Search

Zhanfu Yang

Normal Search 32 [Hit]

```
int normal_search(int *array, int search_key, int beg, int end) {
        for (int i = beg; i < end; i ++) {
                if (array[i] == search_key) {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key, i);
                        }
                        // printf("Found %d at index = %d\n", search_key, i);
                        return i;
                }
        }
        return -1;
}
```

[0] [1] [2] [3] [4] [5] [6]

Normal Search 32 [Hit] 1

```c
int normal_search(int *array, int search_key, int beg, int end) {
        for (int i = beg; i < end; i ++) {
                if (array[i] == search_key) {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key, i);
                        }
                        // printf("Found %d at index = %d\n", search_key, i);
                        return i;
                }
        }
        return -1;
}
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Normal Search 32 [Hit] 2

```
int normal_search(int *array, int search_key, int beg, int end) {
        for (int i = beg; i < end; i ++) {
                if (array[i] == search_key) {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key, i);
                        }
                        // printf("Found %d at index = %d\n", search_key, i);
                        return i;
                }
        }
        return -1;
}
```

[0] [1] [2] [3] [4] [5] [6]

3	6	7	11	32	33	53

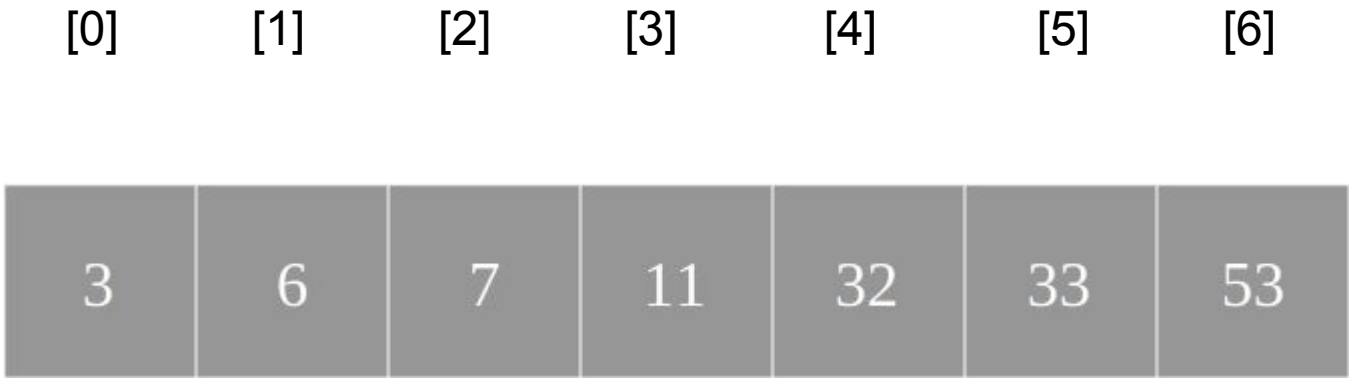
Normal Search 32 [Hit] 6

