

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дата сдачи: 06.03.2023

| | |
|-------------------------|----|
| Описание задачи. | 2 |
| Описание метода/модели. | 2 |
| Выполнение задачи. | 3 |
| Заключение. | 17 |

Описание задачи.

Необходимо реализовать метод быстрой сортировки.

Для реализованного метода сортировки необходимо провести серию тестов для всех значений N из списка (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000), при этом:

- в каждом тесте необходимо по 20 раз генерировать вектор, состоящий из N элементов
- каждый элемент массива заполняется случайным числом с плавающей запятой от -1 до 1

На основании статьи реализовать проверки негативных случаев и устроить на них серии тестов аналогичные второму пункту:

- Отсортированный массив
- Массив с одинаковыми элементами
- Массив с максимальным количеством сравнений при выборе среднего элемента в качестве опорного
- Массив с максимальным количеством сравнений при детерминированном выборе опорного элемента

При работе сортировки подсчитать количество вызовов рекурсивной функции, и высоту рекурсивного стека. Построить график худшего, лучшего, и среднего случая для каждой серии тестов.

Для каждой серии тестов построить график худшего случая.

Подобрать такую константу c , чтобы график функции $c \cdot n \cdot \log(n)$ находился близко к графику худшего случая, если возможно построить такой график.

Проанализировать полученные графики и определить есть ли на них следы деградации метода относительно своей средней сложности.

Описание метода/модели.

Быстрая сортировка является одним из самых быстрых алгоритмов сортировки массивов.

Общая идея алгоритма состоит в следующем:

- Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
- Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньше опорного», «равные» и «большие».
- Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

На практике массив обычно делят не на три, а на две части: например, «меньшие опорного» и «равные и большие»; такой подход в общем случае эффективнее, так как упрощает алгоритм разделения.

Выполнение задачи.

Для реализации данного метода сортировки использовался язык программирования C++.

Алгоритм быстрой сортировки

1. Функция quickSort принимает на вход вектор *v* элементов, начальный и конечный элементы сортировки *begin* и *last*, а также глубина данной ветки стека *depth*.

Функция начинается с увеличения значений количества рекурсий *count_of_recursion* и глубины стека *height*.

Далее функция проверяется является ли начальный элемент сортировки меньше конечного.

Если да, то с помощью функции *partition* находится индекс опорного элемента. После чего вызываются две рекурсивные функции *quickSort* для части вектора до опорного и после него.

```
void quickSort(vector<double> v, int begin, int last, int depth) {
    double pivot;
    count_of_recursion++;
    if (depth > height) height = depth;
    if (begin < last) {
        pivot = partition(v, begin, last);
        quickSort(v, begin, pivot - 1, depth + 1);
        quickSort(v, pivot + 1, last, depth + 1);
    }
}
```

2. Функция *partition* принимает на вход вектор элементов *v*, начальный и конечный элементы сортировки *begin* и *last*.

Выбирается значение опорного элемента *pivot_value*. Далее идет перестановка элементов относительно опорного элемента. После итерации свободный элемент заменяется элементом с индексом *i+1*. Функция возвращает индекс опорного элемента.

```
int partition(vector<double> v, int begin, int last) {
    double pivot_value = v[last];
    int i = begin - 1;
    double temp;
    for (int j = begin; j < last; j++) {
        if (v[j] <= pivot_value) {
            i++;
            temp = v[i];
            v[i] = v[j];
            v[j] = temp;
        }
    }
    temp = v[i + 1];
    v[i + 1] = v[last];
    v[last] = temp;
    return i + 1;
}
```

Начальные значения

1. В качестве начальных значений были заданы:
 - Количество тестов $M = 20$;
 - Массив длин вектора $N[] = \{1000, 2000, 4000, 8000, 16000, 32000\}$;
2. Задание вектора элементов

```
// генерация случайных чисел от -1 до 1
mt19937 engine(time(_Time: 0));
uniform_real_distribution<double> gen(-1.0, 1.0);
for (int el = 0; el < N[j]; el++) {
    v.push_back(_Val: gen(engine));
}
```

В качестве вектора из одинаковых элементов, был создан вектор из нулей.

3. Перед первым вызовом функции quickSort обнуляются значения глубины стека и количества рекурсий (depth, height и count_of_recursion). Также задаются начальные значения индексов начального и конечного элементов сортировки (begin = 0, last = v.size() - 1).

Запись в файл

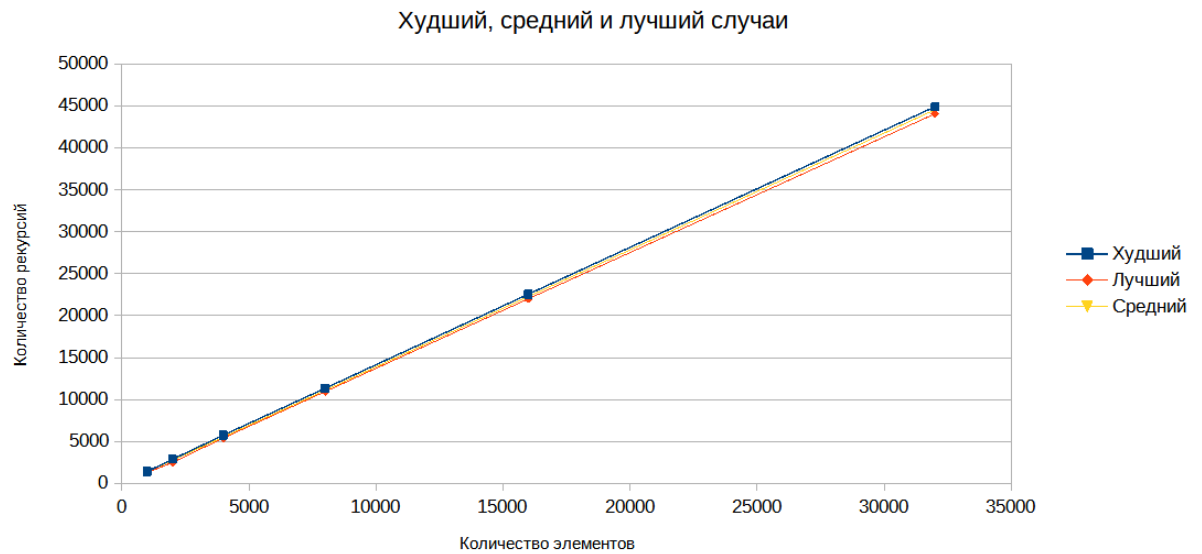
1. С помощью функции writer полученные результаты записываются в файл. А именно: затраченное на сортировку время в миллисекундах, количество рекурсий и максимальная глубина стека.

Обычный вектор

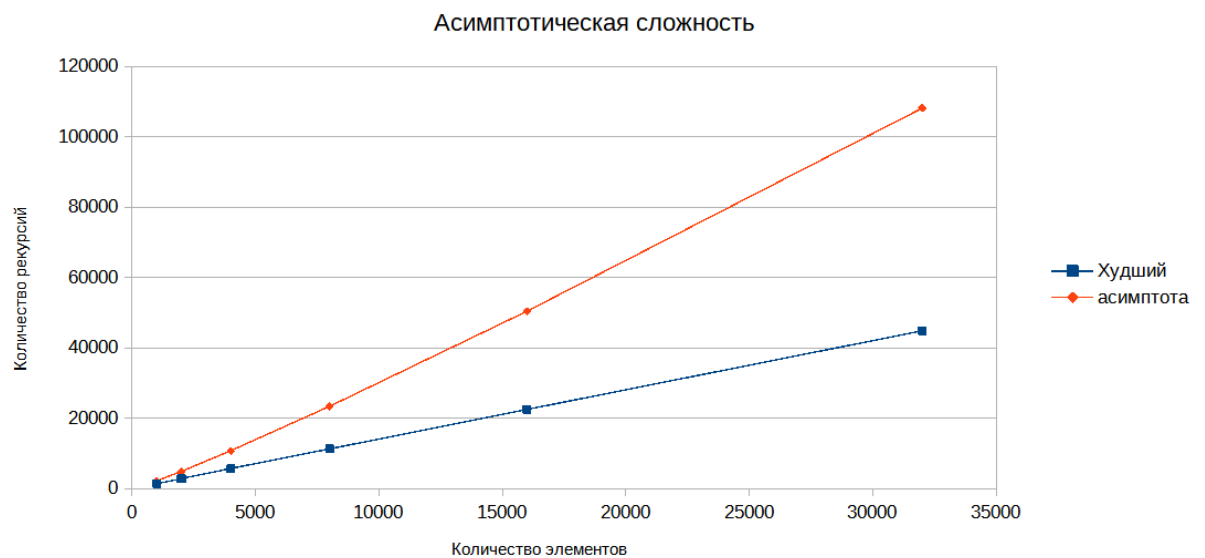
1. Обычный вектор элементов был получен с помощью генерации случайных чисел от -1 до 1. Было проведено 20 тестов для различных размеров вектора (а именно, 1000, 2000, 4000, 8000, 16000, 32000).
2. Количество рекурсий в зависимости от длины вектора.
 - Полученные результаты

| Количество рекурсий | | | | | | |
|---------------------|--------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 1364 | 2754 | 5756 | 11018 | 22544 | 44538 |
| 2 | 1300 | 2714 | 5580 | 11156 | 22266 | 44394 |
| 3 | 1318 | 2694 | 5482 | 11002 | 22360 | 44590 |
| 4 | 1400 | 2504 | 5706 | 11188 | 22212 | 44754 |
| 5 | 1370 | 2796 | 5414 | 11028 | 22306 | 44594 |
| 6 | 1396 | 2688 | 5438 | 11038 | 22194 | 44534 |
| 7 | 1374 | 2830 | 5560 | 11080 | 22380 | 44360 |
| 8 | 1390 | 2776 | 5478 | 11056 | 22462 | 44632 |
| 9 | 1422 | 2788 | 5582 | 11248 | 22162 | 44704 |
| 10 | 1392 | 2856 | 5536 | 11224 | 22288 | 44836 |
| 11 | 1416 | 2718 | 5466 | 11098 | 22214 | 44050 |
| 12 | 1386 | 2698 | 5504 | 11018 | 22316 | 44374 |
| 13 | 1428 | 2748 | 5570 | 11032 | 21994 | 44880 |
| 14 | 1418 | 2840 | 5500 | 11004 | 22120 | 44042 |
| 15 | 1372 | 2756 | 5496 | 11102 | 22032 | 44302 |
| 16 | 1406 | 2748 | 5520 | 10942 | 22084 | 44712 |
| 17 | 1408 | 2884 | 5470 | 11148 | 22308 | 44460 |
| 18 | 1372 | 2788 | 5514 | 11328 | 22264 | 44812 |
| 19 | 1394 | 2794 | 5708 | 11002 | 22154 | 44506 |
| 20 | 1404 | 2740 | 5542 | 11096 | 22440 | 44322 |
| Худший | 1428 | 2884 | 5756 | 11328 | 22544 | 44880 |
| Лучший | 1300 | 2504 | 5414 | 10942 | 21994 | 44042 |
| Средний | 1386,5 | 2755,7 | 5541,1 | 11090,4 | 22255 | 44519,8 |
| 0,75 | 2250 | 4951,544993 | 10806,17997 | 23418,53992 | 50449,43979 | 108123,5995 |

- График худшего, среднего и лучшего случаев для количества рекурсий в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,75$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .

