Компьютерная Безопасность
МИЕМ НИУ ВШЭ

Работа: Мельникова Льва СКБ 161
Вариант 19

Сравнение алгоритмов поиска

Были написаны следующие алгоритмы:

1. Прямой поиск
2. Бинарный поиск
3. Сортировка с бинарным поиском
4. Поиск в контейнере std::map

Была написана структура данных для хранения информации о квартире в доме. Для неё были сгенерированы наборы данных разных размеров. На этих данных были измерено время поисков алгоритмов поиска.

Получены результаты:

Бинарный поиск

Размер массива	Время поиска в мс
10	2
20	2
50	2
100	2
200	3
500	3
1000	3
2000	2
5000	1
10000	1
20000	1
50000	1
100000	3
200000	2
500000	3
1000000	4

Поиск в map

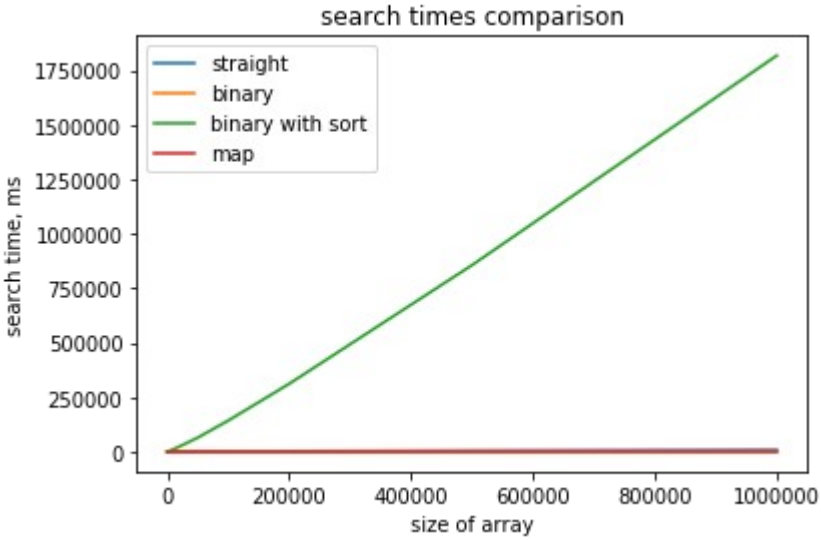
Размер массива	Время поиска в мс
10	0
20	0
50	0
100	0
200	0
500	0
1000	0
2000	0
5000	0
10000	0
20000	1
50000	2
100000	3
200000	3
500000	4
1000000	4

