptr;
        right = nullptr;
    }
    ~DecardNode()
    {
        delete left;
        delete right;
    }
};
class DecardTree
{
    /*слияние двух поддеревьев*/
    DecardNode * merge (DecardNode *l, DecardNode *r)
```

```
        {
            if (l==nullptr) return r;
            if (r==nullptr) return l;
            if (l->priority > r->priority)
            {
                l->right = merge (l->right, r);
                return l;
            }
            else
            {
                r->left = merge (l, r->left);
                return r;
            }
        }
        /*разделение декартово дерева по ключу*/
          void split (DecardNode *t, long long int key, DecardNode * &l,
DecardNode * &r)
          {
            if (t->key < key)
            {
                l = t;
                if (t->right == nullptr)
                   r = nullptr;
                else
                   split (t->right, key, l->right, r);
            }
            else
            {
                r = t;
                if (t->left == nullptr)
                   l = nullptr;
                else
                   split (t->left, key, l, r->left);
            }
          }
          /*удаление ключа из декартова дерева*/
          DecardNode * _delete_key (DecardNode* t, long long int key)
          {
            if (t == nullptr) return nullptr;
            if (t->key == key)
            {
                DecardNode * tmp = t;
                t = merge (t->left, t->right);
                delete tmp;
                return t;
            }
            if (key < t->key)
                t->left = _delete_key (t->left, key);
            else
                t->right = _delete_key (t->right, key);
            return t;
          }
        DecardNode* root;
    public:
        DecardTree()
        {
            root = nullptr;
        }
        ~DecardTree()
        {
            delete root;
        }
        /*добавление вершины в декартово дерево*/
```

```
        void addNode(long long key)
        {
            DecardNode *node = new DecardNode(key);
            if (root == nullptr)
            {
                root = node;
                return;
            }
            DecardNode *l=nullptr, *r=nullptr;
            split (root, key, l, r);
            root = merge (merge (l, node), r);
        }
        void delete_key (long long int key)
        {
            root = _delete_key (root, key);
        }
        /*поиск ключа в декартове дереве*/
        DecardNode* search(long long int key, int* currentComprasions)
        {
            return _search(root, key, currentComprasions);
        }
    };

    int main()
    {

        setlocale(LC_ALL, "rus");

        BinaryTree* bTree = new BinaryTree;
        DecardTree* dTree = new DecardTree;

        char findStr[2][10] = {"Не найден", "Найден"};
        /*comprasionsCount - кол-во сравнений для поиска одного ключа,
allComprasions - для поиска всех ключей, isFind - флаг, было ли число найдено,
b - массив ключей, которые будем искать*/
        int comprasionsCount[2], allComprasions[2], isFind[2], a[N], b[100],
i;
        allComprasions[0] = allComprasions[1] = 0;

        /*указатели на искомый элемент*/
        BinaryNode* binaryResult;
        DecardNode* decardResult;

        std::fstream f("test_numbers.txt", std::ios::in);
        for(i = 0; i<N; i++)
            f>>a[i];
        f.close();
        /*заполняем деревья*/
        for(i = 0; i<N; i++)
        {
            bTree->addNode(a[i]);
            dTree->addNode(a[i]);
        }
        /*формируем набор ключей для поиска*/
        /*ключи, которые точно будут, в деревьях, выбранные рандомно из
массива*/
        for(i = 0; i<70; i++)
            b[i] = a[mersenne()%N];
        /*ключи, которые могут быть, а могут и не быть, в деревьях,
сгенерированные рандомно*/
        for(i = 70; i<95; i++)
            b[i] = mersenne()%100000000;
        /*ключи, которых точно нет в деревьях, сгенерированные рандомно*/
```