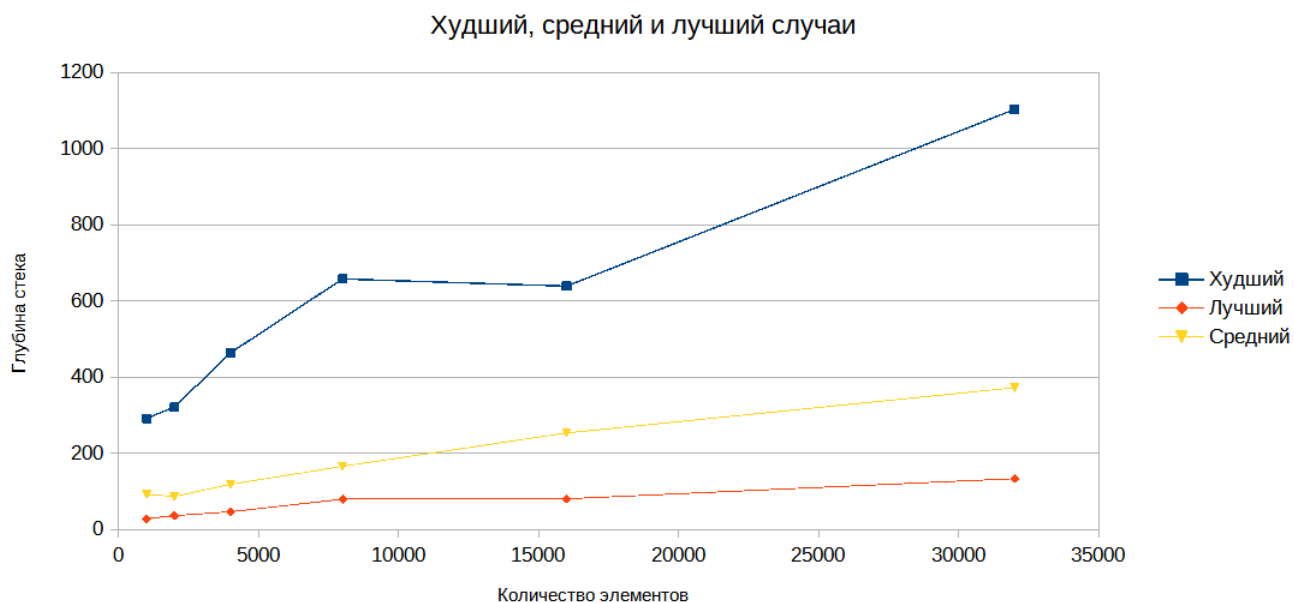


3. Глубина стека в зависимости от длины вектора.

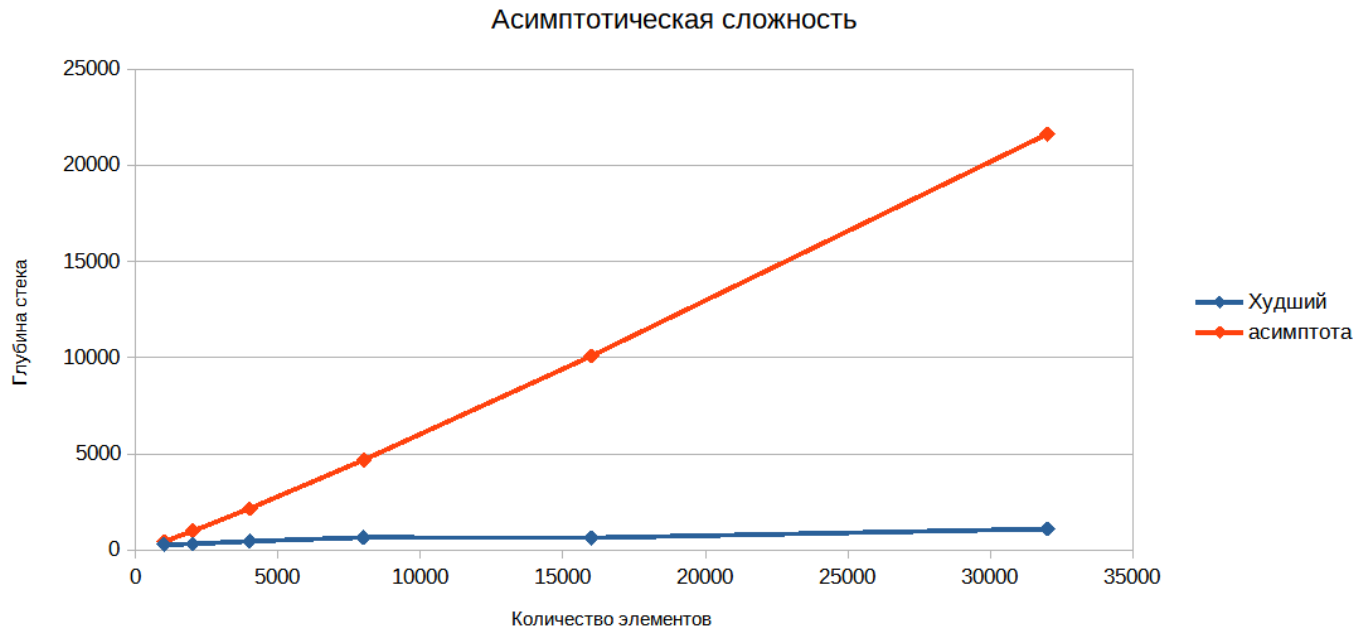
- Полученные результаты

| Глубина стека | | | | | | |
|---------------|------|-------------|-------------|-------------|-------------|------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 44 | 45 | 464 | 107 | 636 | 245 |
| 2 | 111 | 44 | 97 | 185 | 251 | 391 |
| 3 | 224 | 110 | 67 | 158 | 292 | 350 |
| 4 | 43 | 322 | 250 | 116 | 149 | 284 |
| 5 | 117 | 72 | 63 | 105 | 81 | 270 |
| 6 | 71 | 37 | 69 | 84 | 114 | 223 |
| 7 | 66 | 88 | 119 | 105 | 210 | 163 |
| 8 | 51 | 69 | 104 | 187 | 639 | 332 |
| 9 | 43 | 47 | 114 | 181 | 170 | 160 |
| 10 | 113 | 101 | 69 | 80 | 251 | 261 |
| 11 | 29 | 50 | 80 | 110 | 99 | 1103 |
| 12 | 59 | 44 | 75 | 110 | 189 | 188 |
| 13 | 50 | 98 | 55 | 148 | 142 | 886 |
| 14 | 292 | 48 | 139 | 254 | 262 | 159 |
| 15 | 78 | 46 | 47 | 95 | 196 | 154 |
| 16 | 56 | 241 | 61 | 111 | 264 | 350 |
| 17 | 58 | 101 | 96 | 274 | 292 | 815 |
| 18 | 244 | 48 | 85 | 163 | 452 | 164 |
| 19 | 51 | 77 | 237 | 93 | 142 | 134 |
| 20 | 54 | 55 | 88 | 658 | 243 | 820 |
| Худший | 292 | 322 | 464 | 658 | 639 | 1103 |
| Лучший | 29 | 37 | 47 | 80 | 81 | 134 |
| Средний | 92,7 | 87,15 | 118,95 | 166,2 | 253,7 | 372,6 |
| 0,15 | 450 | 990,3089987 | 2161,235995 | 4683,707984 | 10089,88796 | 21624,7199 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,15$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .

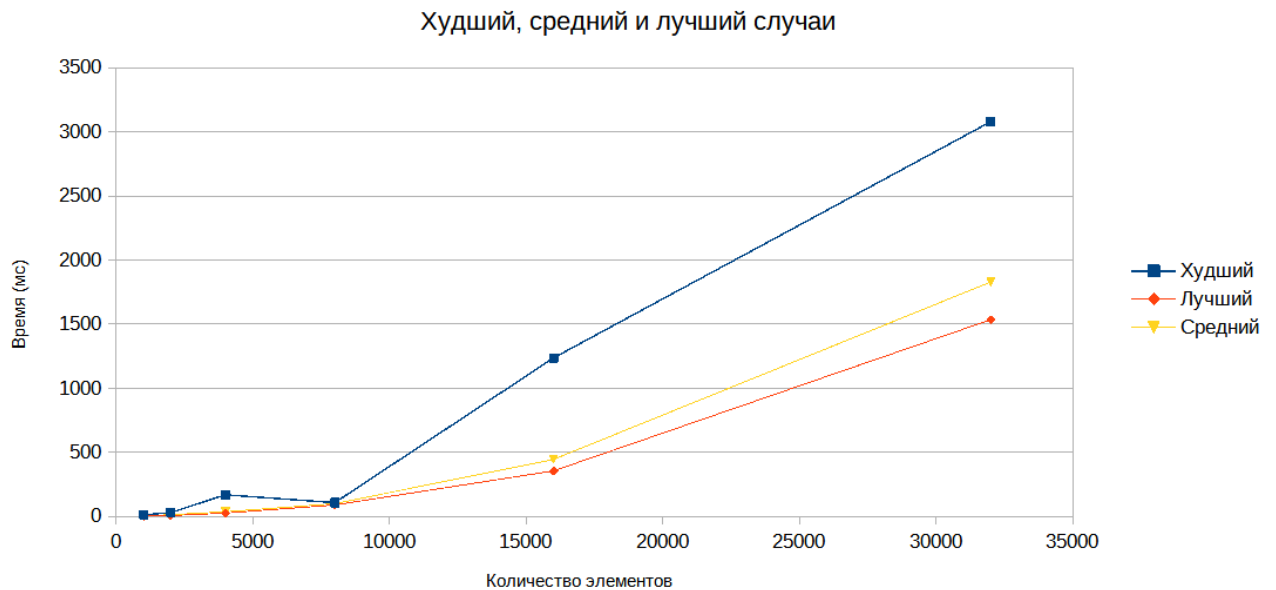


4. Время сортировки в зависимости от длины вектора.

