5.Algorithms_search_sort

December 4, 2018

1 Searching and Sorting Algorithm examples

Example of 2 different searching algorithms - linear search and binary search

```
In [25]: # Create list
         xs = list(range(10000))
```

```
In [26]: %%timeit
         linear_search(xs, 4545)
```

```
260 ţs ś 6.2 ţs per loop (mean ś std. dev. of 7 runs, 1000 loops each)
```

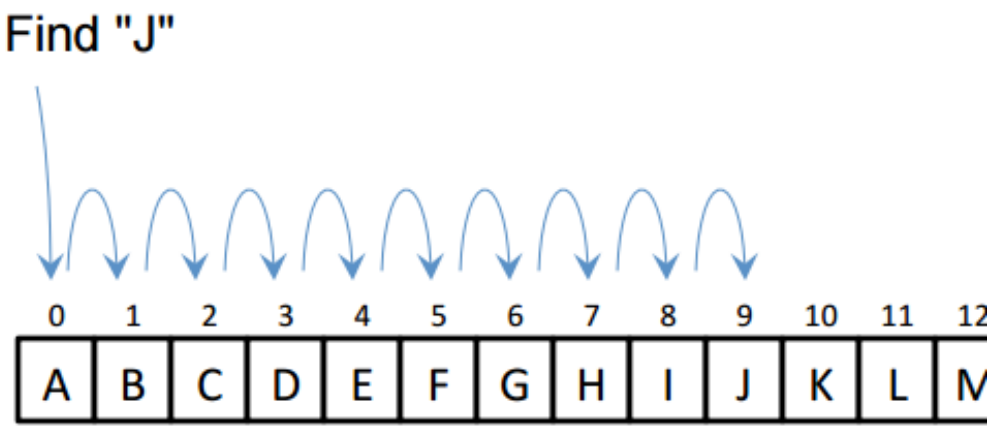
```
In [30]: %%timeit
         binary_search(xs, 4545)
```

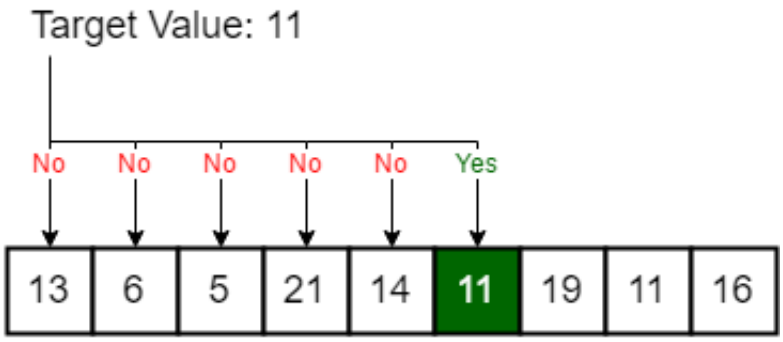
```
8 ţs ś 138 ns per loop (mean ś std. dev. of 7 runs, 100000 loops each)
```

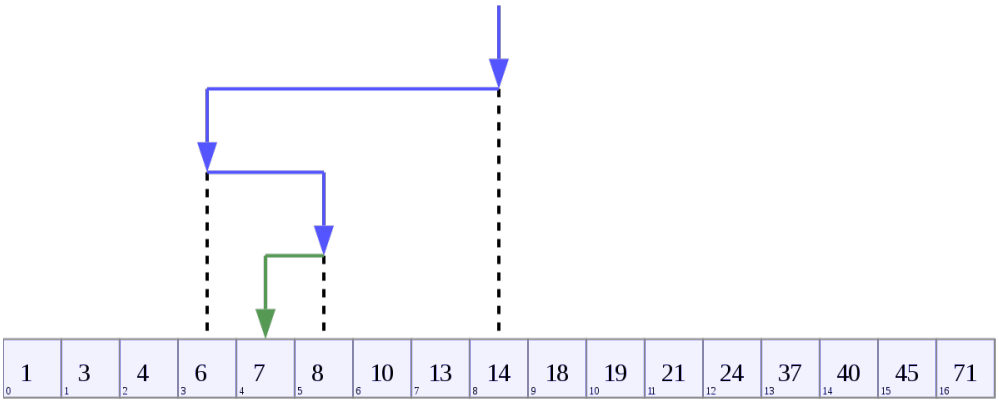
1.1 Linear Search and emulation of list index method