```
          for(i = 95; i<100; i++)
              b[i] = mersenne() % 10000;

          for(i=0; i<100; i++)
              std::cout<<b[i]<<" ";
              std::cout<<"\n";
          printf("|%-41s|%10s|%-41s|\n",     "Бинарное    дерево","","Декартово
дерево");
          printf("-------------------------------------------%10s------------
------------------------------\n","");
          printf("|%-10s|%-s|%-s|%-10s|%-10s|%-s|%-s|\n",   "ключ",   "найден/не
найден", "кол-во сравн.", "", "ключ", "найден/не найден", "кол-во сравн.");
          printf("-------------------------------------------%10s------------
------------------------------\n","");

          for(i = 0; i<100; i++)
          {
              binaryResult = bTree->search(b[i], &comprasionsCount[0]);
              decardResult = dTree->search(b[i], &comprasionsCount[1]);
              /*проверяем правильный ли ключ найден, и найден ли вообще*/
              isFind[0] = binaryResult && binaryResult->key == b[i] ? 1 : 0;
              isFind[1] = decardResult && decardResult->key == b[i] ? 1 : 0;
              allComprasions[0]+= ++comprasionsCount[0];
              allComprasions[1]+= ++comprasionsCount[1];
              printf("|%10d|%16s|%13d|%10s|%10d|%16s|%13d|\n",           b[i],
findStr[isFind[0]],      comprasionsCount[0],"",b[i],      findStr[isFind[1]],
comprasionsCount[1]);
              printf("-------------------------------------------%10s---------
---------------------------------\n","");
          }
          std::cout   << "Среднее кол-во сравнений:\n"
                      <<                    "Бинарное                   дерево:
"<<((double)allComprasions[0]/100)<<"\n"
                      <<                    "Декартово                  дерево:
"<<(double)allComprasions[1]/100<<"\n";
          delete bTree;
          delete dTree;
          return 0;
      }
```

Результаты работы программы

Бинарное дерево			Декартово дерево		
ключ	найден/не найден	кол-во сравн.	ключ	найден/не найден	кол-во сравн.
49117651	Найден	37	49117651	Найден	68
36186109	Найден	36	36186109	Найден	59
74050165	Найден	64	74050165	Найден	52
50286471	Найден	45	50286471	Найден	63
61592488	Найден	54	61592488	Найден	62
10346557	Найден	50	10346557	Найден	62
62363782	Найден	39	62363782	Найден	71
73922392	Найден	79	73922392	Найден	74
96961206	Найден	68	96961206	Найден	47
51839542	Найден	46	51839542	Найден	45
49379470	Найден	48	49379470	Найден	49
48048915	Найден	39	48048915	Найден	38
83518709	Найден	49	83518709	Найден	58
68538628	Найден	59	68538628	Найден	57
70853791	Найден	85	70853791	Найден	68
63959591	Найден	66	63959591	Найден	35
38047866	Найден	46	38047866	Найден	45
88542550	Найден	59	88542550	Найден	47
12588286	Найден	49	12588286	Найден	62
28805679	Найден	47	28805679	Найден	53
94545499	Найден	54	94545499	Найден	56
12496565	Найден	55	12496565	Найден	64
75617033	Найден	51	75617033	Найден	65
13476598	Найден	47	13476598	Найден	52
57165625	Найден	76	57165625	Найден	30
94041114	Найден	64	94041114	Найден	34
81313258	Найден	48	81313258	Найден	41
69935054	Найден	78	69935054	Найден	56
29955459	Найден	72	29955459	Найден	70
77543666	Найден	47	77543666	Найден	53
75995785	Найден	61	75995785	Найден	76
32559879	Найден	67	32559879	Найден	66
52984009	Найден	51	52984009	Найден	65
26623744	Найден	74	26623744	Найден	50

82494427	Найден	50	82494427	Найден	39
25717586	Найден	49	25717586	Найден	53
35369719	Найден	40	35369719	Найден	55
48441088	Найден	45	48441088	Найден	52
43168186	Найден	48	43168186	Найден	39
30346113	Найден	64	30346113	Найден	74
92194708	Найден	63	92194708	Найден	58