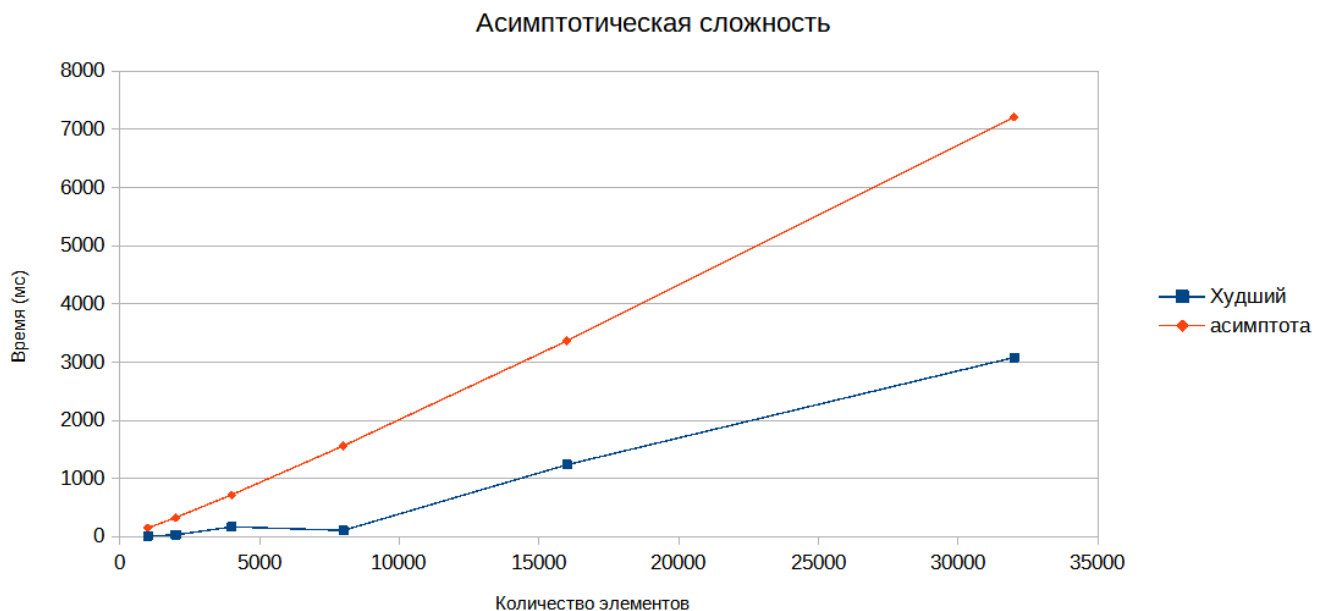
- Полученные результаты

| | Время | | | | | |
|---------|----------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 3,397 | 8,1322 | 170,444 | 106,02 | 1236,91 | 3078,93 |
| 2 | 3,2865 | 8,9188 | 33,2737 | 107,126 | 399,344 | 1669,7 |
| 3 | 3,8518 | 9,8333 | 30,8824 | 108,971 | 410,018 | 1535,46 |
| 4 | 3,5569 | 23,301 | 33,9056 | 99,329 | 394,424 | 1799,56 |
| 5 | 3,4204 | 10,4419 | 29,5102 | 99,5868 | 360,374 | 1708,51 |
| 6 | 4,3285 | 10,8593 | 31,5994 | 97,5601 | 356,308 | 1761,55 |
| 7 | 3,1976 | 13,5181 | 28,5977 | 97,6348 | 385,7 | 1636,65 |
| 8 | 4,236 | 13,84 | 28,5477 | 103,153 | 412,012 | 2315,36 |
| 9 | 5,3561 | 14,4753 | 27,8427 | 105,914 | 369,829 | 1848,47 |
| 10 | 4,682 | 28,1128 | 27,7055 | 101,116 | 368,162 | 1844,87 |
| 11 | 4,4094 | 14,0495 | 27,3303 | 104,716 | 361,993 | 1896,23 |
| 12 | 3,7259 | 13,7336 | 28,5278 | 98,8447 | 429,007 | 1662,85 |
| 13 | 5,161 | 14,4811 | 30,6779 | 98,8908 | 410,011 | 1767,16 |
| 14 | 5,8554 | 16,9836 | 27,7253 | 107,604 | 413,563 | 1644,75 |
| 15 | 6,0157 | 14,6982 | 27,581 | 97,7083 | 418,322 | 1626,27 |
| 16 | 3,8967 | 16,7001 | 31,7712 | 96,9641 | 436,051 | 1831,62 |
| 17 | 7,6821 | 31,3083 | 39,5099 | 99,6397 | 466,061 | 1811,95 |
| 18 | 13,1186 | 21,574 | 32,3032 | 98,619 | 450,23 | 1762,63 |
| 19 | 6,324 | 13,8389 | 33,6224 | 91,0367 | 412,125 | 1644,38 |
| 20 | 9,3255 | 12,953 | 30,4035 | 106,738 | 424,76 | 1742,31 |
| Худший | 13,1186 | 31,3083 | 170,444 | 108,971 | 1236,91 | 3078,93 |
| Лучший | 3,1976 | 8,1322 | 27,3303 | 91,0367 | 356,308 | 1535,46 |
| Средний | 5,241355 | 15,58765 | 37,58807 | 101,3586 | 445,7602 | 1829,4605 |
| 0,05 | 150 | 330,1029996 | 720,4119983 | 1561,235995 | 3363,295986 | 7208,239965 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,05$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .



Отсортированный вектор

1. В качестве отсортированного вектора берется тот же вектор, который был создан ранее и отсортирован быстрой сортировкой. Было также проведено по 20 тестов для различных длин вектора. Результаты также были записаны в файл.
2. Проанализировав получившиеся результаты, можно отметить, что и количество рекурсий, и максимальная глубина стека полностью совпадает с сортировкой обычного вектора. Из этого

следует, что графики количества рекурсий и глубины стека в зависимости от количества элементов в векторе будут аналогичными как и в обычном векторе.

Вектор с одинаковыми элементами (нулевой вектор)

1. Вектор из нулей был получен с помощью изначального задания размера вектора. Было проведено по одному тесту для различных размеров вектора (а именно, 100, 200, 400, 800, 1000). Так как при проведении нескольких тестов на одинаковых по размеру векторах получалось одинаковое количество рекурсий и глубина стека, стало понятно, что есть смысл проводить только по одному тесту на каждую длину.

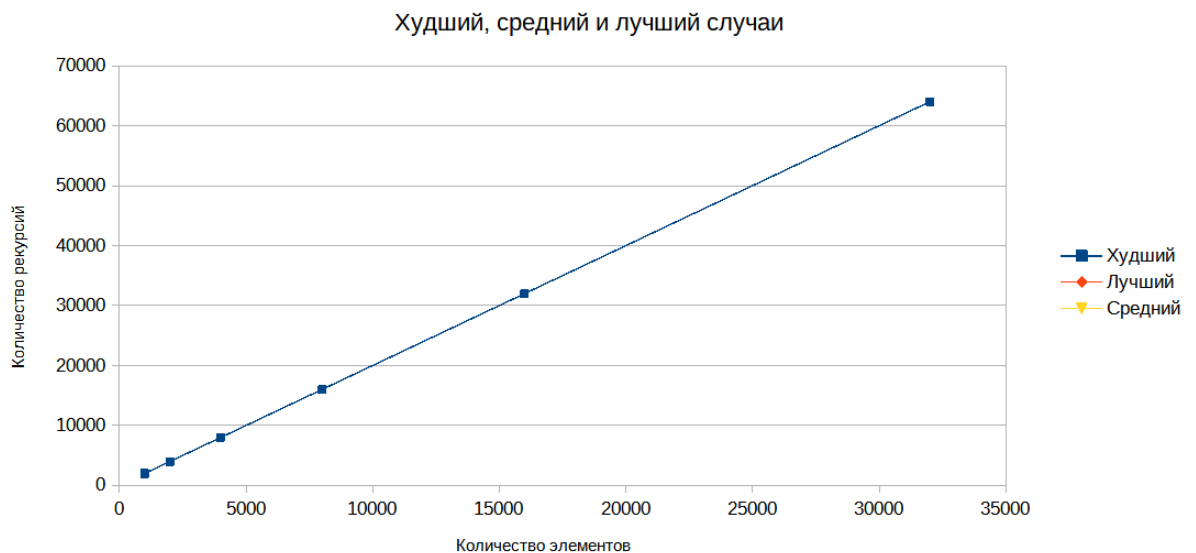
2. Количество рекурсий в зависимости от длины вектора.

