Introduction

When we want to search something, we hope result with minimum latency. There are some algorithms to achieve get high performance. In here I am doing comprehensive performance evaluation of different searching algorithms with time. In here I am used several searching algorithms.

They are,

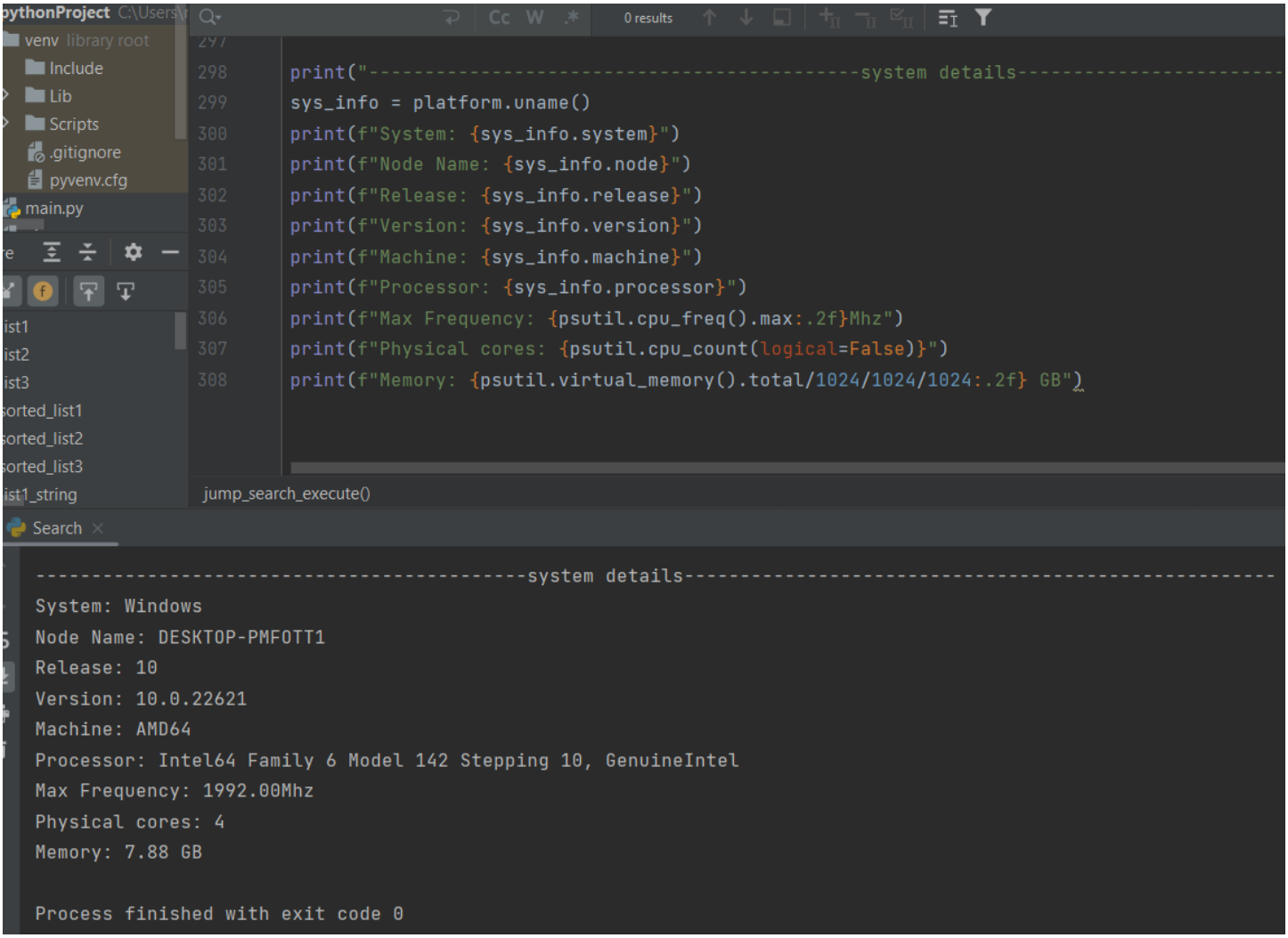
- ✓ Linear Search
- ✓ Binary Search
- ✓ Jump Search
- ✓ Interpolation Search
- ✓ Exponential Search

In here I am used different size dataset with values 1000, 10000 and 100000 which are unique numbers in the dataset. Also, it has sorted and unsorted values. Used i7 8th generation machine with windows 11 operating system and 256 GB solid state disk which has 145 GB free space.

