|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | |
| Федеральное государственное автономное образовательное учреждение  высшего образования | | | | | | | | | | | | |
| **Дальневосточный федеральный университет** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **Кафедра прикладной математики, механики, управления и программного обеспечения** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **О Т Ч Е Т** | | | | | | | | | | | | |
| по лабораторной работе №1.3  дисциплина «Фундаментальные структуры данных и алгоритмы» | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Студент гр. Б9120-09.03.04прогин | | | |
|  |  |  | | В.А. Кадомцев | |
|  | | | | | | |  |  | (подпись) | |  | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Руководитель | | |  |
|  | | | | | | |  |  | ст. преподаватель | | | |
|  | | | |  |  | |  |  |  |  | О.А. Крестникова | |
|  | | | |  |  | |  |  | (подпись) |  | (И.О. Фамилия) | |
|  |  |  |  | | |  |  |  |  | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| г. Владивосток | | | | | | | | | | | | |
| 2022 | | | | | | | | | | | | |

Оглавление

[Неформальная постановка задачи: 2](#_Toc93418212)

[Постановка задачи: 2](#_Toc93418213)

[Описание алгоритма сортировки 3](#_Toc93418214)

[Пирамидальная сортировка 3](#_Toc93418215)

[Сортировка вставками 3](#_Toc93418216)

[Текст программы 3](#_Toc93418217)

[Файл main.cpp 3](#_Toc93418218)

[Файл Flight.h 4](#_Toc93418219)

[Файл HeapSort.h 4](#_Toc93418220)

[Файл InsertionSort.h 5](#_Toc93418221)

[Пример входного файла (на 200 записей): 6](#_Toc93418222)

[Пример выходного файла (на 200 записей): 9](#_Toc93418223)

[Пирамидальная сортировка 9](#_Toc93418224)

[Сортировка вставками 13](#_Toc93418225)

[Тесты 17](#_Toc93418226)

[1. 200 случайно сгенерированных записей: 17](#_Toc93418227)

[2. 10000 случайно сгенерированных записей: 17](#_Toc93418228)

[3. Входные данные уже отсортированы: 19](#_Toc93418229)

[4. Входные данные отсортированы в обратном порядке: 19](#_Toc93418230)

[5. Входные данные содержат 100 одинаковых записей: 19](#_Toc93418231)

[Проверка устойчивости с обоснованием результатов 19](#_Toc93418232)

[Результат: 19](#_Toc93418233)

[Выводы: 20](#_Toc93418234)

# Неформальная постановка задачи:

Сортировать по возрастанию, алгоритмы: пирамидальная сортировка, простые вставки.

В строке файла: время вылета, код авиакомпании, код рейса, код аэропорта.

Сортировать по времени вылета, потом по коду авиакомпании.

# Постановка задачи:

Входные данные: текстовый файл, в первой строке – целое число n - количество строк файла (10 ≤ n ≤ 1000000). Каждая из последующих строк содержит данные заданной предметной области, разделенные 1 пробелом. Все данные во входном файле корректные.

Выходные данные: текстовый файл, в первой строке - время, затраченное на сортировку входного файла, каждая из последующих строк содержит отсортированные данные в табличном виде.

1. Отсортировать данные входного файла по возрастанию

2. Проверить на устойчивость (привести примеры, доказывающие, что сортировка неустойчивая)

3. Сравнить по времени (без учета чтения из файла и записи в файл)

4. Определить последовательности, на которых достигается наихудшее и наилучшее значение времени

# Описание алгоритма сортировки

## Пирамидальная сортировка

Пирамидальная сортировка (или сортировка кучей, HeapSort) — это 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.

## Сортировка вставками

Сортировка вставками (Insertion Sort) — это простой алгоритм сортировки. Суть его заключается в том что, на каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем. Стоит отметить что массив из 1-го элемента считается отсортированным.