- Полученные результаты

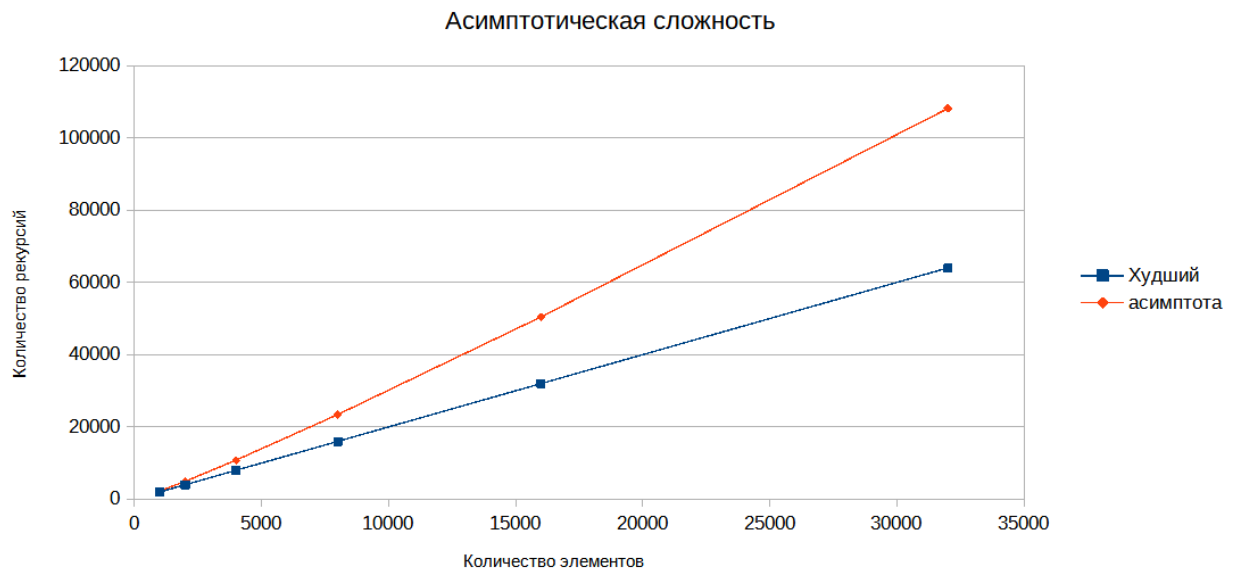
Из полученных результатов можно сказать, что количество рекурсий в зависимости от длины вектора N можно получить по формуле $2*(N - 1)$. Тем самым можно сделать таблицу для более длинных векторов.

| Количество рекурсий | | | | | | |
|---------------------|------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 1998 | 3998 | 7998 | 15998 | 31998 | 63998 |
| Худший | 1998 | 3998 | 7998 | 15998 | 31998 | 63998 |
| Лучший | 1998 | 3998 | 7998 | 15998 | 31998 | 63998 |
| Средний | 1998 | 3998 | 7998 | 15998 | 31998 | 63998 |
| 0,75 | 2250 | 4951,544993 | 10806,17997 | 23418,53992 | 50449,43979 | 108123,5995 |

- График худшего, среднего и лучшего случаев для количества рекурсий в зависимости от длины вектора. Очевидно, что они будут полностью совпадать.



- Был построен график асимптотической сложности с помощью формулы $c*O(N*LOG(N))$. Для данного худшего случая $c = 0,75$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .



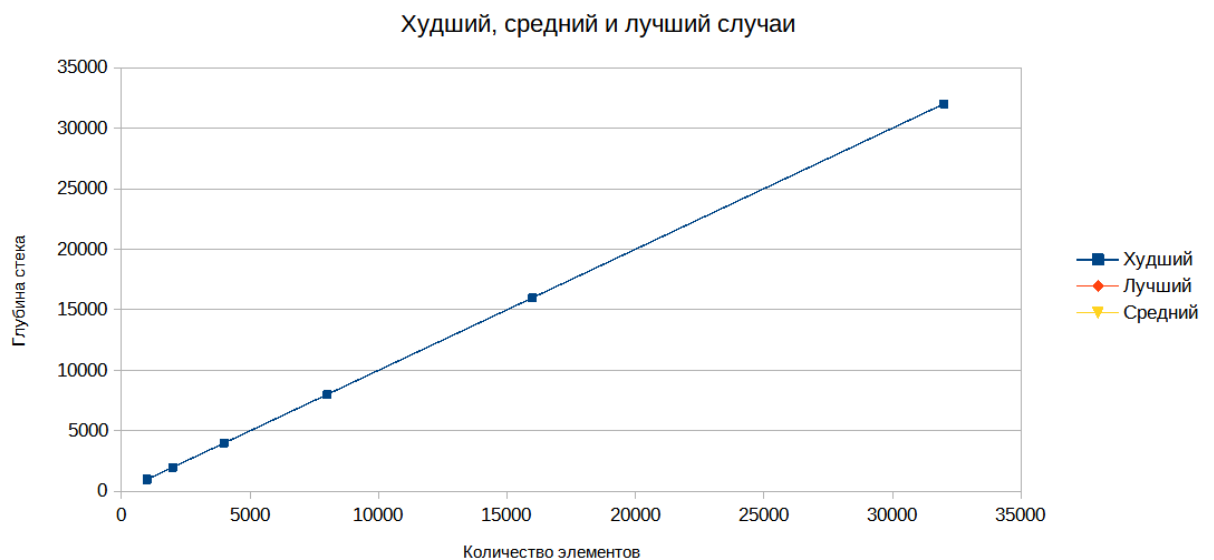
3. Глубина стека в зависимости от длины вектора.

- Полученные результаты

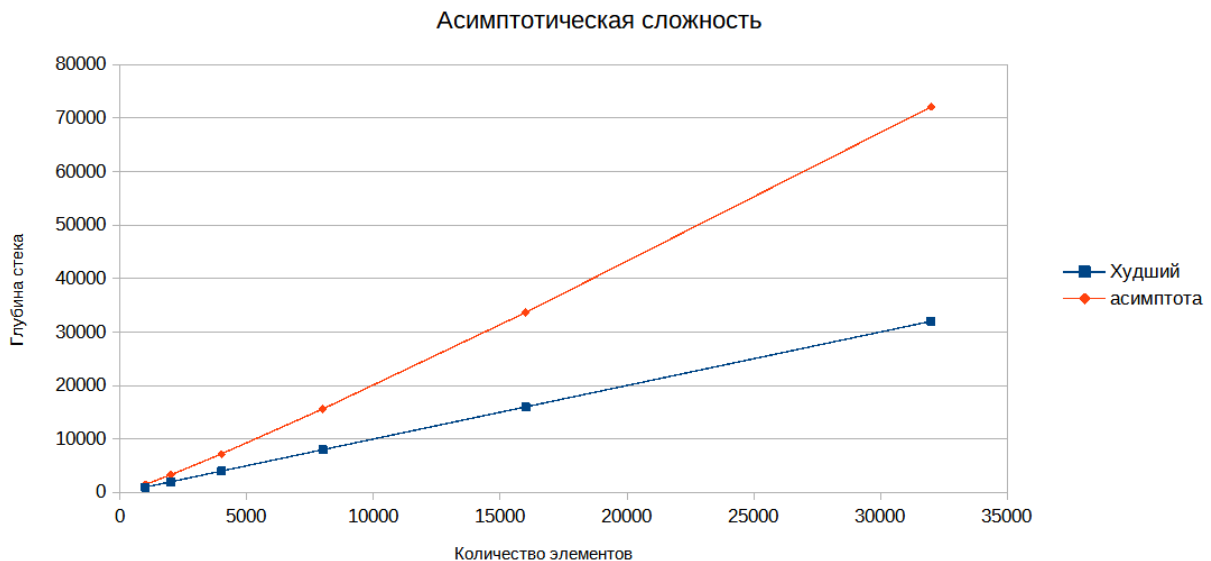
Из полученных результатов можно сказать, что глубину стека в зависимости от длины вектора N можно получить по формуле $N - 1$. Тем самым можно сделать таблицу для более длинных векторов.

| | Глубина стека | | | | | |
|---------|---------------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 999 | 1999 | 3999 | 7999 | 15999 | 31999 |
| Худший | 999 | 1999 | 3999 | 7999 | 15999 | 31999 |
| Лучший | 999 | 1999 | 3999 | 7999 | 15999 | 31999 |
| Средний | 999 | 1999 | 3999 | 7999 | 15999 | 31999 |
| 0,5 | 1500 | 3301,029996 | 7204,119983 | 15612,35995 | 33632,95986 | 72082,39965 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора. Очевидно, что они будут полностью совпадать.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,5$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .



Вектор с максимальным количеством сравнений при выборе среднего элемента в качестве опорного

1. Для получения такого вектора было необходимо выполнить алгоритм, который преобразует наш ранее отсортированный вектор таким образом, чтобы при быстрой сортировке опорный элемент выбирался так, чтобы вектор делился на части по 1 и $N-1$ элементу.
2. Путем итерации элементы меняются местами так, чтобы на каждом шаге в качестве среднего будет выбираться самый крупный элемент.

```
void antiQSort(vector<double> v, int N, int i) {
    for (int j = 2; j <= v.size(); j++) {
        int added_index = N - j;
        int middle = (added_index + (N - 1)) / 2;
        swap(&v[added_index], &v[middle]);
    }
}
```

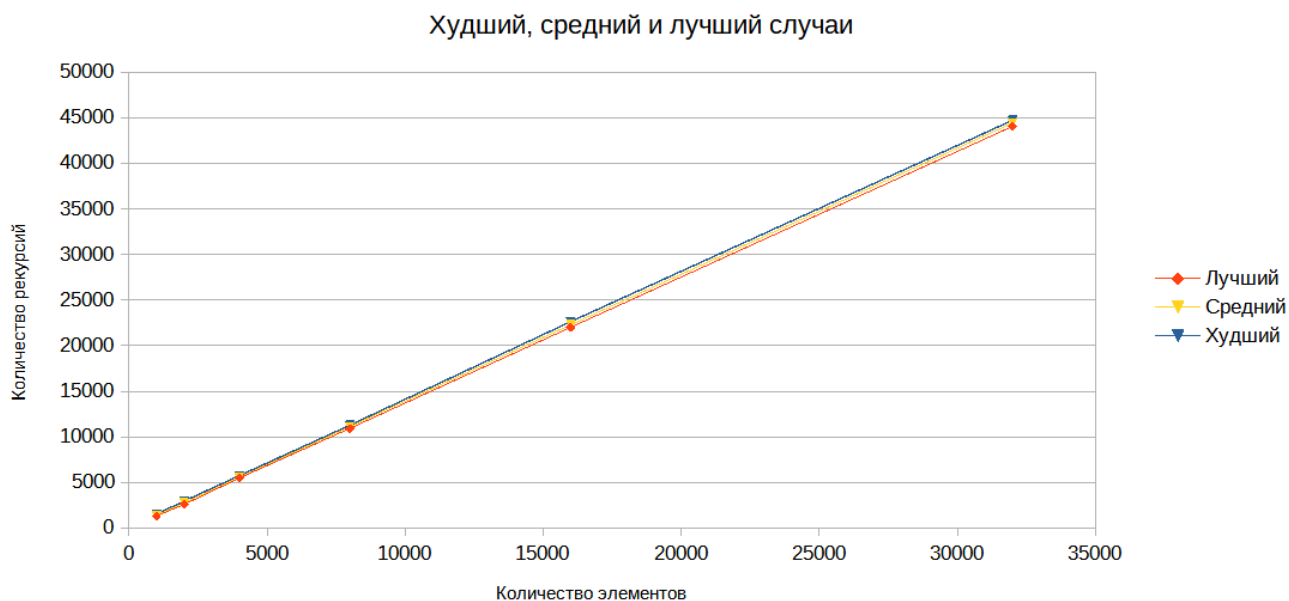
3. Далее также было проведено 20 тестов для различных размеров вектора (а именно, 1000, 2000, 4000, 8000, 16000, 32000).