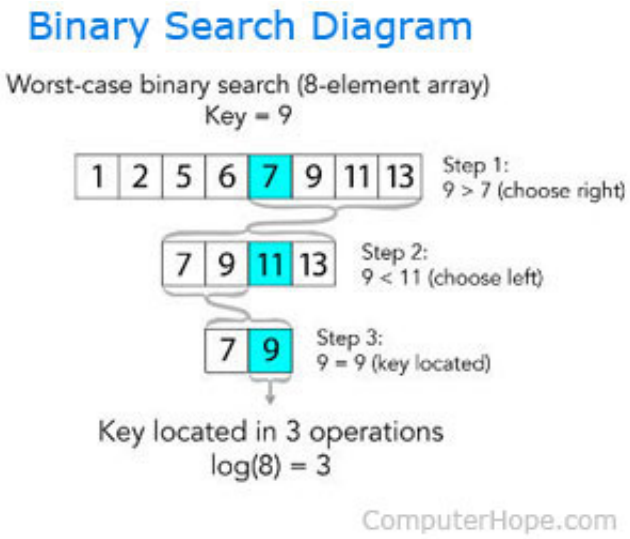
```
In [31]: # Specify what to find
         searching_element = 45
         # Initialize index as None
         index = None

         # Iterate over elements in a list with their indices
         # Assign position of element equal to searching_element to index
         for i, elem in enumerate(xs):
             if elem == searching_element:
                 index = i
                 break

         print(index)
```

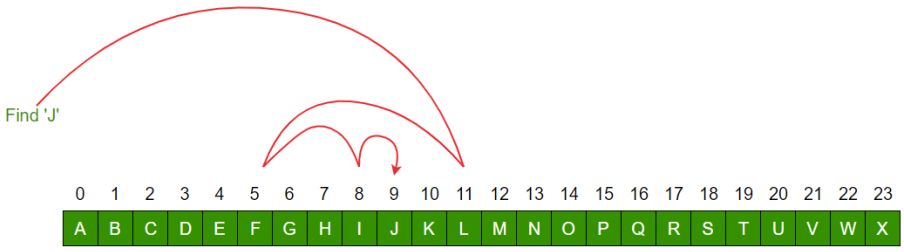
```
45
```


Find "J"
0 1 2 3 4 5 6 7 8 9 10 11 12
A B C D E F G H I J K L M


Target Value: 11
No No No No No Yes
13 6 5 21 14 11 19 11 16


1 3 4 6 7 8 10 13 14 18 19 21 24 37 40 45 71


Binary Search Diagram
Worst-case binary search (8-element array)
Key = 9
1 2 5 6 7 9 11 13
Step 1:
9 > 7 (choose right)
7 9 11 13
Step 2:
9 < 11 (choose left)
7 9
Step 3:
9 = 9 (key located)
Key located in 3 operations
log(8) = 3
ComputerHope.com


Find 'J'
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
A B C D E F G H I J K L M N O P Q R S T U V W X

1.2 Binary Search

In [17]:
```
import math
from IPython.lib.display import YouTubeVideo
```

In [32]:
```
# Specify element to search
searching_element = 0
# Find length of list, and its half
length = len(xs)
delta = length // 2
index = delta

# We start to search at the middle of list
# On each iteration we compare chosen element with one which we need to find
# and exploiting the fact that list is sorted we decide where to go (right/left/stay)
# next we slide half of the current delta in necessary direction and reduce delta to ha
for i in range(round(math.log2(length)) + 1):
    # Compute range to slide in a list
    delta = max(round(delta / 2), 1)
    print(index, delta, xs[index])
    # Element found
    if xs[index] == searching_element:
        break
    # Element somewhere in the greater part of list
    elif xs[index] < searching_element:
        index += delta
    # Element somewhere in the lesser part of list
    else:
        index -= delta

print(index)
```

```
5000 2500 5000
2500 1250 2500
1250 625 1250
625 312 625
313 156 313
157 78 157
79 39 79
40 20 40
20 10 20
10 5 10
5 2 5
3 1 3
2 1 2
1 1 1
0
```

```
In [28]: def linear_search(xs, ele):
             for i, elem in enumerate(xs):
                 if elem == ele:
                     return i

         def binary_search(xs, elem):
             # Find length of list, and its half
             length = len(xs)
             delta = length // 2
             index = delta

             for i in range(round(math.log2(length)) + 2):
                 delta = max(round(delta / 2), 1)

                 if xs[index] == elem:
                     return index
                 elif xs[index] < elem:
                     index += delta
                 else:
                     index -= delta

In [29]: binary_search(range(100000), 135)

Out[29]: 135
```

1.3 Mergesort illustration

```
In [24]: YouTubeVideo('61Z9-RPO4xk', width=750, height=400)

   Out[24]:
```


COMPARISONS:
09