# Текст программы

## Файл main.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <chrono>

#include <time.h>

#include "Flight.h"

#include "HeapSort.h"

#include "InsertionSort.h"

using namespace std;

int main()

{

ifstream in("input.txt");

ofstream out("output.txt");

string line;

int n;

in >> n;

flight\* fl = new flight[n];

//Get info from file

getline(in, line);

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

{

in >> fl[i].time.hour;

in >> fl[i].time.minutes;

in >> fl[i].airline;

in >> fl[i].nflight;

in >> fl[i].airport;

}

//Start Duration

auto begin = chrono::steady\_clock::now();

//Write Type of Sort

heapSort(fl, n);

//Calculate Duration

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = chrono::duration\_cast<chrono::milliseconds>(end - begin);

//Write in file

out << "Duration: " << elapsed\_ms.count() << "ms" << endl;

out << "Time \t Airline \t Flight \t Airport" << endl;

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

{

out << fl[i].time.hour << ":" << fl[i].time.minutes << "\t "

<< fl[i].airline << "\t\t "

<< fl[i].nflight << "\t\t "

<< fl[i].airport << endl;

}

// Generation input file in output.txt

/\*srand(time(NULL));

out << n << endl;

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

{

out << rand() % 24 << " " << rand() % 60 << " " << char('A' + rand() % ('Z' - 'A')) << char('A' + rand() % ('Z' - 'A')) << char('A' + rand() % ('Z' - 'A')) << " " << char('A' + rand() % ('Z' - 'A')) << char('A' + rand() % ('Z' - 'A')) << rand() % 10000 << " " << char('A' + rand() % ('Z' - 'A')) << char('A' + rand() % ('Z' - 'A')) << char('A' + rand() % ('Z' - 'A')) << endl;

}\*/

out.close();

in.close();

cout << "End of program" << endl;

return 0;

}

## Файл Flight.h

#pragma once

struct stime

{

int hour;

int minutes;

};

struct flight

{

stime time;

std::string airline;

std::string nflight;

std::string airport;

};

## Файл HeapSort.h

#pragma once

#include <iostream>

#include "Flight.h"

using namespace std;

void heapify(flight arr[], int n, int i)

{

int max = i; // Initialize max as root

int l = 2 \* i + 1; // left

int r = 2 \* i + 2; // right

if (l < n && arr[l].time.hour > arr[max].time.hour)

max = l;

else if (l < n && arr[l].time.hour == arr[max].time.hour)

if (arr[l].time.minutes > arr[max].time.minutes)

max = l;

else if (arr[l].time.minutes == arr[max].time.minutes)

if (arr[l].airline > arr[max].airline)

max = l;

if (r < n && arr[r].time.hour > arr[max].time.hour)

max = r;

else if (r < n && arr[r].time.hour == arr[max].time.hour)

if (arr[r].time.minutes > arr[max].time.minutes)

max = r;

else if (arr[r].time.minutes == arr[max].time.minutes)

if (arr[r].airline > arr[max].airline)

max = r;

// If max is not root

if (max != i)

{

swap(arr[i], arr[max]);

// Recursively heapifybyAirline the affected sub-tree

heapify(arr, n, max);

}

}

void heapSort(flight arr[], int n)

{

// Build heap (rearrange array)

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// One by one extract an element from heap

for (int i = n - 1; i >= 0; i--)

{

// Move current root to end

swap(arr[0], arr[i]);

// call max heapifybyAirline on the reduced heap

heapify(arr, i, 0);

}

}

## Файл InsertionSort.h

#pragma once

#include <iostream>

#include "Flight.h"

using namespace std;

void insertionSort(flight arr[], int n)

{

for (int i = 1; i < n; i++)

for (int j = i; j > 0; j--)

if (arr[j - 1].time.hour > arr[j].time.hour)

swap(arr[j - 1], arr[j]);

else if (arr[j - 1].time.hour == arr[j].time.hour)

if (arr[j - 1].time.minutes > arr[j].time.minutes)

swap(arr[j - 1], arr[j]);

else if (arr[j - 1].time.minutes == arr[j].time.minutes)

if (arr[j - 1].airline > arr[j].airline)

swap(arr[j - 1], arr[j]);