```c
int normal_search(int *array, int search_key, int beg, int end) {
        for (int i = beg; i < end; i ++) {
                if (array[i] == search_key) {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key, i);
                        }
                        // printf("Found %d at index = %d\n", search_key, i);
                        return i;
                }
        }
        return -1;
}
```

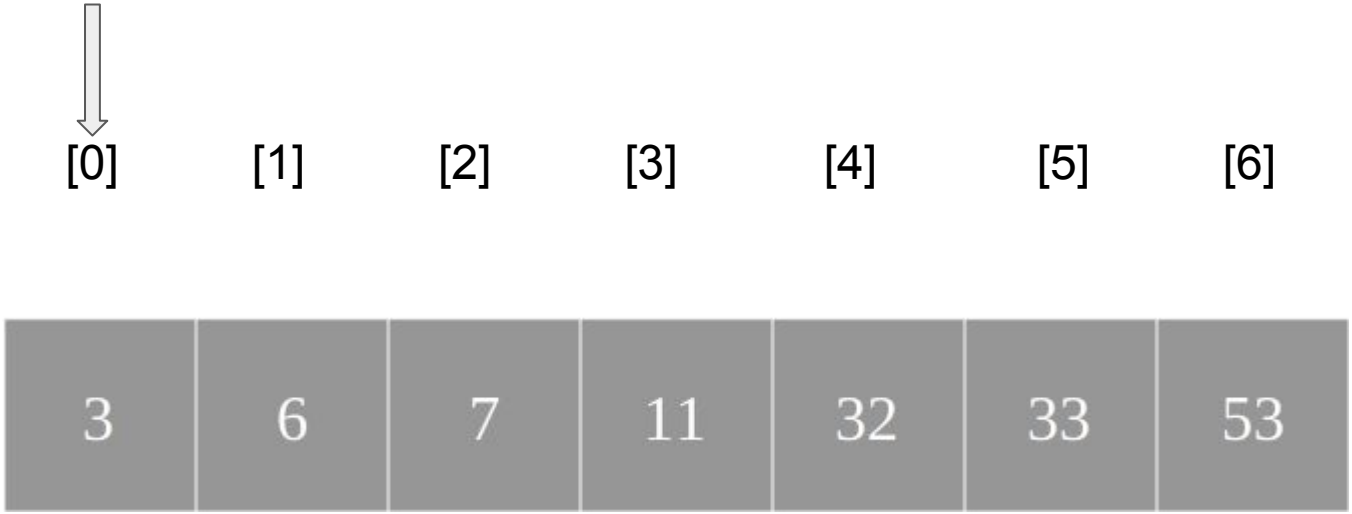
[0] [1] [2] [3] [4] [5] [6]

3	6	7	11	32	33	53

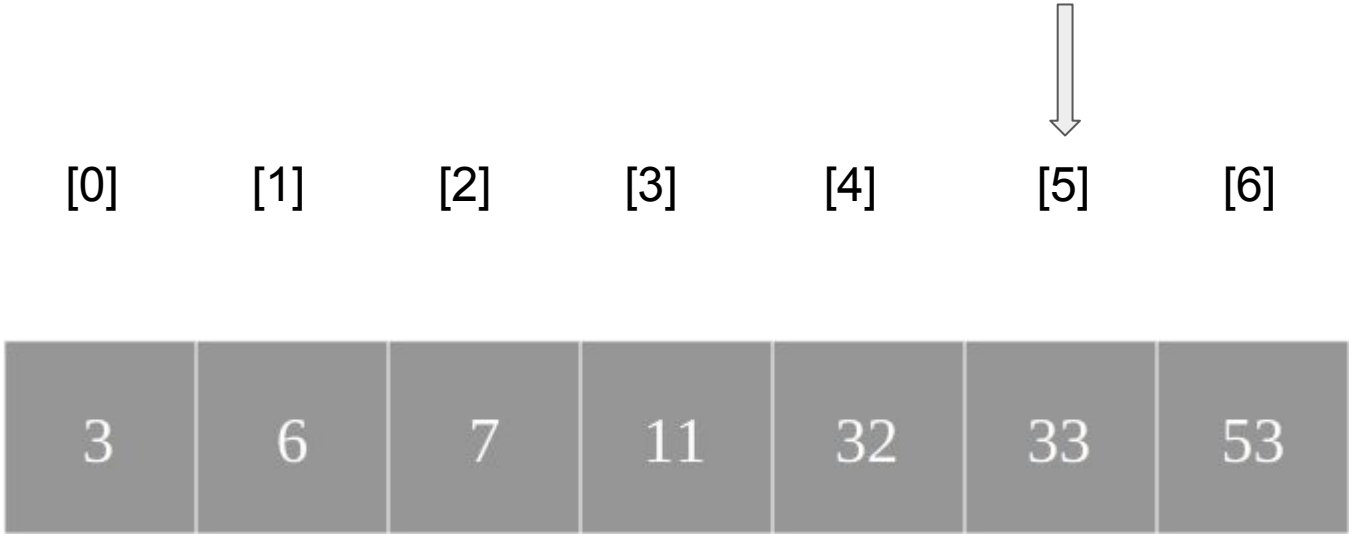
Normal Search 34 [Miss]



Normal Search 34 [Miss] 1

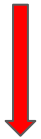


Normal Search 34 [Miss] 6



Normal Search 34 [Miss] 7 X