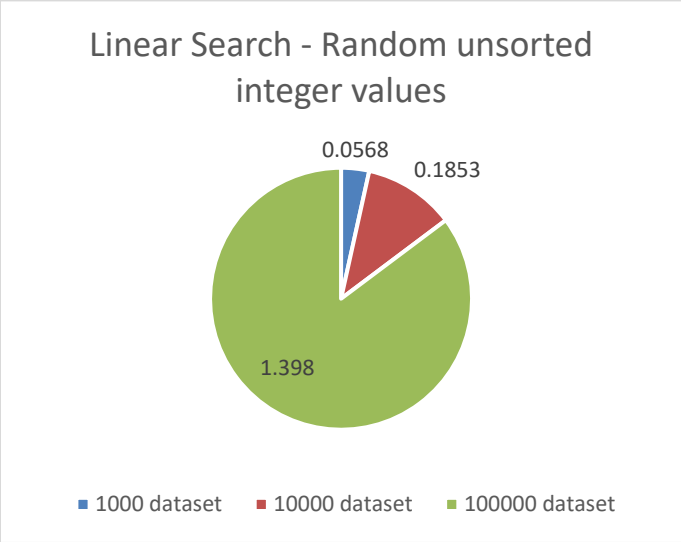
Test case description			Time performance for comparison algorithm	Describe the results of the test
Algorithm	**Variation 1**	**Variation 2**		
1. Linear Search	1000 values	Random unsorted integer values	0.0568 ms	When the size of dataset increase time also increase.
	10000 values	Random unsorted integer values	0.1853 ms	
	100000 values	Random unsorted integer values	1.398 ms	
	1000 values	Random sorted integer values	0.0524 ms	When the size of dataset increase time also increase.
	10000 values	Random sorted integer values	0.3856 ms	
	100000 values	Random sorted integer values	3.5177 ms	
	1000 values	Random string values	0.0601 ms	When the size of dataset increase time also increase.
	10000 values	Random string values	0.3105 ms	
	100000 values	Random string values	2.1221 ms	
2. Binary Search	1000 values	Random sorted integer values	0.0332 ms	Searching time change with the position of searching value.
	10000 values	Random sorted integer values	0.012 ms	
	100000 values	Random sorted integer values	0.0119 ms	
3. Jump Search	1000 values	Random sorted integer values	0.0362 ms	When the size of dataset increase time also increase.
	10000 values	Random sorted integer values	0.131 ms	
	100000 values	Random sorted integer values	1.1263 ms	

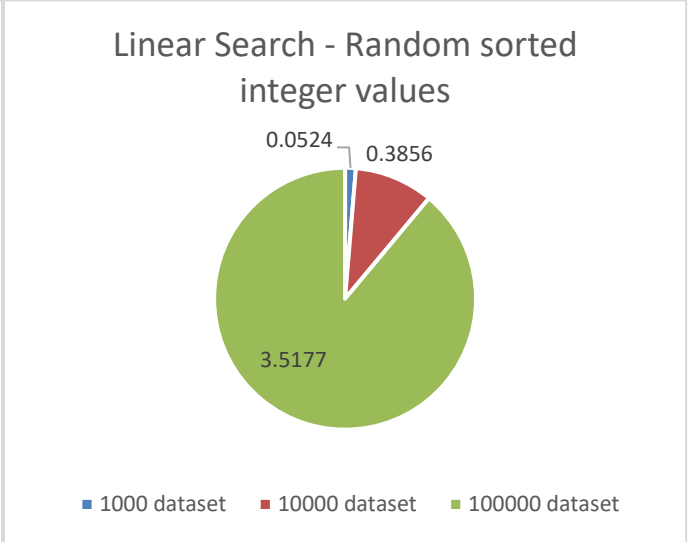
4. Interpolation Search	1000 values	Random sorted integer values	0.0097 ms	The size of the dataset is not impact to the time.
	10000 values	Random sorted integer values	0.0076 ms	
	100000 values	Random sorted integer values	0.0073 ms	
5. Exponential Search	1000 values	Random sorted integer values	0.0118 ms	The size of the dataset is not impact to the time.
	10000 values	Random sorted integer values	0.0113 ms	
	100000 values	Random sorted integer values	0.0114 ms	