}

# Пример входного файла (на 200 записей):

200

5 56 BOQ HS3988 PSM

17 32 HUB GI8578 RFX

6 29 XNH MQ7113 EAR

9 34 TII OP933 HWJ

13 25 WQL AV5835 RIF

20 33 VVW IO1891 AHD

18 19 KWL KI8707 BAC

19 14 LKR RF1248 VTA

4 45 ULY UE2288 WRC

16 37 FGG DA2713 AMP

12 55 BHO DX2855 BYR

9 51 GWT AL736 WQI

11 49 LVC UN2880 TRG

3 40 ADB KN9730 BMO

7 49 SWJ SV9540 MVE

8 21 POU GC6934 WQH

0 57 MMF FM6588 TDR

20 50 STJ XS7595 XLY

7 40 KSM UO5824 DRE

19 7 QSY MR7778 SBJ

4 30 UUQ XH7184 JTV

1 36 DVB CC2976 DWK

10 39 ILV JE5781 EHV

15 36 XYT HM6861 VKW

3 35 ILL JP9071 CLE

11 3 AAL QL9157 TKI

0 16 XEF WY1257 KUB

22 25 GBL IT9737 YGE

18 43 HXJ VN1618 ELI

23 37 JBK JL8212 EQC

23 9 GRE ES1822 YIK

11 48 VRX RT1852 EGX

13 35 MJH TY850 VFD

6 37 OUY SX6408 UKQ

10 28 AHF NO2025 NNC

10 16 KAU OC8744 QJQ

10 27 QNV LM356 FSV

14 29 XBQ UQ3721 PII

18 11 EOS TC4501 CVP

4 18 XHS II9567 HLA

0 54 NJJ HJ5812 WYY

0 58 PSF IP5722 TIB

22 39 HBM RA250 UMX

9 36 HCT SA85 CQY

13 46 OCA PJ2835 SGU

13 16 QNF OI4864 YXT

0 37 XQN FL8473 LHS

14 32 GID YA2374 EAC

15 12 KCE JQ7477 BRE

5 51 LRD PO1763 EGE

16 54 KXH XI4715 GNY

15 16 IGX VT5227 CFT

8 28 XGW XD3526 NDU

7 33 XCL PQ6090 FCA

13 26 KTY FH9498 MRO

16 59 ELL NW3383 KAT

0 40 MBP FC8268 DJA

7 19 YBD FH594 QRQ

17 31 KGC PY2367 BEJ

16 31 ECB MM8711 WNS

6 59 DGU EP2386 OST

12 5 CSX VK8976 CLX

13 47 HYL UY2460 AUT

6 1 VTU PH6346 AVN

7 44 JJD UW6259 USV

1 59 TPB RD1560 JIV

14 44 AJO DH3292 MLX

8 4 MVB QR5533 MQQ

19 52 GOH MG53 SBT

14 18 NKI IU5197 CKA

22 7 NEH TB8394 XTS

9 54 EHT JS7738 UHP

12 56 PSY OY8168 VYF

6 8 NDF LM7962 RJM

22 39 AEE NG9343 NQS

2 52 KNV PC66 DFP

1 31 AQC KO3726 VXT

11 30 JMW QN2753 XOW

18 59 HLI RE2383 PJW

12 21 TSF LY3161 TKR

10 51 RXN PL2604 GKP

0 29 LPN SC530 DII

14 13 OEL FA1765 PKP

12 21 NWD MI6098 REP

5 57 CRG EQ1617 VFV

7 35 KIL XC1598 YBB

8 41 CQM IN6012 HRT

7 6 BQD DG1451 MTS

9 34 YVR TS1877 OTN

12 51 GGB JB748 MUL

14 6 PMW DJ6321 ITU

7 35 CDA PI1177 VBQ

0 24 VVL TL4885 RWS

1 44 TAQ WD3596 TEU

18 19 OSP FB7304 PGH

6 48 ISW AI8823 IYS

3 10 LQI PC4510 YCG

18 51 TUV MV8059 TSW

19 53 RRS QH3579 FTT

10 34 VJK WA2076 ULC

9 47 SOD PK9655 XLY

3 35 CDA YC3574 GKV

6 15 XBS SL7513 VOX

0 59 QOG FK555 OQG

16 20 EDC PJ8004 BWF

0 9 WGD KK2451 OJE

21 42 QPW UM525 POH

14 31 SRY MQ5631 CIN

7 5 FID TK6071 RNW

4 15 FDB MF9165 KIP

23 6 YFE DI2029 MYC

19 32 MXB YA1623 MGM

3 49 BOT HL1240 OYO

22 20 UPS EI479 FPJ

8 38 JTK WM9670 IKS

1 16 BTV SI6862 ODX

2 53 GYD LJ5737 KAC