```
int normal_search(int *array, int search_key, int beg, int end) {
        for (int i = beg; i < end; i ++) {
                if (array[i] == search_key) {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key, i);
                        }
                        // printf("Found %d at index = %d\n", search_key, i);
                        return i;
                }
        }
        return -1;
}
```



[0] [1] [2] [3] [4] [5] [6]

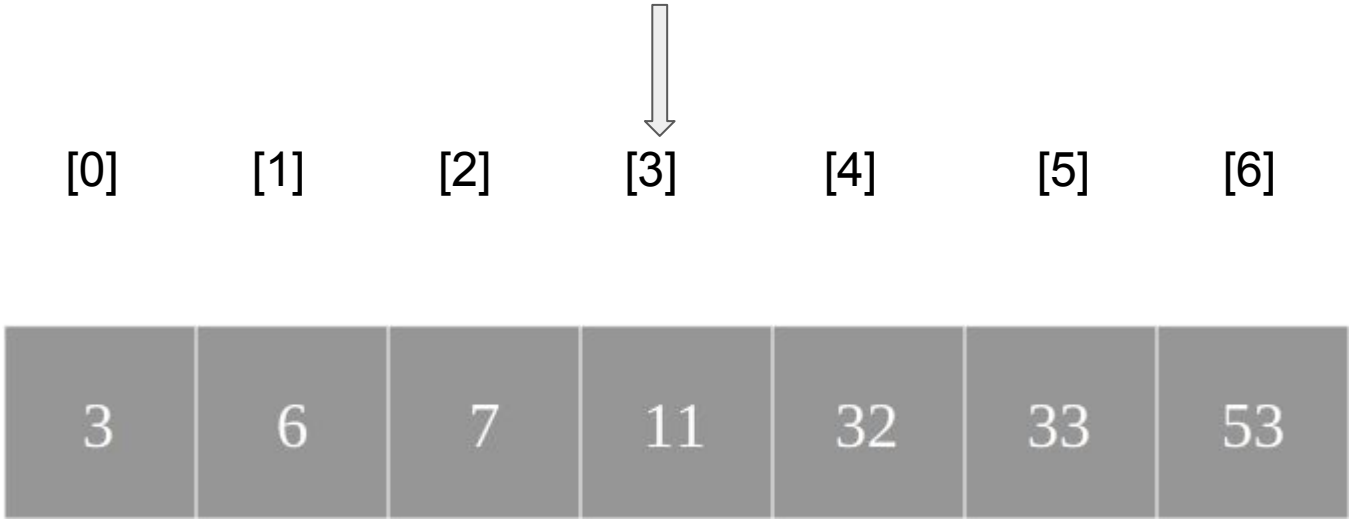
3	6	7	11	32	33	53

Binary Search 32 [Hit]

```
int binary_search(int *array, int search_key, int beg, int end)
{

        while(beg<=end)
        {
                int mid = (beg + end)/2;
                if(array[mid] == search_key)
                {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key,mid);
                        }
                        return 1;
                }else if(array[mid] < search_key){
                        beg = mid + 1;

                }else{
                        end = mid - 1;
                }
                //searching range changed to
                if (print_out) {
                        printf("Valid range: array[%d] = %d -- array[%d] = %d for search_key = %d.\n", beg,array[beg],end,array[end-1], search_key);
                }
        }

        printf("Didn't find %d in this array\n", search_key);
        return -1;
}
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Binary Search 32 [Hit] 1



Binary Search 32 [Hit] 2

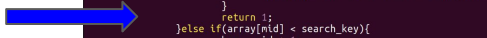
```c
int binary_search(int *array, int search_key, int beg, int end)
{

        while(beg<=end)
        {
                int mid = (beg + end)/2;
                if(array[mid] == search_key)
                {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key,mid);
                        }
                        return 1;
                }else if(array[mid] < search_key){
                        beg = mid + 1;

                }else{
                        end = mid - 1;
                }
                //searching range changed to
                if (print_out) {
                        printf("Valid range: array[%d] = %d -- array[%d] = %d for search_key = %d.\n", beg,array[beg],end,array[end-1], search_key);
                }
        }

        printf("Didn't find %d in this array\n", search_key);
        return -1;
}
```