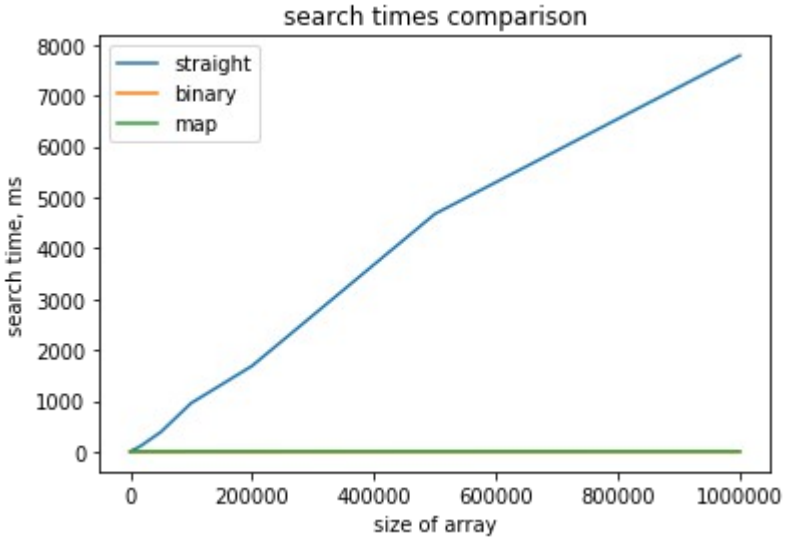
Сортировка и бинарный поиск

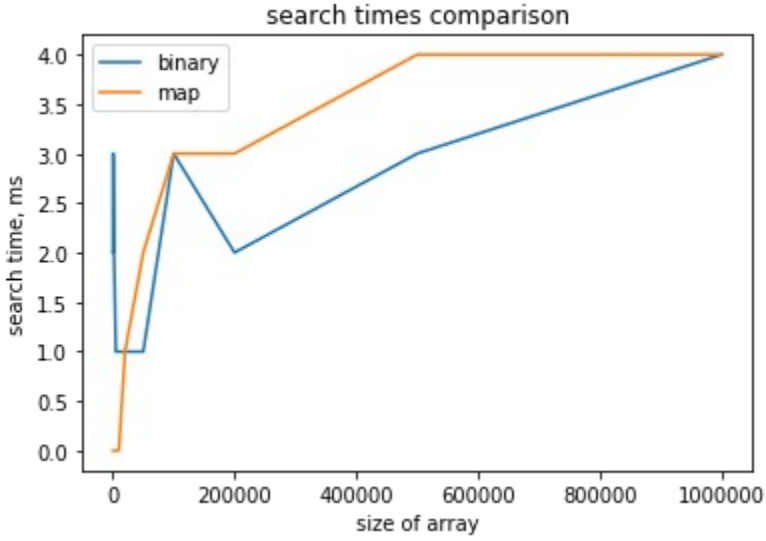
Размер массива	Время поиска в мс
10	4
20	9
50	29
100	63
200	
500	394
1000	857
2000	1815
5000	5026
10000	10867
20000	23487
50000	64372
100000	142519
200000	311255
500000	855917
1000000	1817452

Прямой поиск

Размер массива	Время поиска в мс
10	0
20	0
50	0
100	1
200	2
500	3
1000	7
2000	16
5000	41
10000	63
20000	135
50000	384
100000	954
200000	1686
500000	4672
1000000	7782


search times comparison
straight
binary
binary with sort
map
search time, ms
size of array
0
250000
500000
750000
1000000
1250000
1500000
1750000
0
200000
400000
600000
800000
1000000


search times comparison
straight
binary
map
search time, ms
size of array
0
1000
2000
3000
4000
5000
6000
7000
8000
0
200000
400000
600000
800000
1000000


search times comparison
binary
map
search time, ms
size of array
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
200000
400000
600000
800000
1000000

Выводы:

1. Самый плохой с точки зрения времени поиск – сортировка с бинарным поиском, прямой поиск лучше.
2. Если данные сортированы, то бинарный поиск – наилучший.
3. Самописный бинарный поиск не сильно отличается от поиска в контейнере std::map.

Код:

```
/*
 * Searches for element in array. Using binary search.
 *
 * @param begin - iterator to the first element
 * @param end   - iterator to the element after last
 * @param value - value to search for
 *
 * @return iterator to the found element or end iterator if it wasn't found.
 */
template<typename RandomIterator, typename T>
template<typename RandomIterator, typename T>
RandomIterator binarySearch (RandomIterator begin, RandomIterator end, const T& value) {
   RandomIterator notFound = end;
   std::size_t distance = end - begin;
   while (distance) {
      RandomIterator current = begin + (distance - 1) / 2;
      if (*current == value)
         return current;
      if (value < *current)
         end = current;
      else
         begin = current + 1;

      distance = end - begin;
   }
   return notFound;
}
```

```
/*
 * Searches for element in array. Using straight search.
 *
 * @param begin - iterator to the first element
 * @param end   - iterator to the element after last
 * @param value - value to search for
 *
 * @return iterator to the found element or end iterator if it wasn't found.
 */
template<typename RandomIterator, typename T>
RandomIterator straightSearch (RandomIterator begin, RandomIterator end, const T& value) {
   while (begin != end) {
      if (*begin == value)
         return begin;
      begin++;
   }
   return end;
}
```

Весь код проекта можно посмотреть здесь:
https://github.com/MarkGrin/sortringHomework