0 26 ISU WC3891 BDE

19 16 GDK WW7062 NFR

13 38 VGL NL3359 LWX

21 18 UEY KU9906 LYE

1 48 IEF CC8811 YPI

13 43 TBI HH7161 FIR

5 35 ISS JM6935 CTS

17 33 UKM MY5571 DYE

13 30 VQE FW1359 HNA

20 28 SYD BH8103 PTY

2 58 WCP GW2731 FIH

15 15 PPP IY8155 CAK

10 29 UCD MX2280 AIQ

1 39 YWL OO3917 VKO

16 28 AKG RQ6076 ORT

19 57 WOQ NK5763 TNS

22 13 GBG MT4845 UGL

18 11 EXY OT971 UDE

11 32 ABJ WV1656 TPJ

3 30 WTG CF2224 FOK

14 30 HBN HD1789 NEO

2 0 NPI RW2557 XAX

3 50 WCA LX4196 OUX

0 47 DJD SI3960 CRK

1 48 HKJ LY4141 UHJ

21 40 SUB YR7388 SID

3 34 EBJ KU6791 PPP

10 43 JQQ CJ3683 NHJ

6 58 STY KR5010 YCH

23 51 OXK LK9495 VYG

14 44 MBD AA921 WCD

15 47 LPJ GK342 AGS

11 25 XAI IM561 IHI

13 15 JAB TJ6366 HPQ

18 26 ASH PH3019 JDY

12 59 QCL BE619 WPT

7 19 XSN EG9093 JAC

15 40 GNM OJ2831 BES

8 43 UIR UF5481 KYN

8 30 ISA IJ7236 NYP

1 11 UNU PT7521 CHY

4 20 WLM WW8596 XIK

4 13 RAU WT4830 OTV

21 23 PUG LW8001 HLV

1 40 YPP NM5720 YJV

7 46 TWB FH2635 OVD

18 48 KKY LU2167 YUV

17 49 THF WP688 MKV

6 48 HSK XP3737 WIR

17 27 AMN CS1559 LGT

1 17 XQB JM4155 XUG

13 49 XNR RJ3220 NIH

10 43 OGH FE9231 TUR

7 37 QKG OR9798 VFX

19 56 LKP RS4569 PTX

19 26 OYD DN7191 HMM

18 35 QUG VN919 AHT

5 0 NBK NA5956 BJE

4 6 LRK HG4850 EOA

21 24 DSX ON9556 EJP

1 58 GJD CB3716 KEX

19 38 TEQ EI5136 JKF

3 10 NPF TU4137 BRD

4 56 ONH RF4528 WDB

23 18 FIC AQ704 SJJ

16 38 GWP LX8556 TUS

23 35 UGI AG2121 TOO

7 50 BEJ QI4376 CHM

21 11 SCD BE8260 HTQ

4 47 UMK HE1338 NUF

18 33 OFN GI3238 YOD

5 12 NVL TK8384 UMK

12 26 LNI CY1611 SJW

10 54 IHQ CA3429 LAA

11 4 IEO DK6887 NRL

13 19 NHU VR8650 WHN

22 26 BRN IK2225 XWW

22 14 ASN WX9197 UIV

23 57 KYB DV9139 PWL

20 4 CRA KF309 EFF

20 24 WON YJ6482 VXN

4 23 CPW EJ6822 TYR

18 3 CVT BD8314 AGP

# Пример выходного файла (на 200 записей):

## Пирамидальная сортировка

**output.txt**

Duration: 14ms

Time Airline Flight Airport

0:9 WGD KK2451 OJE

0:16 XEF WY1257 KUB

0:24 VVL TL4885 RWS

0:26 ISU WC3891 BDE

0:29 LPN SC530 DII

0:37 XQN FL8473 LHS

0:40 MBP FC8268 DJA

0:47 DJD SI3960 CRK

0:54 NJJ HJ5812 WYY

0:57 MMF FM6588 TDR

0:58 PSF IP5722 TIB

0:59 QOG FK555 OQG

1:11 UNU PT7521 CHY

1:16 BTV SI6862 ODX

1:17 XQB JM4155 XUG

1:31 AQC KO3726 VXT

1:36 DVB CC2976 DWK

1:39 YWL OO3917 VKO

1:40 YPP NM5720 YJV

1:44 TAQ WD3596 TEU

1:48 HKJ LY4141 UHJ

1:48 IEF CC8811 YPI

1:58 GJD CB3716 KEX

1:59 TPB RD1560 JIV

2:0 NPI RW2557 XAX

2:52 KNV PC66 DFP

2:53 GYD LJ5737 KAC

2:58 WCP GW2731 FIH

3:10 LQI PC4510 YCG

3:10 NPF TU4137 BRD

3:30 WTG CF2224 FOK