4. Количество рекурсий в зависимости от длины вектора.

- Полученные результаты

| Количество рекурсий | | | | | | |
|---------------------|--------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 1410 | 2722 | 5564 | 11118 | 22298 | 44318 |
| 2 | 1378 | 2720 | 5522 | 11138 | 22266 | 44040 |
| 3 | 1326 | 2812 | 5564 | 11084 | 22422 | 44394 |
| 4 | 1416 | 2588 | 5732 | 11128 | 22386 | 44554 |
| 5 | 1412 | 2800 | 5486 | 11270 | 22228 | 44702 |
| 6 | 1470 | 2758 | 5472 | 11178 | 22006 | 44304 |
| 7 | 1430 | 2774 | 5494 | 10924 | 22198 | 44476 |
| 8 | 1402 | 2754 | 5540 | 11182 | 22476 | 44192 |
| 9 | 1394 | 2856 | 5554 | 11200 | 22364 | 44476 |
| 10 | 1426 | 2730 | 5526 | 11232 | 22188 | 44406 |
| 11 | 1404 | 2742 | 5524 | 10944 | 22318 | 44400 |
| 12 | 1418 | 2950 | 5554 | 11056 | 22628 | 44452 |
| 13 | 1394 | 2746 | 5528 | 11112 | 22162 | 44236 |
| 14 | 1392 | 2758 | 5516 | 11062 | 22138 | 44498 |
| 15 | 1396 | 2734 | 5534 | 11076 | 22392 | 44666 |
| 16 | 1384 | 2744 | 5512 | 11140 | 22560 | 44434 |
| 17 | 1366 | 2788 | 5640 | 10986 | 22162 | 44190 |
| 18 | 1320 | 2732 | 5548 | 11138 | 22076 | 44468 |
| 19 | 1372 | 2892 | 5738 | 11084 | 22202 | 44480 |
| 20 | 1508 | 2788 | 5610 | 11160 | 22380 | 44596 |
| Худший | 1508 | 2950 | 5738 | 11270 | 22628 | 44702 |
| Лучший | 1320 | 2588 | 5472 | 10924 | 22006 | 44040 |
| Средний | 1400,9 | 2769,4 | 5557,9 | 11110,6 | 22292,5 | 44414,1 |
| 0,75 | 2250 | 4951,544993 | 10806,17997 | 23418,53992 | 50449,43979 | 108123,5995 |

- График худшего, среднего и лучшего случаев для количества рекурсий в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,75$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .

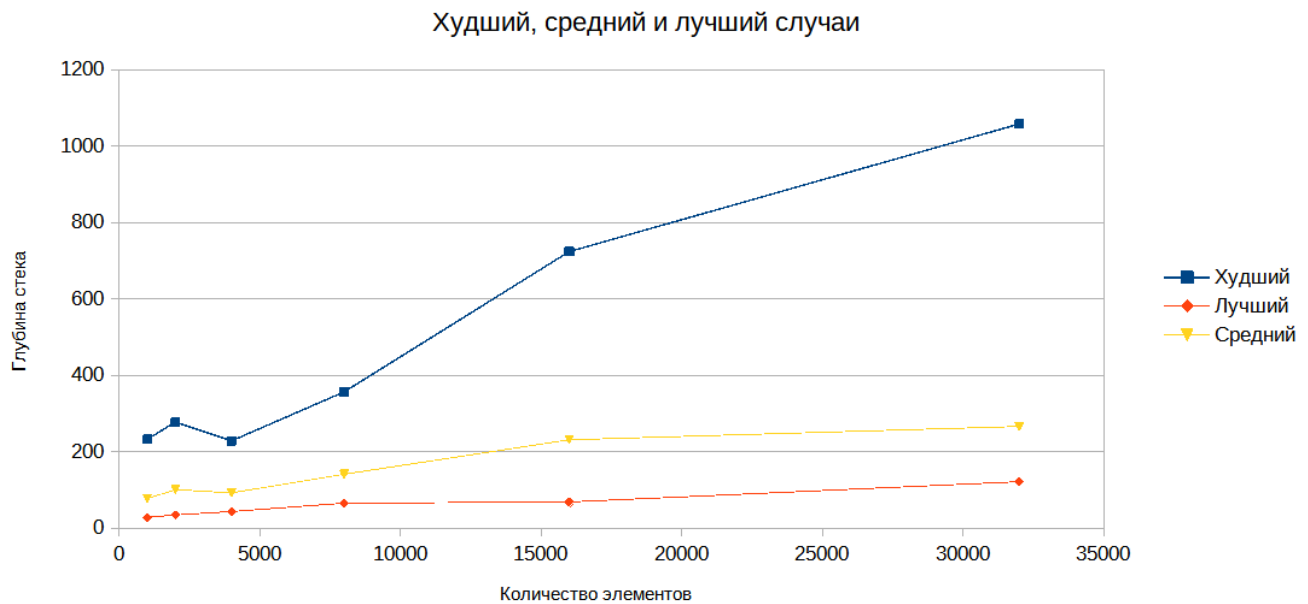


5. Глубина стека в зависимости от длины вектора.

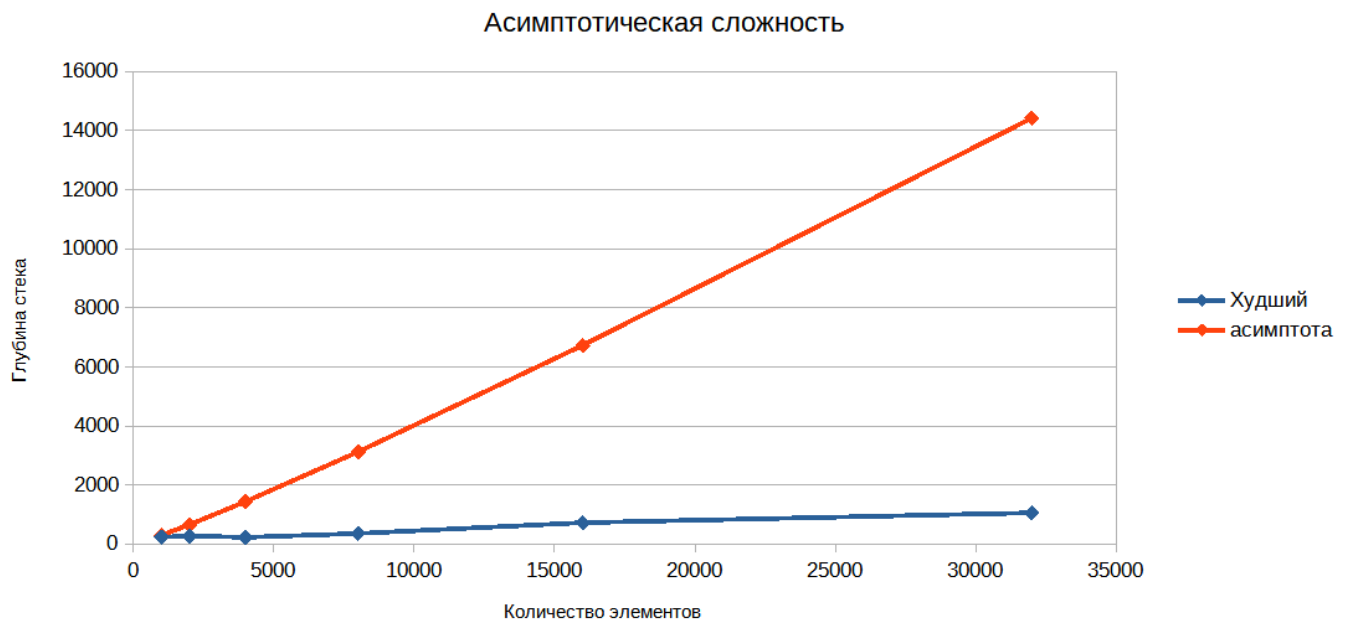
- Полученные результаты

| Глубина стека | | | | | | |
|---------------|------|-------------|-------------|------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 171 | 42 | 97 | 141 | 110 | 215 |
| 2 | 37 | 153 | 66 | 147 | 189 | 1058 |
| 3 | 226 | 53 | 53 | 74 | 351 | 220 |
| 4 | 43 | 278 | 229 | 96 | 271 | 217 |
| 5 | 60 | 61 | 60 | 145 | 302 | 250 |
| 6 | 60 | 119 | 61 | 215 | 109 | 201 |
| 7 | 57 | 91 | 65 | 157 | 131 | 140 |
| 8 | 45 | 54 | 100 | 116 | 275 | 167 |
| 9 | 29 | 154 | 135 | 102 | 184 | 143 |
| 10 | 108 | 64 | 52 | 357 | 69 | 150 |
| 11 | 64 | 65 | 67 | 101 | 125 | 326 |
| 12 | 35 | 145 | 104 | 168 | 724 | 325 |
| 13 | 53 | 46 | 83 | 103 | 130 | 198 |
| 14 | 43 | 61 | 133 | 236 | 94 | 218 |
| 15 | 38 | 65 | 57 | 80 | 116 | 451 |
| 16 | 45 | 260 | 95 | 122 | 469 | 202 |
| 17 | 33 | 94 | 63 | 256 | 383 | 122 |
| 18 | 233 | 35 | 44 | 71 | 263 | 428 |
| 19 | 40 | 125 | 201 | 66 | 140 | 147 |
| 20 | 132 | 61 | 90 | 77 | 198 | 153 |
| Худший | 233 | 278 | 229 | 357 | 724 | 1058 |
| Лучший | 29 | 35 | 44 | 66 | 69 | 122 |
| Средний | 77,6 | 101,3 | 92,75 | 141,5 | 231,65 | 266,55 |
| 0,1 | 300 | 660,2059991 | 1440,823997 | 3122,47199 | 6726,591972 | 14416,47993 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,1$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .

