

빅 데이터 혁신 공유 대학

파이썬으로 배우는 데이터 구조

한동대학교 전산전자공학부

김영섭 교수



교육부



한국연구재단



Data Structures in Python

Chapter 5 - 1

- Binary Search
- **Recursive Binary Search**
- Bubble sort
- Selection sort
- Insertion sort

Agenda & Readings

- Binary Search
- **Recursive Binary Search**

Recursive Binary Search

- Binary search is an efficient algorithm for finding an item from **a sorted list** of items.
 - It works by repeatedly **dividing in half the portion of the list** that could contain the item, until you've **narrowed down the possible locations to just one**.

key=23

0	1	2	3	4	5	6	7	8	9
2	5	8	9	16	23	31	56	62	71

Recursive Binary Search

- For instance, we want to search "23" from the array. If we find it, we return its array index; otherwise, -1 or something else.

key=23

0	1	2	3	4	5	6	7	8	9
2	5	8	9	16	23	31	56	62	71

(lo+hi)//2=mi

(0+9)//2=4

key>mi
23>16

lo=0	1	2	3	mi=4	5	6	7	8	hi=9
2	5	8	9	16	23	31	56	62	71

(5+9)//2=7

key<mi
23<56

0	1	2	3	4	lo=5	6	mi=7	8	hi=9
2	5	8	9	16	23	31	56	62	71

(5+6)//2=5

key=mi
23=23

0	1	2	3	4	mi=5 lo=5	hi=6	7	8	9
2	5	8	9	16	23	31	56	62	71

Recursive Binary Search

- For instance, we want to search "23" from the array. If we find it, we return its array index; otherwise, -1 or something else.

0	1	2	3	4	5	6	7	8	9
2	5	8	9	16	23	31	56	62	71

lo=0	1	2	3	mi=4	5	6	7	8	hi=9
2	5	8	9	16	23	31	56	62	71

0	1	2	3	4	lo=5	6	mi=7	8	hi=9
2	5	8	9	16	23	31	56	62	71

0	1	2	3	4	mi=5 lo=5	hi=6	7	8	9
2	5	8	9	16	23	31	56	62	71

key	key sought
list	sorted array
mi	midpoint
lo	smallest possible index
hi	largest possible index

```
def binary_search(list, key, lo, hi):
    if lo > hi: return -1

    mi = (lo + hi) // 2
    if key == list[mi]: return mi
    if key < list[mi]:
        return binary_search(list, key, lo, mi-1)
    else:
        return binary_search(list, key, mi+1, hi )
```

Recursive Binary Search

- Given the numbers 1 to 100, what is **the number of guesses** at most needed to find a specific number if you are given the hint 'higher' or 'lower' for each guess you make?
 - Since the numbers are sequential (or sorted), we can use binary search.
 - Look at the middle element: if it's after than the number we're looking for, search the first half. If it's before the number we're looking for, look at the second half.
 - Each check cuts the size of the list numbers in half; how many times can we do this?
 - If we think backwards, in terms of doubling the list, we'll need n doublings to generate a list of length $2^n = 100$. What is the value of n ?
 - Since $2^6 = 64$ and $2^7 = 128$ (or $\log_2 64 = 6$, $\log_2 128 = 7$), $n = 6.x$
Therefore $n = 7$ guesses will be enough.
- Binary search guarantees to take no more than $\log n$ guesses or comparison.
- More precisely, $\lceil \log_2 n \rceil$ if n is not a power of 2.
- <https://jwoop.tistory.com/9>

Exercise: Exception Filter

- The **exception filter** reads in a sorted array of strings from a file(which we refer to as the **whitelist**) and an arbitrary sequence of strings from standard input and write those in the sequence that are **not** in the **whitelist**.
- For example:
 - When checking the spelling of a word, you need to know only whether your word is in the dictionary (or **whitelist**) and are not interested in the definition.
- Another examples:
 - Your email application might use an **exception filter** to reject any messages that are not on a **whitelist** that contains the email addresses of your friends.
 - Your operating system might have an **exception filter** that disallows network connections to your computer from any device having an IP address that is not on a preapproved **whitelist**.
 - In a computer search, we keep the whitelist sorted in order of the key.
 - The array length n need not be a power of 2.

Exercise: Exception Filter

- **Task:** Find and print all misspelled words in `words.txt` using binary search algorithm. The correct words in `dict.txt` are listed in sorted order. A skeleton code, `dict.txt` and `words.txt` files are provided.
 - When checking the spelling of a word, you need to know only whether your word is in the dictionary (or **whitelist**) and are not interested in the definition.
 - We keep the **whitelist** sorted in order of the key.
 - The array length **n** need not be a power of 2.

whitelist in
sorted order

```
$ more dict.txt
accommodate
broccoli
conscience
definitely
embarrass
necessary
occurred
publicly
receive
separate
until
```

word list to check
misspelled

```
$ more words.txt
definatly
occured
occurred
untill
until
recieve
acommodate
seperate
separate
```

redirect standard
input through the file

```
$ python exfilter.py < words.txt
definatly
occured
untill
recieve
acommodate
seperate
```

misspelled words,
not in dict.txt (or
whitelist)

Exercise: Exception Filter

- Skeleton Code:

```
%%writefile exfilter.py
import sys

def binary_search(arr, key, lo, hi):
    pass

def search(arr, key):
    pass

if __name__ == "__main__":
    with open('dict.txt') as f:
        arr = f.read().splitlines()
    for line in sys.stdin:
        key = line.rstrip()
        if search(arr, key) < 0:
            print(key)                # not found in dict
```

```
%%cmd
python exfilter.py < words.txt
```

i/o reference: <https://stackoverflow.com/questions/1450393/how-do-you-read-from-stdin>

Exercise: Exception Filter

- Sample Runs in Jupyter Lab and in Windows cmd shell:

```
Overwriting exfilter.py

[2]: %%cmd
python exfilter.py < words.txt

Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. All rights reserved.

C:\GitHub\DSpyx\jupyter>python exfilter.py < words.txt
definitely
occured
untill
recieve
acommodate
seperate

C:\GitHub\DSpyx\jupyter>

[3]: %%bash
python exfilter.py < words.txt

definitely
occured
untill
recieve
acommodate
seperate
```

Note: This does not work in Windows PowerShell. Use cmd.

```
C:\GitHub\DSpyx\jupyter>cmd
Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. All rights reserved.

C:\GitHub\DSpyx\jupyter>python exfilter.py < words.txt
definitely
occured
untill
recieve
acommodate
seperate

C:\GitHub\DSpyx\jupyter>
```

i/o reference: <https://stackoverflow.com/questions/1450393/how-do-you-read-from-stdin>

Summary

- Binary search is simple, but powerful!
- Binary search may be implemented using either iteration or recursion.
- Its time complexity is $O(\log n)$.

Data Structures in Python

Chapter 5 - 1

- Binary Search
- **Recursive Binary Search**
- Bubble sort
- Selection sort
- Insertion sort