3:34 EBJ KU6791 PPP

3:35 CDA YC3574 GKV

3:35 ILL JP9071 CLE

3:40 ADB KN9730 BMO

3:49 BOT HL1240 OYO

3:50 WCA LX4196 OUX

4:6 LRK HG4850 EOA

4:13 RAU WT4830 OTV

4:15 FDB MF9165 KIP

4:18 XHS II9567 HLA

4:20 WLM WW8596 XIK

4:23 CPW EJ6822 TYR

4:30 UUQ XH7184 JTV

4:45 ULY UE2288 WRC

4:47 UMK HE1338 NUF

4:56 ONH RF4528 WDB

5:0 NBK NA5956 BJE

5:12 NVL TK8384 UMK

5:35 ISS JM6935 CTS

5:51 LRD PO1763 EGE

5:56 BOQ HS3988 PSM

5:57 CRG EQ1617 VFV

6:1 VTU PH6346 AVN

6:8 NDF LM7962 RJM

6:15 XBS SL7513 VOX

6:29 XNH MQ7113 EAR

6:37 OUY SX6408 UKQ

6:48 HSK XP3737 WIR

6:48 ISW AI8823 IYS

6:58 STY KR5010 YCH

6:59 DGU EP2386 OST

7:5 FID TK6071 RNW

7:6 BQD DG1451 MTS

7:19 XSN EG9093 JAC

7:19 YBD FH594 QRQ

7:33 XCL PQ6090 FCA

7:35 CDA PI1177 VBQ

7:35 KIL XC1598 YBB

7:37 QKG OR9798 VFX

7:40 KSM UO5824 DRE

7:44 JJD UW6259 USV

7:46 TWB FH2635 OVD

7:49 SWJ SV9540 MVE

7:50 BEJ QI4376 CHM

8:4 MVB QR5533 MQQ

8:21 POU GC6934 WQH

8:28 XGW XD3526 NDU

8:30 ISA IJ7236 NYP

8:38 JTK WM9670 IKS

8:41 CQM IN6012 HRT

8:43 UIR UF5481 KYN

9:34 TII OP933 HWJ

9:34 YVR TS1877 OTN

9:36 HCT SA85 CQY

9:47 SOD PK9655 XLY

9:51 GWT AL736 WQI

9:54 EHT JS7738 UHP

10:16 KAU OC8744 QJQ

10:27 QNV LM356 FSV

10:28 AHF NO2025 NNC

10:29 UCD MX2280 AIQ

10:34 VJK WA2076 ULC

10:39 ILV JE5781 EHV

10:43 JQQ CJ3683 NHJ

10:43 OGH FE9231 TUR

10:51 RXN PL2604 GKP

10:54 IHQ CA3429 LAA

11:3 AAL QL9157 TKI

11:4 IEO DK6887 NRL

11:25 XAI IM561 IHI

11:30 JMW QN2753 XOW

11:32 ABJ WV1656 TPJ

11:48 VRX RT1852 EGX

11:49 LVC UN2880 TRG

12:5 CSX VK8976 CLX

12:21 NWD MI6098 REP

12:21 TSF LY3161 TKR

12:26 LNI CY1611 SJW

12:51 GGB JB748 MUL

12:55 BHO DX2855 BYR

12:56 PSY OY8168 VYF

12:59 QCL BE619 WPT

13:15 JAB TJ6366 HPQ

13:16 QNF OI4864 YXT

13:19 NHU VR8650 WHN

13:25 WQL AV5835 RIF

13:26 KTY FH9498 MRO

13:30 VQE FW1359 HNA

13:35 MJH TY850 VFD

13:38 VGL NL3359 LWX

13:43 TBI HH7161 FIR

13:46 OCA PJ2835 SGU

13:47 HYL UY2460 AUT

13:49 XNR RJ3220 NIH

14:6 PMW DJ6321 ITU

14:13 OEL FA1765 PKP

14:18 NKI IU5197 CKA

14:29 XBQ UQ3721 PII

14:30 HBN HD1789 NEO

14:31 SRY MQ5631 CIN

14:32 GID YA2374 EAC

14:44 AJO DH3292 MLX

14:44 MBD AA921 WCD

15:12 KCE JQ7477 BRE

15:15 PPP IY8155 CAK

15:16 IGX VT5227 CFT

15:36 XYT HM6861 VKW

15:40 GNM OJ2831 BES

15:47 LPJ GK342 AGS

16:20 EDC PJ8004 BWF

16:28 AKG RQ6076 ORT

16:31 ECB MM8711 WNS

16:37 FGG DA2713 AMP

16:38 GWP LX8556 TUS

16:54 KXH XI4715 GNY

16:59 ELL NW3383 KAT

17:27 AMN CS1559 LGT

17:31 KGC PY2367 BEJ

17:32 HUB GI8578 RFX