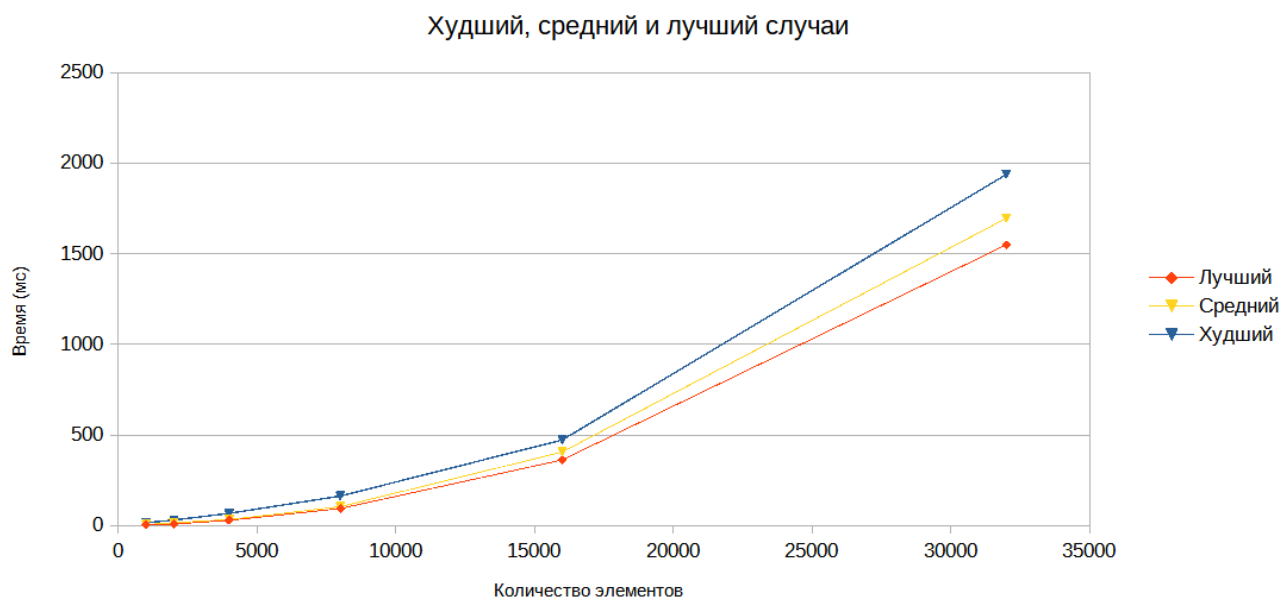


6. Время сортировки в зависимости от длины вектора.

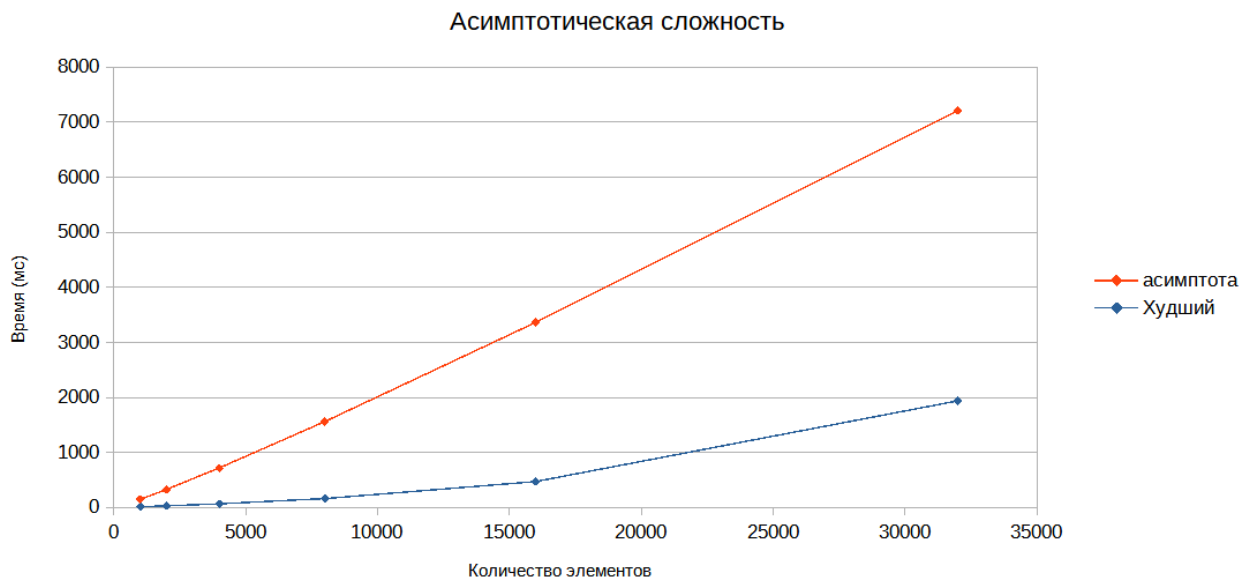
- Полученные результаты

| | Время | | | | | |
|---------|----------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 4,2084 | 9,3323 | 67,56 | 104,058 | 380,808 | 1549,2 |
| 2 | 3,841 | 10,1458 | 30,4603 | 106,024 | 390,411 | 1636,46 |
| 3 | 5,1935 | 13,1525 | 30,826 | 105,325 | 400,269 | 1816,49 |
| 4 | 4,1491 | 30,8071 | 42,0147 | 100,789 | 387,315 | 1662,69 |
| 5 | 4,1825 | 9,2744 | 31,087 | 103,004 | 378,495 | 1706,55 |
| 6 | 4,5475 | 14,2114 | 29,6373 | 110,664 | 362,55 | 1643,52 |
| 7 | 5,5374 | 9,3185 | 32,2431 | 105,063 | 364,234 | 1688,78 |
| 8 | 3,4026 | 20,5677 | 29,0761 | 104,903 | 386,265 | 1833,13 |
| 9 | 5,8778 | 18,9336 | 33,0273 | 99,5366 | 379,782 | 1937,63 |
| 10 | 5,2838 | 18,5452 | 28,8525 | 163,003 | 365,036 | 1792,71 |
| 11 | 5,6424 | 17,0572 | 30,6601 | 104,548 | 418,221 | 1615,09 |
| 12 | 5,0878 | 26,4983 | 30,5445 | 101,773 | 462,76 | 1683,15 |
| 13 | 5,5078 | 19,6149 | 30,0684 | 99,365 | 419,805 | 1649,75 |
| 14 | 3,9196 | 17,1577 | 30,413 | 101,621 | 408,691 | 1702,65 |
| 15 | 5,379 | 18,5875 | 31,0431 | 94,4824 | 472,841 | 1647,33 |
| 16 | 16,3858 | 26,7258 | 31,6636 | 100,004 | 453,704 | 1701,94 |
| 17 | 6,6653 | 14,5983 | 33,4141 | 99,0811 | 439,618 | 1633,39 |
| 18 | 10,067 | 9,6914 | 31,0072 | 96,3406 | 418,3 | 1679,37 |
| 19 | 4,8264 | 12,9642 | 36,0277 | 94,9532 | 412,839 | 1712,23 |
| 20 | 5,7222 | 12,5348 | 29,5391 | 94,2229 | 422,557 | 1625,48 |
| Худший | 16,3858 | 30,8071 | 67,56 | 163,003 | 472,841 | 1937,63 |
| Лучший | 3,4026 | 9,2744 | 28,8525 | 94,2229 | 362,55 | 1549,2 |
| Средний | 5,771345 | 16,48593 | 33,458255 | 104,43804 | 406,22505 | 1695,877 |
| 0,05 | 150 | 330,1029996 | 720,4119983 | 1561,235995 | 3363,295986 | 7208,239965 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,05$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .



Вектор с максимальным количеством сравнений при детерминированном выборе опорного элемента

1. Для получения такого вектора было необходимо выполнить алгоритм, который преобразует наш ранее отсортированный вектор таким образом, чтобы при быстрой сортировке опорный элемент выбирался так, чтобы вектор делился на части по 1 и $N-1$ элементу.
2. Путем итерации элементы меняются местами так, чтобы на каждом шаге в качестве среднего будет выбираться самый крупный элемент.

```
void determQSort(vector<double> v, int N, int i) {
    double temp_array;
    int k = 0, n = N - 1;

    for (int j = 0; j < N; j++) {
        temp_array = v[n - j];
        v[n - j] = v[(n - j) / 2];
        v[(n - j) / 2] = temp_array;
    }
    timer(v, N, i);
}
```

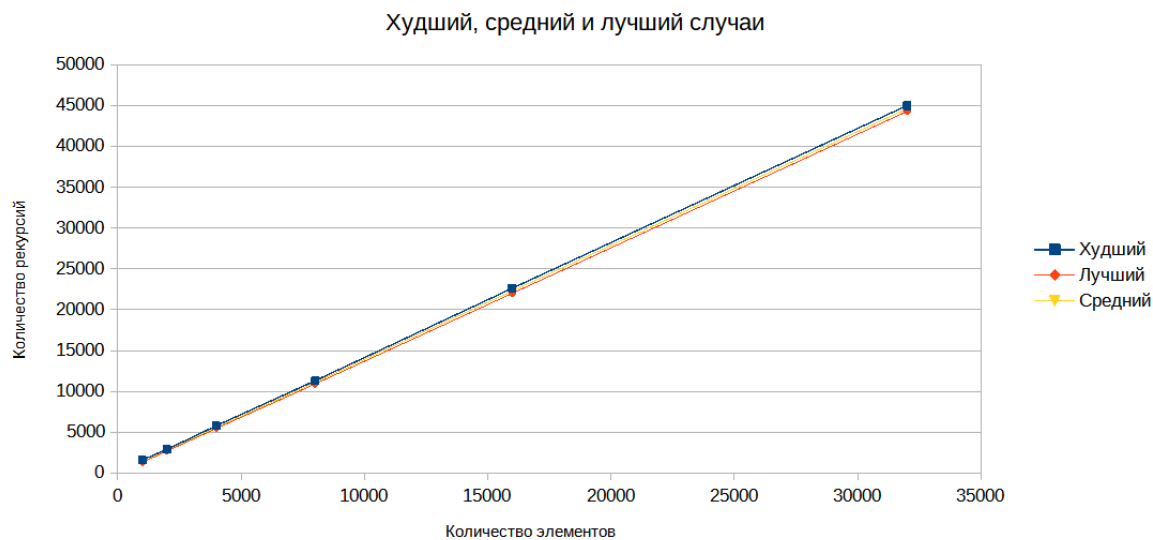
3. Далее также было проведено 20 тестов для различных размеров вектора (а именно, 1000, 2000, 4000, 8000, 16000, 32000).