[0] [1] [2] [3] [4] [5] [6]

3	6	7	11	32	33	53

Binary Search 32 [Hit] 2

```c
int binary_search(int *array, int search_key, int beg, int end)
{

        while(beg<=end)
        {
                int mid = (beg + end)/2;
                if(array[mid] == search_key)
                {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key,mid);
                        }
                        return 1;
                }else if(array[mid] < search_key){
                        beg = mid + 1;

                }else{
                        end = mid - 1;
                }
                //searching range changed to
                if (print_out) {
                        printf("Valid range: array[%d] = %d -- array[%d] = %d for search_key = %d.\n", beg,array[beg],end,array[end-1], search_key);
                }
        }

        printf("Didn't find %d in this array\n", search_key);
        return -1;
}
```



[0] [1] [2] [3] [4] [5] [6]

3	6	7	11	32	33	53

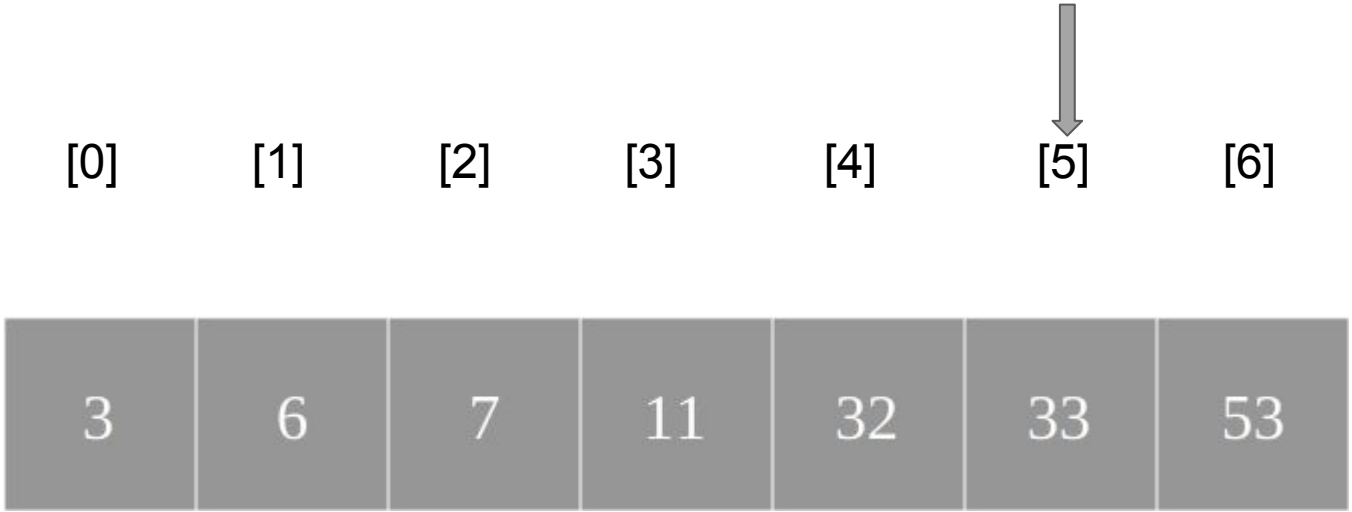
Binary Search 34 [Miss] 1

```c
int binary_search(int *array, int search_key, int beg, int end)
{

        while(beg<=end)
        {
                int mid = (beg + end)/2;
                if(array[mid] == search_key)
                {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key,mid);
                        }
                        return 1;
                }else if(array[mid] < search_key){
                        beg = mid + 1;

                }else{
                        end = mid - 1;
                }
                //searching range changed to
                if (print_out) {
                        printf("Valid range: array[%d] = %d -- array[%d] = %d for search_key = %d.\n", beg,array[beg],end,array[end-1], search_key);
                }
        }

        printf("Didn't find %d in this array\n", search_key);
        return -1;
}
```

[0] [1] [2] [3] [4] [5] [6]

