Q1 Dual Search

Q2 Dual Sorted Search

Q3 Compare performance between Q1 and Q2

Q4 Find median of two sorted arrays

Q1 Dual Search

10 mins

Given an array of n elements. Find two elements in the array such that their sum is equal to K. The two elements can be the same element. Once a pair of elements is found, the program can be terminated. The function prototype is given below:

```
def dual_search(A, size, K, dual_index):
    #A (list): The input array of integers.
    #size (int): The size of the array.
    #K (int): The target sum.
    #dual_index (list): A list to store the indices of the two elements.
    #Returns (bools): True if a pair is found, False otherwise
```

Sample input:	Output:
A = [3, 1, 7, 4, 5, 9]	Pair found at indices: [0, 4]
K = 8	Elements: 3 + 5 = 8

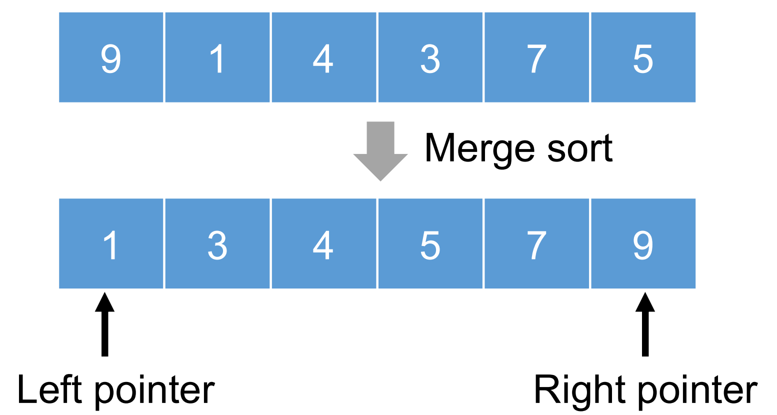
- Two for loops
- A[i] + A[j] == K?

Q2 Dual Sorted Search **10 mins**

Given a sorted array of n elements (you can use merge sort to sort the array). Find two elements in the array such that their sum is equal to K. The two elements can be the same element. Once a pair of elements are found, the program can be terminated. The results may be different from the results of Question 1. The function prototype is given below:

```
def dual_sorted_search(A, size, K, dual_index):
    #A (list): The input array of integers.
    #size (int): The size of the array.
    #K (int): The target sum.
    #dual_index (list): A list to store the indices of the two elements.
    # Returns (bools): True if a pair is found, False otherwise
```

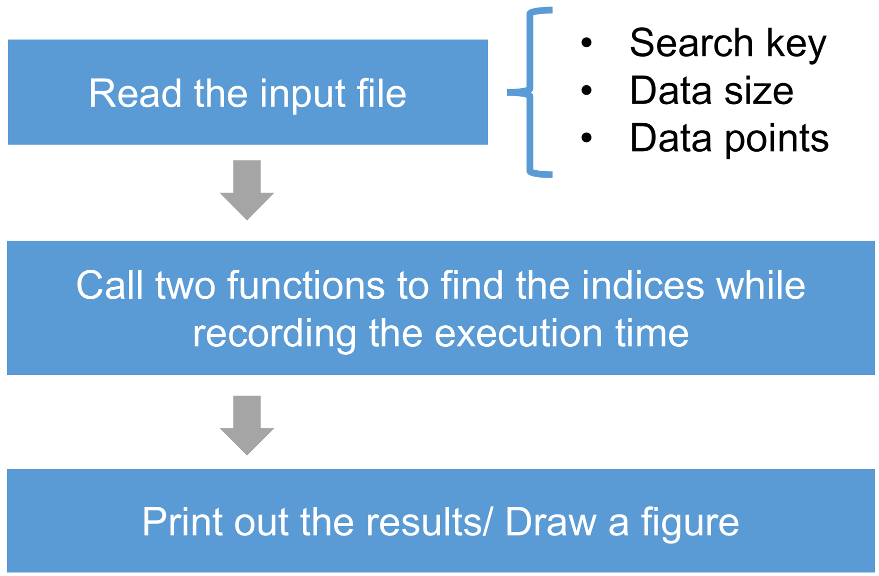
Sample input:	Output:
A = [9, 1, 4, 3, 7, 5]	Pair found at indices: [0, 4]
K = 8	Elements: 1 + 7 = 8



- **== K**: Return the pair
- **< K**: Move the left pointer right (increase sum)
- **> K**: Move the right pointer left (decrease sum)

Q3 Compare performance between Q1 and Q2 **10 mins**

Compare the performance between Q1 and Q2. Try to use a large input file to evaluate their running time. A 500k data and 1 million data file are attached. The first line is a search key and the second line is a data size. The rest are the data.