34457100	Найден	52	34457100	Найден	51
35880683	Найден	60	35880683	Найден	51
25382853	Найден	59	25382853	Найден	66
20213799	Найден	47	20213799	Найден	50
41315834	Найден	63	41315834	Найден	74
41613254	Найден	59	41613254	Найден	59
44911543	Найден	43	44911543	Найден	54
43595696	Найден	40	43595696	Найден	50
84840870	Найден	53	84840870	Найден	51
36296599	Найден	41	36296599	Найден	50
75491285	Найден	56	75491285	Найден	65
80897972	Найден	66	80897972	Найден	34
90453881	Найден	61	90453881	Найден	49
44447209	Найден	46	44447209	Найден	52
46540891	Найден	32	46540891	Найден	39
73749457	Найден	70	73749457	Найден	66
80548018	Найден	66	80548018	Найден	35
83169581	Найден	50	83169581	Найден	45
13463365	Найден	54	13463365	Найден	60
14523788	Найден	42	14523788	Найден	48
45019956	Найден	45	45019956	Найден	64
74151124	Найден	60	74151124	Найден	57
27984588	Найден	57	27984588	Найден	67
38523648	Найден	44	38523648	Найден	62
76196738	Найден	66	76196738	Найден	44
32283584	Найден	56	32283584	Найден	95
95409296	Найден	69	95409296	Найден	49

:

88718644	Найден	67	88718644	Найден	32
67020491	Найден	79	67020491	Найден	34
68548750	Не найден	57	68548750	Не найден	56
45979104	Не найден	50	45979104	Не найден	50
16159476	Не найден	48	16159476	Не найден	70
94855968	Не найден	58	94855968	Не найден	52
38914691	Не найден	48	38914691	Не найден	72
89489880	Не найден	71	89489880	Не найден	47
79986422	Не найден	72	79986422	Не найден	56
72300925	Не найден	85	72300925	Не найден	64
13248050	Не найден	57	13248050	Не найден	63
60599938	Не найден	60	60599938	Не найден	60
38179949	Не найден	48	38179949	Не найден	55
29139706	Не найден	67	29139706	Не найден	66
98662397	Не найден	52	98662397	Не найден	51
82460363	Не найден	64	82460363	Не найден	54
81486164	Не найден	45	81486164	Не найден	37
93895457	Не найден	61	93895457	Не найден	47
45992239	Не найден	48	45992239	Не найден	48
68168305	Не найден	71	68168305	Не найден	49
56487257	Не найден	50	56487257	Не найден	37
65508395	Не найден	58	65508395	Не найден	46
28880652	Не найден	60	28880652	Не найден	53
90018761	Не найден	62	90018761	Не найден	60
863259	Не найден	30	863259	Не найден	18
18405904	Не найден	53	18405904	Не найден	55
89779491	Не найден	54	89779491	Не найден	34
2190	Не найден	30	2190	Не найден	18
3821	Не найден	30	3821	Не найден	18
1841	Не найден	30	1841	Не найден	18
8042	Не найден	30	8042	Не найден	18
4252	Не найден	30	4252	Не найден	18

Среднее кол-во операций

```
Среднее кол-во сравнений:
Бинарное дерево:  54.51
Декартово дерево: 52.06
```

Выводы:

По полученным результатам среднее кол-во операций у деревьев схоже. Следовательно, и затраченное время должно быть одинаково. Но просто так получилось, что последовательность данных не была упорядочена или же слегка упорядочена, не важно по возрастанию или убыванию. Если бы это было так, то несбалансированное бинарное дерево проявило бы свои недостатки. Дерево получилось бы вырожденное и кол-во сравнений необходимых для поиска ключа стало бы равным ~3*n, что заметно бы понизило скорость работы. Но для Декартова дерева это бы не стало проблемой. У него есть приоритет для каждой вершины, из-за него построение вырожденного дерево очень маловероятно, а значит в таком случае поиск затратит ~3*logn сравнений. Худший случай, к сожалению, для Декартова дерева определить не получится, из-за рандомного приоритета. По памяти же однозначно лучше несбалансированное бинарное дерево, так как в Декартовом дополнительно нужно хранить приоритет.

Если выбирать из двух деревьев, то как раз из-за малой вероятности построения вырожденного дерева, лучшим вариантом будет Декартово дерево.