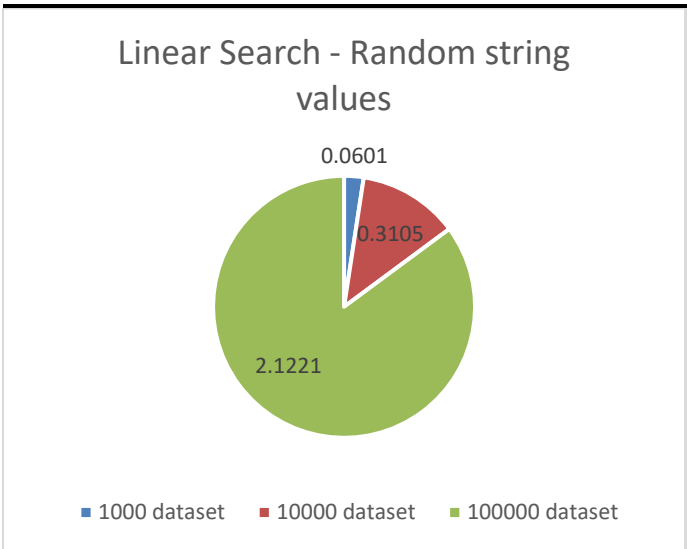
System specification of the computer

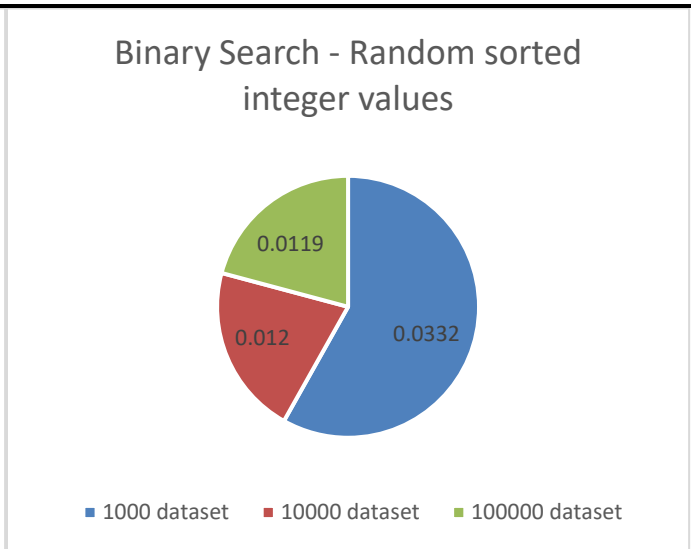


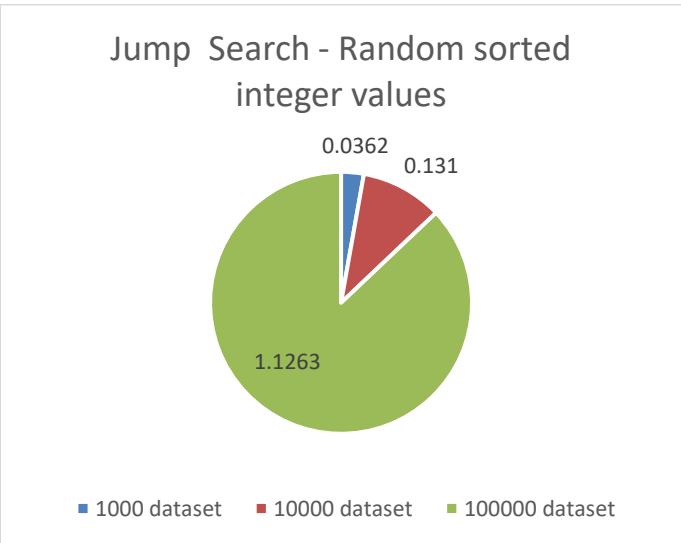
Graphs analysis

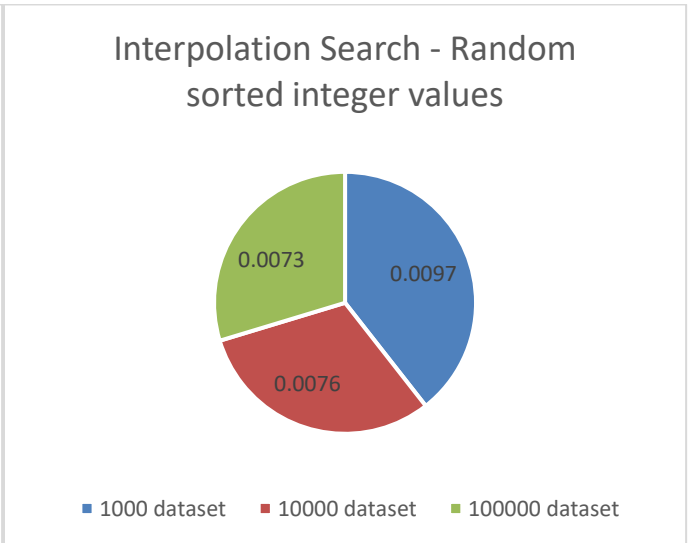


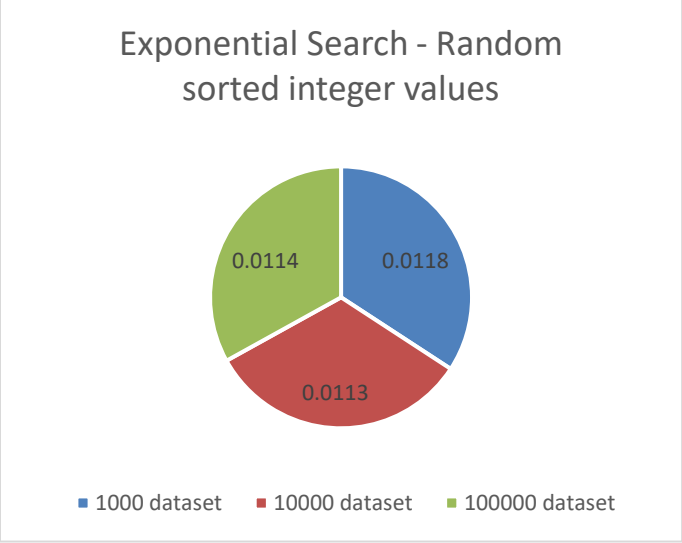




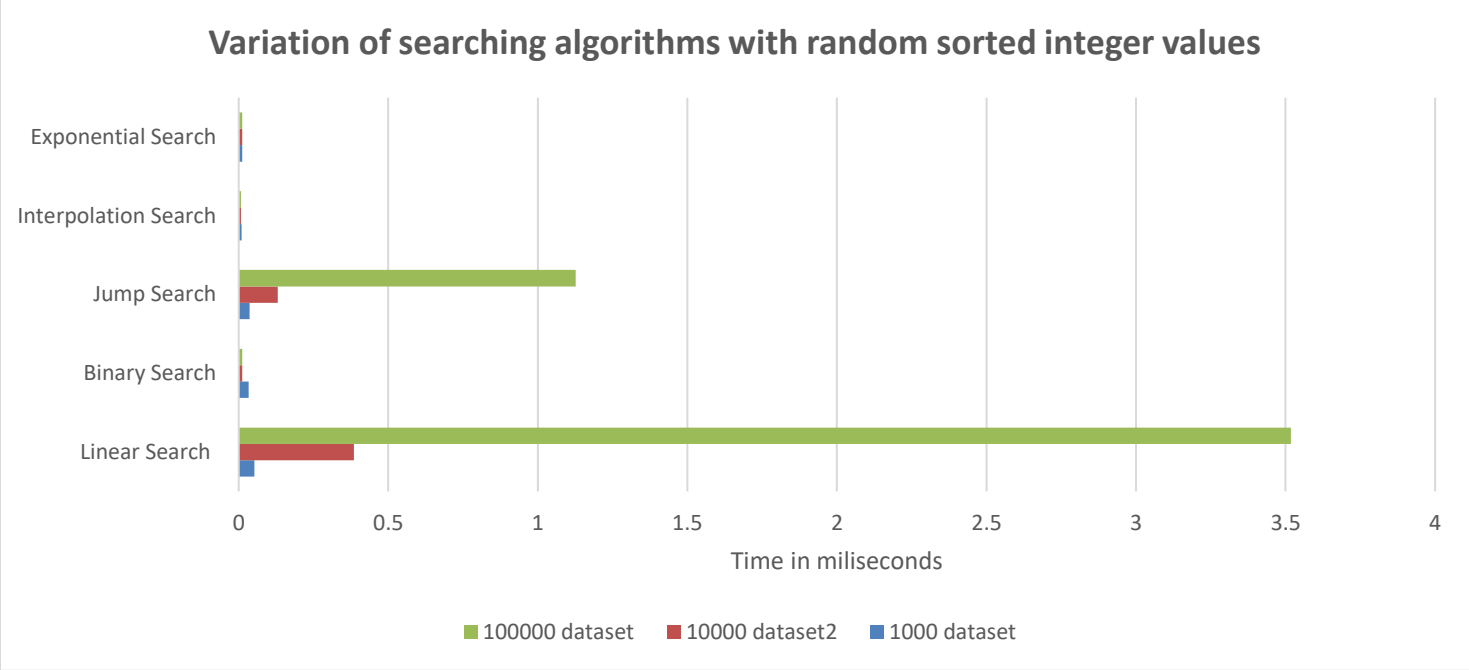








Comparison for random sorted integer values



Conclusion

With the above data we can clearly identified interpolation searching algorithm shows best performance. When we look the time of taken, we can arrange them from low to high performance like this. linear search < jump search < binary search < exponential search < interpolation search.

Another thing is average time of strings get higher value than integer. So, it says searching string is difficult than integer from this algorithm. Also, sorted data set get less time than unsorted data.

Python Source Code

1. **Linear search**

```
import platform
import random
import math
import psutil
from timeit import default_timer as dt

list1 = random.sample(range(1, 1001), 1000)
list2 = random.sample(range(1, 10001), 10000)
list3 = random.sample(range(1, 100001), 100000)
sorted_list1 = sorted(list1)
sorted_list2 = sorted(list2)
sorted_list3 = sorted(list3)


list1_string = map(str, list1)
list1_string_list = list(list1_string)
list2_string = map(str, list2)
list2_string_list = list(list2_string)