4. Количество рекурсий в зависимости от длины вектора.

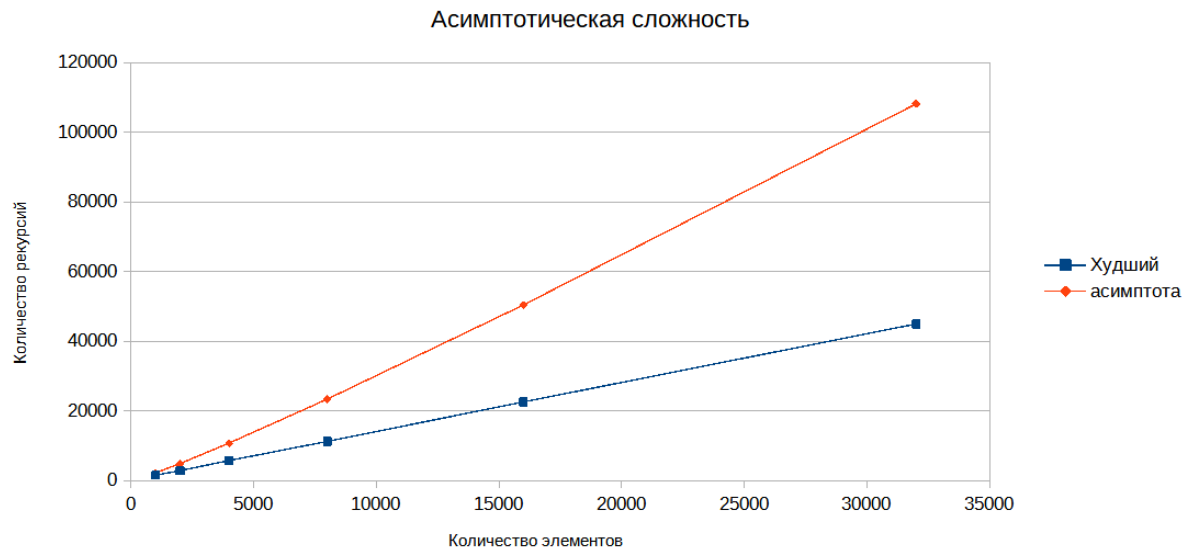
- Полученные результаты

| Количество рекурсий | | | | | | |
|---------------------|--------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 1364 | 2906 | 5648 | 10892 | 22616 | 44464 |
| 2 | 1346 | 2820 | 5548 | 10978 | 22014 | 44276 |
| 3 | 1402 | 2780 | 5522 | 11108 | 22060 | 44474 |
| 4 | 1438 | 2856 | 5512 | 10932 | 22510 | 44462 |
| 5 | 1372 | 2700 | 5526 | 11234 | 22116 | 44778 |
| 6 | 1428 | 2752 | 5454 | 11024 | 22314 | 44520 |
| 7 | 1396 | 2828 | 5608 | 11134 | 22532 | 44260 |
| 8 | 1388 | 2778 | 5490 | 11294 | 22534 | 44808 |
| 9 | 1366 | 2750 | 5548 | 10990 | 22348 | 44406 |
| 10 | 1310 | 2732 | 5560 | 11164 | 21996 | 44632 |
| 11 | 1384 | 2752 | 5550 | 11046 | 22578 | 44672 |
| 12 | 1400 | 2798 | 5486 | 11154 | 22002 | 44462 |
| 13 | 1462 | 2686 | 5608 | 11054 | 22240 | 44760 |
| 14 | 1372 | 2818 | 5540 | 11074 | 22056 | 44600 |
| 15 | 1350 | 2850 | 5534 | 11008 | 22328 | 44492 |
| 16 | 1416 | 2690 | 5638 | 11004 | 22260 | 44526 |
| 17 | 1342 | 2762 | 5600 | 11208 | 22572 | 44974 |
| 18 | 1378 | 2818 | 5592 | 11170 | 22032 | 44718 |
| 19 | 1402 | 2752 | 5502 | 11148 | 22150 | 44514 |
| 20 | 1600 | 2898 | 5790 | 11114 | 22402 | 44822 |
| Худший | 1600 | 2906 | 5790 | 11294 | 22616 | 44974 |
| Лучший | 1310 | 2686 | 5454 | 10892 | 21996 | 44260 |
| Средний | 1395,8 | 2786,3 | 5562,8 | 11086,5 | 22283 | 44581 |
| 0,75 | 2250 | 4951,544993 | 10806,17997 | 23418,53992 | 50449,43979 | 108123,5995 |

- График худшего, среднего и лучшего случаев для количества рекурсий в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,75$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .

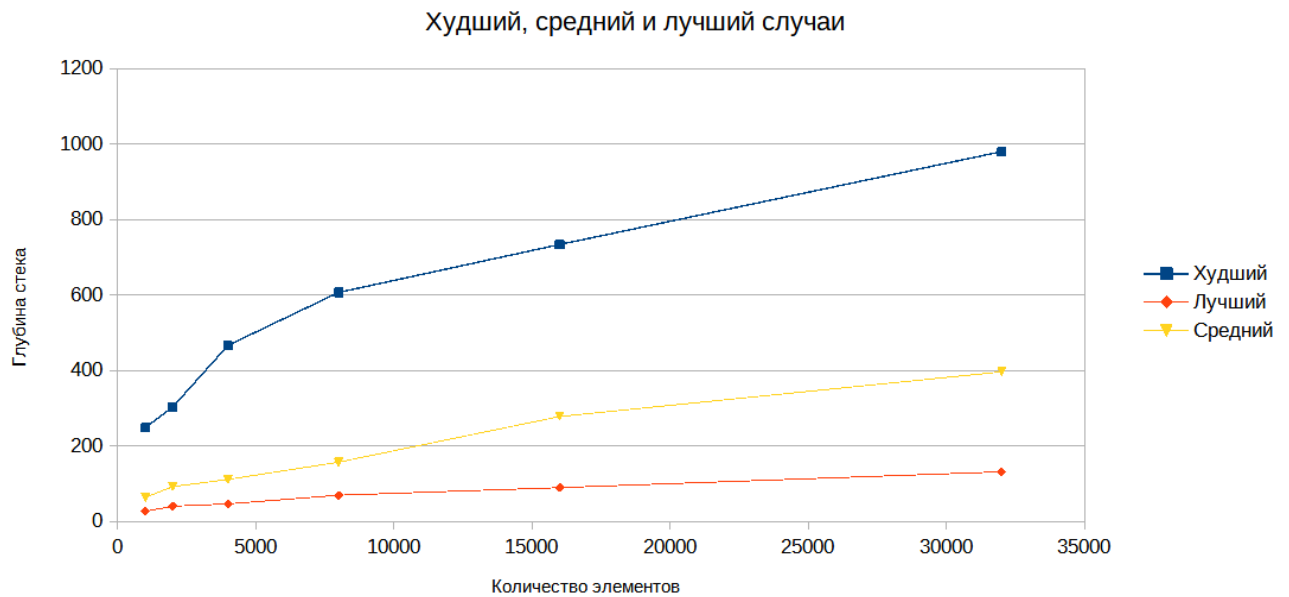


5. Глубина стека в зависимости от длины вектора.