17:33 UKM MY5571 DYE

17:49 THF WP688 MKV

18:3 CVT BD8314 AGP

18:11 EOS TC4501 CVP

18:11 EXY OT971 UDE

18:19 KWL KI8707 BAC

18:19 OSP FB7304 PGH

18:26 ASH PH3019 JDY

18:33 OFN GI3238 YOD

18:35 QUG VN919 AHT

18:43 HXJ VN1618 ELI

18:48 KKY LU2167 YUV

18:51 TUV MV8059 TSW

18:59 HLI RE2383 PJW

19:7 QSY MR7778 SBJ

19:14 LKR RF1248 VTA

19:16 GDK WW7062 NFR

19:26 OYD DN7191 HMM

19:32 MXB YA1623 MGM

19:38 TEQ EI5136 JKF

19:52 GOH MG53 SBT

19:53 RRS QH3579 FTT

19:56 LKP RS4569 PTX

19:57 WOQ NK5763 TNS

20:4 CRA KF309 EFF

20:24 WON YJ6482 VXN

20:28 SYD BH8103 PTY

20:33 VVW IO1891 AHD

20:50 STJ XS7595 XLY

21:11 SCD BE8260 HTQ

21:18 UEY KU9906 LYE

21:23 PUG LW8001 HLV

21:24 DSX ON9556 EJP

21:40 SUB YR7388 SID

21:42 QPW UM525 POH

22:7 NEH TB8394 XTS

22:13 GBG MT4845 UGL

22:14 ASN WX9197 UIV

22:20 UPS EI479 FPJ

22:25 GBL IT9737 YGE

22:26 BRN IK2225 XWW

22:39 AEE NG9343 NQS

22:39 HBM RA250 UMX

23:6 YFE DI2029 MYC

23:9 GRE ES1822 YIK

23:18 FIC AQ704 SJJ

23:35 UGI AG2121 TOO

23:37 JBK JL8212 EQC

23:51 OXK LK9495 VYG

23:57 KYB DV9139 PWL

## Сортировка вставками

**output.txt**

Duration: 72ms

Time Airline Flight Airport

0:9 WGD KK2451 OJE

0:16 XEF WY1257 KUB

0:24 VVL TL4885 RWS

0:26 ISU WC3891 BDE

0:29 LPN SC530 DII

0:37 XQN FL8473 LHS

0:40 MBP FC8268 DJA

0:47 DJD SI3960 CRK

0:54 NJJ HJ5812 WYY

0:57 MMF FM6588 TDR

0:58 PSF IP5722 TIB

0:59 QOG FK555 OQG

1:11 UNU PT7521 CHY

1:16 BTV SI6862 ODX

1:17 XQB JM4155 XUG

1:31 AQC KO3726 VXT

1:36 DVB CC2976 DWK

1:39 YWL OO3917 VKO

1:40 YPP NM5720 YJV

1:44 TAQ WD3596 TEU

1:48 HKJ LY4141 UHJ

1:48 IEF CC8811 YPI

1:58 GJD CB3716 KEX

1:59 TPB RD1560 JIV

2:0 NPI RW2557 XAX

2:52 KNV PC66 DFP

2:53 GYD LJ5737 KAC

2:58 WCP GW2731 FIH

3:10 LQI PC4510 YCG

3:10 NPF TU4137 BRD

3:30 WTG CF2224 FOK

3:34 EBJ KU6791 PPP

3:35 CDA YC3574 GKV

3:35 ILL JP9071 CLE

3:40 ADB KN9730 BMO

3:49 BOT HL1240 OYO

3:50 WCA LX4196 OUX

4:6 LRK HG4850 EOA

4:13 RAU WT4830 OTV

4:15 FDB MF9165 KIP

4:18 XHS II9567 HLA

4:20 WLM WW8596 XIK

4:23 CPW EJ6822 TYR

4:30 UUQ XH7184 JTV

4:45 ULY UE2288 WRC

4:47 UMK HE1338 NUF

4:56 ONH RF4528 WDB

5:0 NBK NA5956 BJE

5:12 NVL TK8384 UMK

5:35 ISS JM6935 CTS

5:51 LRD PO1763 EGE

5:56 BOQ HS3988 PSM

5:57 CRG EQ1617 VFV

6:1 VTU PH6346 AVN

6:8 NDF LM7962 RJM

6:15 XBS SL7513 VOX

6:29 XNH MQ7113 EAR

6:37 OUY SX6408 UKQ

6:48 HSK XP3737 WIR

6:48 ISW AI8823 IYS

6:58 STY KR5010 YCH

6:59 DGU EP2386 OST

7:5 FID TK6071 RNW

7:6 BQD DG1451 MTS

7:19 XSN EG9093 JAC

7:19 YBD FH594 QRQ

7:33 XCL PQ6090 FCA