list3_string = map(str, list3)
list3_string_list = list(list3_string)


# linear search execution
================================================================================
=====
def linear_search(lists, key):
    for i in lists:
        if i == key:
            return lists.index(i)
    return -1


def linear_search_execute(lin_res):
    if lin_res == -1:

        print("not found")
    else:
        print("found ||  index is: ", lin_res)


# linear search random unsorted list with 1000 values
start_time = dt()
linear_search_execute(linear_search(list1, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search random unsorted list with 1000 values Time:
{round(execute_time,4)}ms\n")


# linear search random unsorted list with 10000 values
start_time = dt()
linear_search_execute(linear_search(list2, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search random unsorted list with 10000 values Time:
{round(execute_time,4)}ms\n")
```

```
# linear search random unsorted list with 100000 values
start_time = dt()
linear_search_execute(linear_search(list3, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search random unsorted list with 100000 values Time:
{round(execute_time,4)}ms\n")

# linear search random sorted list with 1000 values
start_time = dt()
linear_search_execute(linear_search(sorted_list1, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search random sorted list with 1000 values Time:
{round(execute_time,4)}ms\n")


# linear search random sorted list with 10000 values
start_time = dt()
linear_search_execute(linear_search(sorted_list2, 10000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search random sorted list with 10000 values Time:
{round(execute_time,4)}ms\n")

# linear search random sorted list with 100000 values
start_time = dt()
linear_search_execute(linear_search(sorted_list3, 100000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search random sorted list with 100000 values Time:
{round(execute_time,4)}ms\n")

# linear search String with 1000 values
start_time = dt()
linear_search_execute(linear_search(list1_string_list, '500'))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search String with 1000 values Time: {round(execute_time,4)}ms\n")

# linear search String with 10000 values
start_time = dt()
linear_search_execute(linear_search(list2_string_list, '500'))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search String with 10000 values Time: {round(execute_time,4)}ms\n")

# linear search String with 100000 values
start_time = dt()
linear_search_execute(linear_search(list3_string_list, '500'))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"linear search String with 100000 values Time: {round(execute_time,4)}ms\n")
```