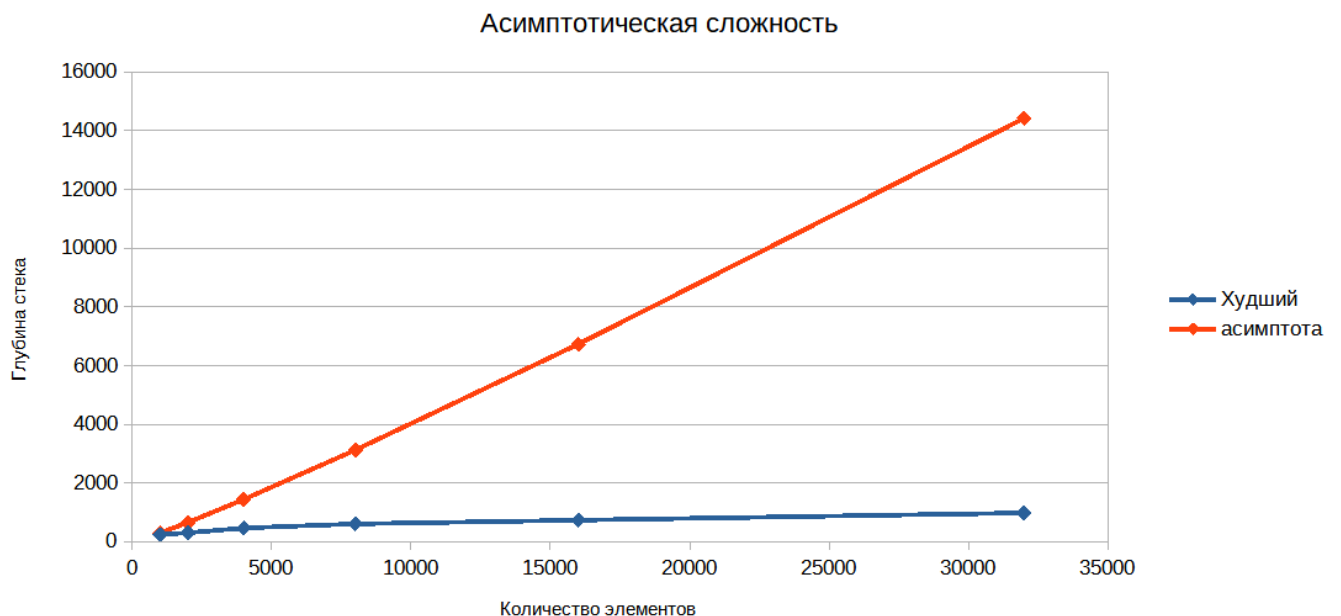
- Полученные результаты

| | Глубина стека | | | | | |
|---------|---------------|-------------|-------------|------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 63 | 88 | 90 | 97 | 734 | 221 |
| 2 | 84 | 46 | 54 | 106 | 261 | 216 |
| 3 | 27 | 58 | 63 | 109 | 361 | 292 |
| 4 | 65 | 141 | 47 | 98 | 219 | 134 |
| 5 | 70 | 56 | 101 | 175 | 159 | 704 |
| 6 | 71 | 75 | 79 | 91 | 368 | 504 |
| 7 | 31 | 83 | 93 | 607 | 634 | 250 |
| 8 | 83 | 65 | 125 | 163 | 128 | 920 |
| 9 | 49 | 50 | 75 | 110 | 144 | 281 |
| 10 | 37 | 113 | 72 | 142 | 269 | 353 |
| 11 | 30 | 75 | 89 | 99 | 187 | 377 |
| 12 | 46 | 96 | 111 | 130 | 90 | 386 |
| 13 | 136 | 41 | 98 | 108 | 226 | 726 |
| 14 | 57 | 51 | 110 | 218 | 154 | 160 |
| 15 | 37 | 106 | 198 | 82 | 200 | 228 |
| 16 | 30 | 83 | 80 | 276 | 181 | 305 |
| 17 | 35 | 304 | 102 | 196 | 468 | 980 |
| 18 | 33 | 49 | 82 | 154 | 192 | 432 |
| 19 | 51 | 62 | 100 | 118 | 144 | 132 |
| 20 | 249 | 212 | 467 | 70 | 448 | 333 |
| Худший | 249 | 304 | 467 | 607 | 734 | 980 |
| Лучший | 27 | 41 | 47 | 70 | 90 | 132 |
| Средний | 64,2 | 92,7 | 111,8 | 157,45 | 278,35 | 396,7 |
| 0,1 | 300 | 660,2059991 | 1440,823997 | 3122,47199 | 6726,591972 | 14416,47993 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,1$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .

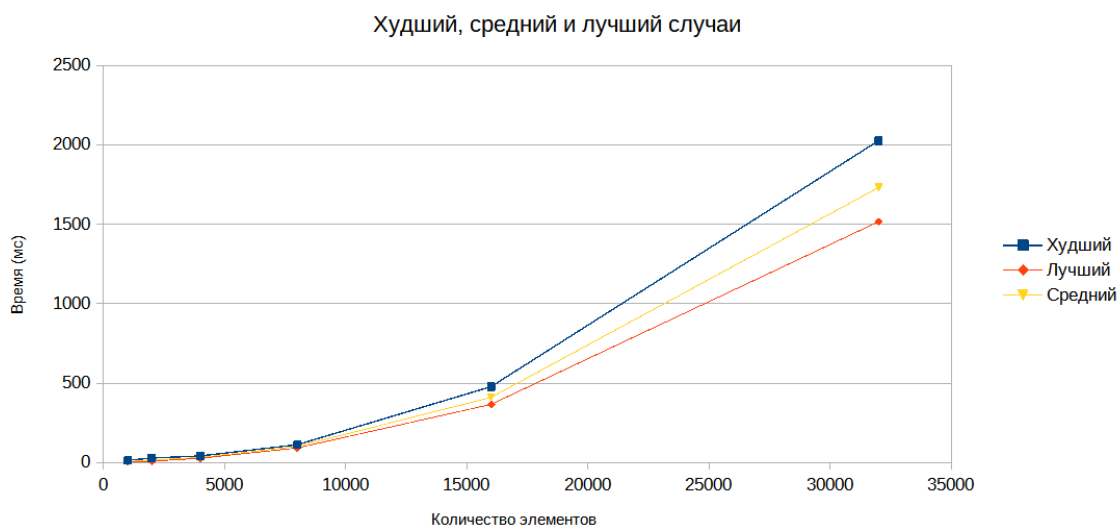


6. Время сортировки в зависимости от длины вектора.

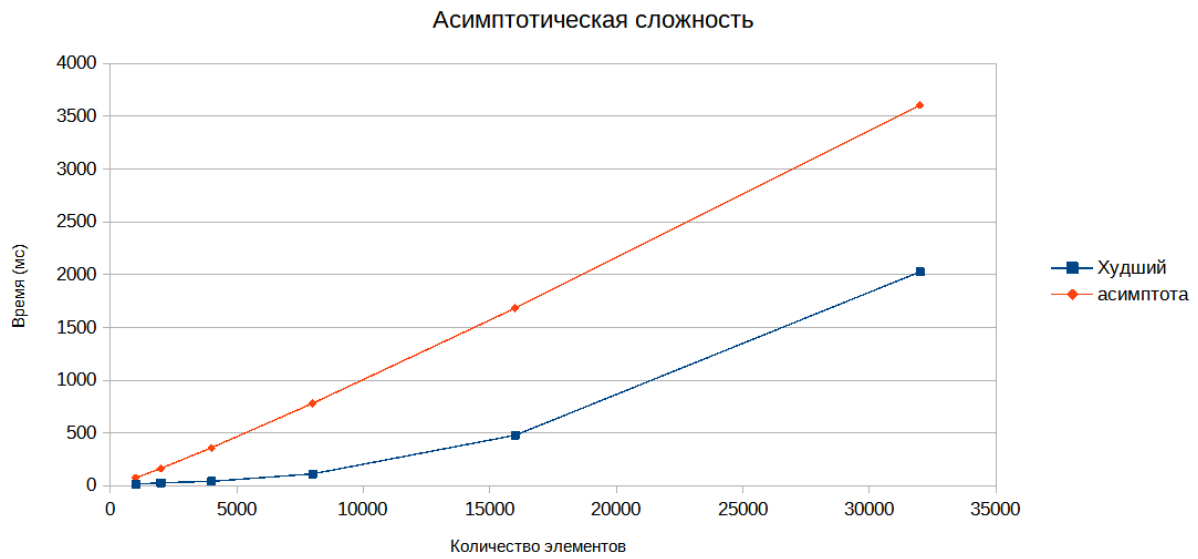
- Полученные результаты

| | Время | | | | | |
|---------|---------|-------------|-------------|-------------|-------------|-------------|
| | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
| 1 | 4,4734 | 10,4381 | 38,3184 | 100,887 | 453,923 | 1599,06 |
| 2 | 3,5789 | 8,9996 | 32,4471 | 92,0659 | 376,317 | 1516,28 |
| 3 | 3,7615 | 8,8357 | 29,8364 | 95,6873 | 374,96 | 1699,84 |
| 4 | 4,1619 | 12,4686 | 30,7138 | 99,5399 | 392,588 | 1680,76 |
| 5 | 3,6994 | 14,3954 | 32,5722 | 102,246 | 366,311 | 1779,02 |
| 6 | 4,3736 | 11,8703 | 29,0048 | 98,0536 | 395,566 | 1728,11 |
| 7 | 6,3466 | 14,8217 | 31,2954 | 101,774 | 422,863 | 1666 |
| 8 | 5,2057 | 26,7831 | 29,0829 | 106,162 | 371,877 | 2025,67 |
| 9 | 4,5224 | 17,7766 | 27,6634 | 100,221 | 376,068 | 1967,04 |
| 10 | 7,8745 | 14,2623 | 28,251 | 103,463 | 378,007 | 1815,7 |
| 11 | 5,2877 | 22,3993 | 28,7676 | 99,3126 | 427,317 | 1781,8 |
| 12 | 5,5881 | 14,9443 | 29,5216 | 99,1558 | 405,103 | 1670,37 |
| 13 | 4,4215 | 13,2099 | 29,3952 | 103,904 | 432,399 | 1763,21 |
| 14 | 4,7896 | 21,1111 | 31,8411 | 110,197 | 414,817 | 1680,53 |
| 15 | 4,9924 | 19,5981 | 32,8708 | 97,8015 | 435,896 | 1712,98 |
| 16 | 6,7916 | 17,3959 | 42,0284 | 105,729 | 432,678 | 1696,34 |
| 17 | 9,2106 | 28,7277 | 37,7758 | 112,606 | 455,803 | 1781,97 |
| 18 | 7,1458 | 14,5774 | 34,4979 | 111,788 | 410,197 | 1659,24 |
| 19 | 6,6986 | 11,5466 | 38,1261 | 103,523 | 407,987 | 1634,87 |
| 20 | 14,4406 | 25,1467 | 32,5658 | 101,717 | 478,019 | 1758,31 |
| Худший | 14,4406 | 28,7277 | 42,0284 | 112,606 | 478,019 | 2025,67 |
| Лучший | 3,5789 | 8,8357 | 27,6634 | 92,0659 | 366,311 | 1516,28 |
| Средний | 5,86822 | 16,46542 | 32,328785 | 102,29168 | 410,4348 | 1730,855 |
| 0,025 | 75 | 165,0514998 | 360,2059991 | 780,6179974 | 1681,647993 | 3604,119983 |

- График худшего, среднего и лучшего случаев для глубины стека в зависимости от длины вектора.



- Был построен график асимптотической сложности с помощью формулы $c \cdot O(N \cdot \log(N))$. Для данного худшего случая $c = 0,025$ для того, чтобы асимптота была выше графика худшего случая при увеличении длины вектора N .



Заключение.

В заключении можно заметить, что метод быстрой сортировки действительно занимает меньше времени на сортировку массива. Также алгоритм самой сортировке достаточно просто в реализации. Однако скорость зависит от того, какой массив был сгенерирован изначально. В частных случаях опорный элемент может выбираться не эффективно, поэтому массив будет разделен на части из 1 и $N-1$ элементов.

Помимо этого, проблема данной сортировки в том, что из-за того, что функция сортировки вызывает сама себя, тем самым происходит заполнение стека рекурсий. При достаточно большом векторе может произойти такое, что стек рекурсий будет переполнен, что приведет к остановке программы. Именно поэтому в лабораторной работе не были выполнены тесты на 64000 и 128000 элементов.

Также из недостатков можно выделить то, что данная сортировка сильно деградирует по скорости до $O(N^2)$. На это сильно влияет сгенерированный изначально массив элементов.

Для того, чтобы избежать переполнение стека рекурсий стоит использовать другой метод сортировки, например, более медленных.