7:35 CDA PI1177 VBQ

7:35 KIL XC1598 YBB

7:37 QKG OR9798 VFX

7:40 KSM UO5824 DRE

7:44 JJD UW6259 USV

7:46 TWB FH2635 OVD

7:49 SWJ SV9540 MVE

7:50 BEJ QI4376 CHM

8:4 MVB QR5533 MQQ

8:21 POU GC6934 WQH

8:28 XGW XD3526 NDU

8:30 ISA IJ7236 NYP

8:38 JTK WM9670 IKS

8:41 CQM IN6012 HRT

8:43 UIR UF5481 KYN

9:34 TII OP933 HWJ

9:34 YVR TS1877 OTN

9:36 HCT SA85 CQY

9:47 SOD PK9655 XLY

9:51 GWT AL736 WQI

9:54 EHT JS7738 UHP

10:16 KAU OC8744 QJQ

10:27 QNV LM356 FSV

10:28 AHF NO2025 NNC

10:29 UCD MX2280 AIQ

10:34 VJK WA2076 ULC

10:39 ILV JE5781 EHV

10:43 JQQ CJ3683 NHJ

10:43 OGH FE9231 TUR

10:51 RXN PL2604 GKP

10:54 IHQ CA3429 LAA

11:3 AAL QL9157 TKI

11:4 IEO DK6887 NRL

11:25 XAI IM561 IHI

11:30 JMW QN2753 XOW

11:32 ABJ WV1656 TPJ

11:48 VRX RT1852 EGX

11:49 LVC UN2880 TRG

12:5 CSX VK8976 CLX

12:21 NWD MI6098 REP

12:21 TSF LY3161 TKR

12:26 LNI CY1611 SJW

12:51 GGB JB748 MUL

12:55 BHO DX2855 BYR

12:56 PSY OY8168 VYF

12:59 QCL BE619 WPT

13:15 JAB TJ6366 HPQ

13:16 QNF OI4864 YXT

13:19 NHU VR8650 WHN

13:25 WQL AV5835 RIF

13:26 KTY FH9498 MRO

13:30 VQE FW1359 HNA

13:35 MJH TY850 VFD

13:38 VGL NL3359 LWX

13:43 TBI HH7161 FIR

13:46 OCA PJ2835 SGU

13:47 HYL UY2460 AUT

13:49 XNR RJ3220 NIH

14:6 PMW DJ6321 ITU

14:13 OEL FA1765 PKP

14:18 NKI IU5197 CKA

14:29 XBQ UQ3721 PII

14:30 HBN HD1789 NEO

14:31 SRY MQ5631 CIN

14:32 GID YA2374 EAC

14:44 AJO DH3292 MLX

14:44 MBD AA921 WCD

15:12 KCE JQ7477 BRE

15:15 PPP IY8155 CAK

15:16 IGX VT5227 CFT

15:36 XYT HM6861 VKW

15:40 GNM OJ2831 BES

15:47 LPJ GK342 AGS

16:20 EDC PJ8004 BWF

16:28 AKG RQ6076 ORT

16:31 ECB MM8711 WNS

16:37 FGG DA2713 AMP

16:38 GWP LX8556 TUS

16:54 KXH XI4715 GNY

16:59 ELL NW3383 KAT

17:27 AMN CS1559 LGT

17:31 KGC PY2367 BEJ

17:32 HUB GI8578 RFX

17:33 UKM MY5571 DYE

17:49 THF WP688 MKV

18:3 CVT BD8314 AGP

18:11 EOS TC4501 CVP

18:11 EXY OT971 UDE

18:19 KWL KI8707 BAC

18:19 OSP FB7304 PGH

18:26 ASH PH3019 JDY

18:33 OFN GI3238 YOD

18:35 QUG VN919 AHT

18:43 HXJ VN1618 ELI

18:48 KKY LU2167 YUV

18:51 TUV MV8059 TSW

18:59 HLI RE2383 PJW

19:7 QSY MR7778 SBJ

19:14 LKR RF1248 VTA

19:16 GDK WW7062 NFR

19:26 OYD DN7191 HMM

19:32 MXB YA1623 MGM

19:38 TEQ EI5136 JKF

19:52 GOH MG53 SBT

19:53 RRS QH3579 FTT

19:56 LKP RS4569 PTX

19:57 WOQ NK5763 TNS

20:4 CRA KF309 EFF

20:24 WON YJ6482 VXN

20:28 SYD BH8103 PTY

20:33 VVW IO1891 AHD

20:50 STJ XS7595 XLY

21:11 SCD BE8260 HTQ

21:18 UEY KU9906 LYE

21:23 PUG LW8001 HLV

21:24 DSX ON9556 EJP

21:40 SUB YR7388 SID