output:

```
Search
C:\Users\rajit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\rajit\PycharmP
found ||  index is:  102
linear search random unsorted list with 1000 values Time: 0.0568ms

found ||  index is:  2140
linear search random unsorted list with 10000 values Time: 0.1853ms

found ||  index is:  23505
linear search random unsorted list with 100000 values Time: 1.398ms

found ||  index is:  999
linear search random sorted list with 1000 values Time: 0.0524ms

found ||  index is:  9999
linear search random sorted list with 10000 values Time: 0.3856ms

found ||  index is:  99999
linear search random sorted list with 100000 values Time: 3.5177ms

found ||  index is:  833
linear search String with 1000 values Time: 0.0601ms

found ||  index is:  5734
linear search String with 10000 values Time: 0.3105ms

found ||  index is:  37943
linear search String with 100000 values Time: 2.1221ms
```

2. Binary search

```
# binary search execution
================================================================================
=====
def binary_search(lists, x):
    low = 0
    high = len(lists) - 1
    mid = 0

    while low <= high:

        mid = (high + low) // 2

        if lists[mid] < x:
            low = mid + 1
        elif lists[mid] > x:
            high = mid - 1
        else:
            return mid
    return -1


def binary_search_execute(bin_res):
    if bin_res != -1:
        print("found ||  index is: ", bin_res)
    else:
        print("not found")
```

```
# binary search random sorted list with 1000 values
start_time = dt()
binary_search_execute(binary_search(sorted_list1, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"binary search random sorted list with 1000 values Time:
{round(execute_time,4)}ms\n")

# binary search random sorted list with 10000 values
start_time = dt()
binary_search_execute(binary_search(sorted_list2, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"binary search random sorted list with 10000 values Time:
{round(execute_time,4)}ms\n")

# binary search random sorted list with 100000 values
start_time = dt()
binary_search_execute(binary_search(sorted_list3, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"binary search random sorted list with 100000 values Time:
{round(execute_time,4)}ms\n")
```

output:

```
found ||  index is:  999
binary search random sorted list with 1000 values Time: 0.0332ms

found ||  index is:  999
binary search random sorted list with 10000 values Time: 0.012ms

found ||  index is:  999
binary search random sorted list with 100000 values Time: 0.0119ms
```

3. Jump search

```
# jump search execution
================================================================================
=======
def jump_search(lists, search):
    low = 0
    interval = int(math.sqrt(len(lists)))

    for i in range(0, len(lists), interval):
        if lists[i] < search:
            low = i
        elif lists[i] == search:
            return i
        else:
            break  # bigger number is found
    c = low
    for j in lists[low:]:
        if j == search:
            return c
        c += 1
    return -1
```

```
def jump_search_execute(res):
    if res == -1:
        print("not found")
    else:
        print("found ||  index is: ", res)


# jump search random sorted list with 1000 values
start_time = dt()
jump_search_execute(jump_search(sorted_list1, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"jump search random sorted list with 1000 values Time:
{round(execute_time,4)}ms\n")

# jump search random sorted list with 10000 values
start_time = dt()
jump_search_execute(jump_search(sorted_list2, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"jump search random sorted list with 10000 values Time:
{round(execute_time,4)}ms\n")

# jump search random sorted list with 100000 values
start_time = dt()
jump_search_execute(jump_search(sorted_list3, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"jump search random sorted list with 100000 values Time:
{round(execute_time,4)}ms\n")
```

output:

```
found ||  index is:  999
jump search random sorted list with 1000 values Time: 0.0362ms

found ||  index is:  999
jump search random sorted list with 10000 values Time: 0.131ms

found ||  index is:  999
jump search random sorted list with 100000 values Time: 1.1263ms
```