3	6	7	11	32	33	53

Binary Search 34 [Miss] 3



Binary Search 34 [Miss] 4

```c
int binary_search(int *array, int search_key, int beg, int end)
{

        while(beg<=end)
        {
                int mid = (beg + end)/2;
                if(array[mid] == search_key)
                {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key,mid);
                        }
                        return 1;
                }else if(array[mid] < search_key){
                        beg = mid + 1;

                }else{
                        end = mid - 1;
                }
                //searching range changed to
                if (print_out) {
                        printf("Valid range: array[%d] = %d -- array[%d] = %d for search_key = %d.\n", beg,array[beg],end,array[end-1], search_key);
                }
        }

        printf("Didn't find %d in this array\n", search_key);
        return -1;
}
```

[0] [1] [2] [3] [4] [5] [6]

3	6	7	11	32	33	53

Normal Search Code

```
int normal_search(int *array, int search_key, int beg, int end) {
        for (int i = beg; i < end; i ++) {
                if (array[i] == search_key) {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key, i);
                        }
                        // printf("Found %d at index = %d\n", search_key, i);
                        return i;
                }
        }
        return -1;
}
```

Binary Search Code

```
int binary_search(int *array, int search_key, int beg, int end)
{

        while(beg<=end)
        {
                int mid = (beg + end)/2;
                if(array[mid] == search_key)
                {
                        if (print_out) {
                                printf("Found %d at index = %d\n",search_key,mid);
                        }
                        return 1;
                }else if(array[mid] < search_key){
                        beg = mid + 1;

                }else{
                        end = mid - 1;
                }
                //searching range changed to
                if (print_out) {
                        printf("Valid range: array[%d] = %d -- array[%d] = %d for search_key = %d.\n", beg,array[beg],end,array[end-1], search_key);
                }
        }

        printf("Didn't find %d in this array\n", search_key);
        return -1;
}
```

Time Measure Method

[1] Chrono in C++. https://cplusplus.com/reference/chrono/

```
auto start = std::chrono::high_resolution_clock::now();
// void qsort(void *base, size_t nmemb, size_t size,
//                   int (*compar)(const void *, const void *));
//sort the array
qsort(array, size, sizeof(int), my_compare_func);

auto end = std::chrono::high_resolution_clock::now();
```

[2] Print out:

```
auto duration = std::chrono::duration_cast<std::chrono::microseconds>(end - start);

auto duration2 = std::chrono::duration_cast<std::chrono::microseconds>(end2 - end);

auto duration3 = std::chrono::duration_cast<std::chrono::microseconds>(start - start0);

std::cout << "Time taken by normal search: " << duration3.count() << " microseconds" << std::endl;
std::cout << "Time taken by function quick sort: "  << duration.count() << " microseconds" << std::endl;
std::cout << "Time taken by function binary search: " << duration2.count() << " microseconds" << std::endl;
```

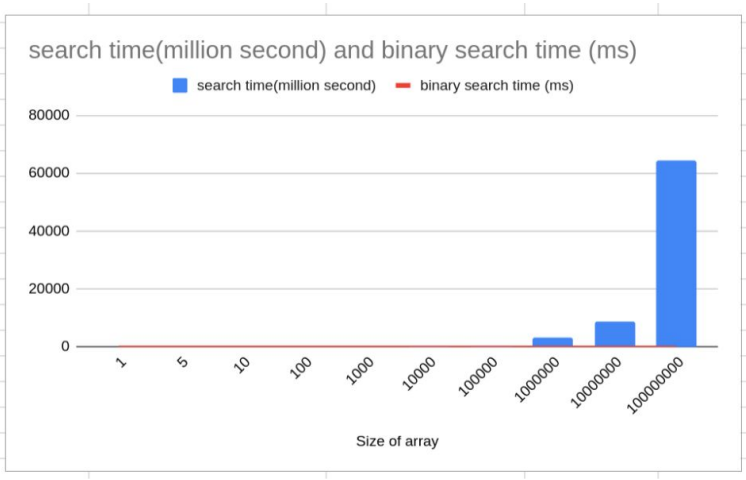
How to run ?