21:42 QPW UM525 POH

22:7 NEH TB8394 XTS

22:13 GBG MT4845 UGL

22:14 ASN WX9197 UIV

22:20 UPS EI479 FPJ

22:25 GBL IT9737 YGE

22:26 BRN IK2225 XWW

22:39 AEE NG9343 NQS

22:39 HBM RA250 UMX

23:6 YFE DI2029 MYC

23:9 GRE ES1822 YIK

23:18 FIC AQ704 SJJ

23:35 UGI AG2121 TOO

23:37 JBK JL8212 EQC

23:51 OXK LK9495 VYG

23:57 KYB DV9139 PWL

# Тесты

Рассмотрим 3 случая:

Мы будем автоматически генерировать случайные данные.

## 1. 200 случайно сгенерированных записей:

**Количество записей: 200**

**Пирамидальная сортировка:**

Время, затраченное на сортировку: 14ms

**Сортировка вставками:**

Время, затраченное на сортировку: 72ms

Пирамидальная сортировка в 5,14 раза быстрее, сортировка Вставками.

## 2. 10000 случайно сгенерированных записей:

**Количество записей: 10000**

**Пирамидальная сортировка:**

Время, затраченное на сортировку: 1020ms

**Сортировка вставками:**

Время, затраченное на сортировку: 171387ms

Пирамидальная сортировка в 168 раза быстрее, сортировка Вставками.

## 3. Входные данные уже отсортированы:

**Количество записей: 100**

**Пирамидальная сортировка:**

Время, затраченное на сортировку: 4ms

**Сортировка вставками:**

Время, затраченное на сортировку: 0ms

## 4. Входные данные отсортированы в обратном порядке:

**Количество записей: 100**

**Пирамидальная сортировка:**

Время, затраченное на сортировку: 4ms

**Сортировка вставками:**

Время, затраченное на сортировку: 58ms

## 5. Входные данные содержат 100 одинаковых записей:

**Количество записей: 100**

**Пирамидальная сортировка:**

Время, затраченное на сортировку: 0ms

**Сортировка вставками:**

Время, затраченное на сортировку: 1ms

# Проверка устойчивости с обоснованием результатов

**Входной файл, содержащий одинаковые для сортировки по времени и коду авиакомпании, но с разным кодом рейса:**

17 42 MID YT6501 JRQ

17 42 MID YT6502 JRQ

**Пирамидальная сортировка**

Выходной файл:

17:42 MID YT6502 JRQ

17:42 MID YT6501 JRQ

**Сортировка вставками**

Выходной файл:

17:42 MID YT6501 JRQ

17:42 MID YT6502 JRQ

## Результат:

Как мы видим, Сортировка вставками является устойчивой, а Пирамидальная сортировка является неустойчивой.

# Выводы:

Сортировка вставками работает быстрее тогда, когда нужно отсортировать небольшое количество записей. Также эта сортировка показывает хорошую производительность, когда нужно отсортировать уже упорядоченные записи.

Пирамидальная сортировка работает намного быстрее, чем Сортировка вставками, при больших количествах случайных данных и в случае, когда данные отсортированы в обратном порядке.