4. Interpolation Search

```
# interpolation Search execution
================================================================================
def nearest_mid(input_list, lower_bound_index, upper_bound_index, search_value):
    return lower_bound_index + (( upper_bound_index - lower_bound_index) //
(input_list[upper_bound_index] - input_list [lower_bound_index])) * (search_value -
input_list[lower_bound_index])


def interpolation_search(ordered_list, term):
    size_of_list = len(ordered_list) - 1
    index_of_first_element = 0
    index_of_last_element = size_of_list
    while index_of_first_element <= index_of_last_element:
        mid_point = nearest_mid(ordered_list, index_of_first_element,
index_of_last_element, term)
        if mid_point > index_of_last_element or mid_point < index_of_first_element:
            return None
        if ordered_list[mid_point] == term:
            return mid_point
        if term > ordered_list[mid_point]:
            index_of_first_element = mid_point + 1
        else:
            index_of_last_element = mid_point - 1
    return -1


def interpolation_search_execute(a):
    if a is not None:
        print("found ||  index is: ", a)
    else:
        print("not found")


# interpolation search random sorted list with 1000 values
start_time = dt()
interpolation_search_execute(interpolation_search(sorted_list1, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"interpolation search random sorted list with 1000 values Time:
{round(execute_time,4)}ms\n")

# interpolation search random sorted list with 10000 values
start_time = dt()
interpolation_search_execute(interpolation_search(sorted_list2, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"interpolation search random sorted list with 10000 values Time:
{round(execute_time,4)}ms\n")

# interpolation search random sorted list with 100000 values
start_time = dt()
interpolation_search_execute(interpolation_search(sorted_list3, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"interpolation search random sorted list with 100000 values Time:
{round(execute_time,4)}ms\n")
```

output:

```
found ||  index is:  999
interpolation search random sorted list with 1000 values Time: 0.0097ms

found ||  index is:  999
interpolation search random sorted list with 10000 values Time: 0.0076ms

found ||  index is:  999
interpolation search random sorted list with 100000 values Time: 0.0073ms
```

5. **Exponential Search**

```
# exponential Search execution
================================================================================
def exponential_search(lists, x):
    # IF x is present at first
    # location itself
    if lists[0] == x:
        return 0

    # Find range for binary search
    # j by repeated doubling
    i = 1
    while i < len(lists) and lists[i] <= x:
        i = i * 2

    # Call binary search for the found range
    return binary_search(lists, x)


def exponential_search_execute(result):
    if result == -1:
        print("not found")
    else:
        print("found ||  index is: ", result)


# exponential search random sorted list with 1000 values
start_time = dt()
exponential_search_execute(exponential_search(sorted_list1, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"exponential search random sorted list with 1000 values Time:
{round(execute_time,4)}ms\n")

# exponential search random sorted list with 10000 values
start_time = dt()
exponential_search_execute(exponential_search(sorted_list2, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"exponential search random sorted list with 10000 values Time:
{round(execute_time,4)}ms\n")
```

```
# exponential search random sorted list with 100000 values
start_time = dt()
exponential_search_execute(exponential_search(sorted_list3, 1000))
end_time = dt()
execute_time = (end_time - start_time)*1000
print(f"exponential search random sorted list with 100000 values Time:
{round(execute_time,4)}ms\n")
```

output:

```
found ||  index is:  999
exponential search random sorted list with 1000 values Time: 0.0118ms

found ||  index is:  999
exponential search random sorted list with 10000 values Time: 0.0113ms

found ||  index is:  999
exponential search random sorted list with 100000 values Time: 0.0114ms
```