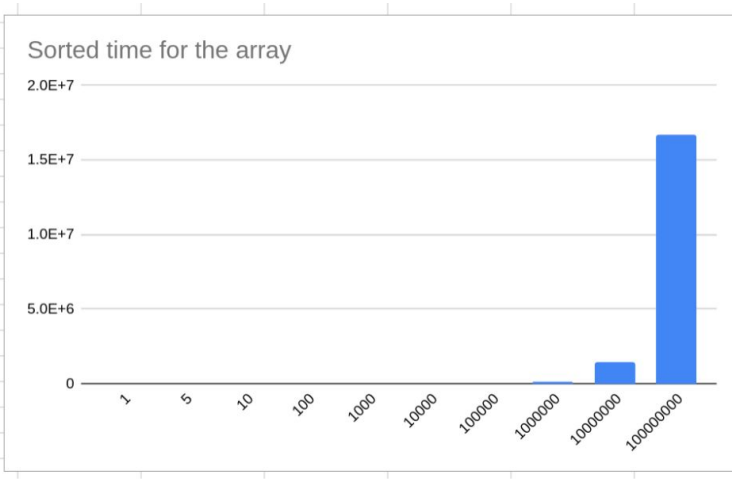
./a.out or in gdb-online:

Input:

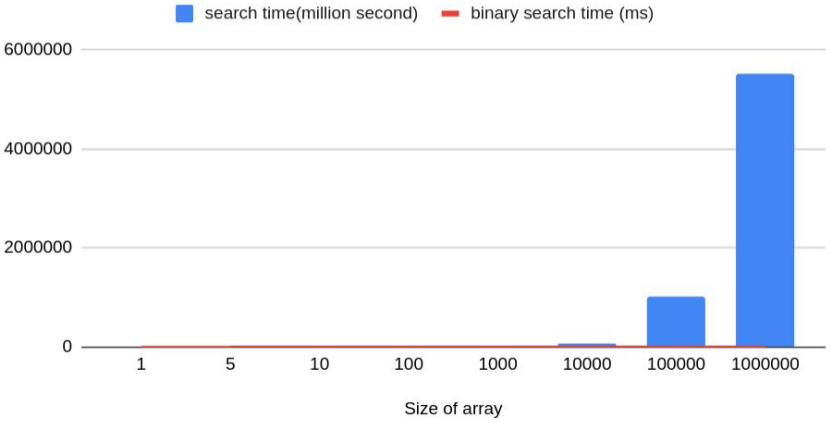
Array_size
search_number

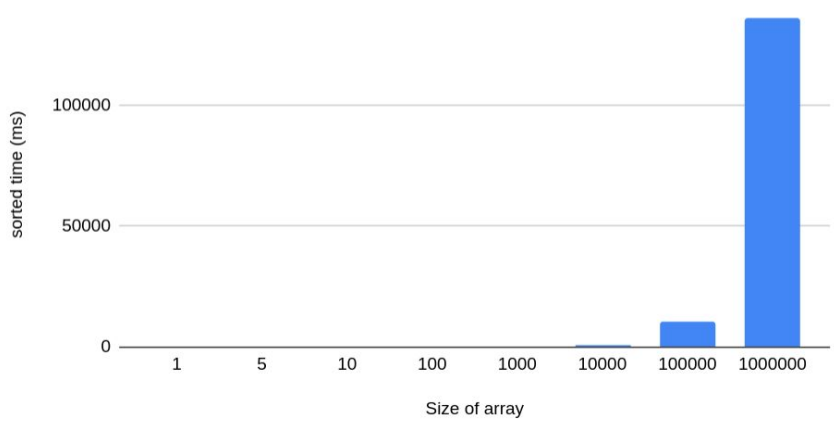
Result statistic





10000 result





Size of array	search time(ms)	binary search time (ms)	sorted time (ms)	Size of search number	sort and binary time	LOG of earch time	Log of total time
1	5	5	1	100	6	2.321928095	2.584962501
5	14	15	3	100	18	3.807354922	4.169925001
10	16	17	4	100	21	4	4.392317423
100	53	32	31	100	63	5.727920455	5.977279923
1000	473	49	382	100	431	8.885696373	8.751544059
10000	5104	63	4720	100	4783	12.31741261	12.22370008
100000	29569	15	10588	100	10603	14.85179783	13.3721849
1000000	84618	44	126266	100	126310	16.36867697	16.94660934
10000000	798745	65	1488512	100	1488577	19.60737547	20.50550242