Q3 Compare performance between Q1 and Q2

Apply for VS Code

Differences between the paths for package import and file access

Module import mechanism

- When using import xx to import a .py file in the current folder, Python will search for it according to the path in sys.path.
- Usually, when running a script, Python will automatically add the directory where the current script is located to sys.path, so we can import directly by module name without writing the path.
- How about import .py files in other folders?

Path parsing for file reading (open)

- When using the $open$ function to read a file, by default, the file path is parsed relative to the current working directory.
- In VS Code, the working directory is usually the root directory of the project, not the directory where a single script is located.

if __name__ == "__main__": used when code should be executed only when a file is run as a script rather than imported as a module.

Q4 Find median of two sorted arrays

Given two arrays num1 and num2. Both are sorted in an ascending order. The length is m and n, respectively. Please find the median of the two arrays **with time complexity being $O(\log(m+n))$**. You can **apply binary search to partition the arrays**. The function prototype is given below:

```
def find_median_sorted_arrays(num1, num2):
    #num1 (list): First sorted array.
    #num2 (list): Second sorted array.
    #Returns (float): The median of the combined sorted array.
```

Sample input:
num1 = [1, 3, 8]
num2 = [7, 9, 10, 11]

Output:
Median of the two sorted arrays is: 8

First reaction: merge two lists and find the median directly

→ Time complexity: O(m+n)

Binary search → O(log(m+n))

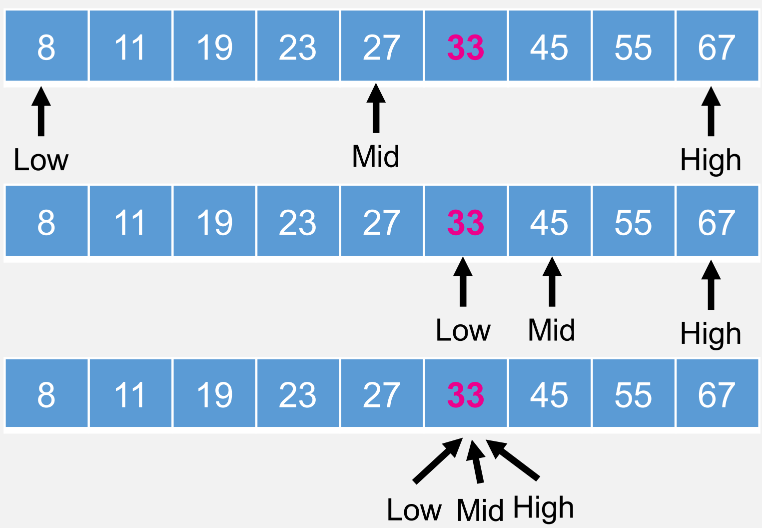
Q4 Find median of two sorted arrays

Binary Search

A searching algorithm used in a sorted array by repeatedly dividing the search interval in half => O(log N).

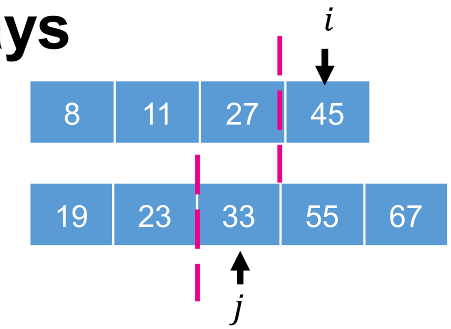
E.g., Find 33 in sorted array [8, 11, 19, 23, 27, 33, 45, 55, 67].

Mid = (Low +High) // 2



num_1: n=4

num_2: m=5

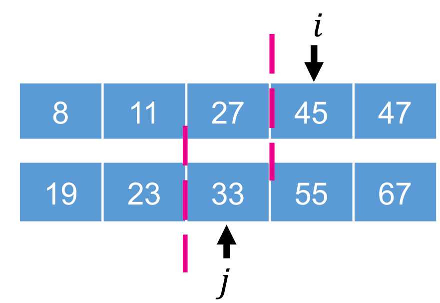


IF i elements from num_1 and j elements from num_2 are on the left half of the median (including the median itself), then

$$Median \; = \max(num_1[i], num_2[j])$$

num_1: n=5

num_2: m=5



$$Median \; = \frac{num_1[i] + num_2[j]}{2}$$

Find out such i and j using binary search, so

$$\boldsymbol{num_1[i-1] \leq num_2[j]}$$
$$\boldsymbol{num_2[j-1] \leq num_1[i]}$$

Q4 Find median of two sorted arrays

8	11	27	45

$len(num_1) = m = 4$

19	23	33	55	67

$len(num_2) = n = 5$

8	11	19	23	27	33	45	55	67

Mid

$$Median = i + j = \frac{\boldsymbol{m+n+1}}{\mathbf{2}} = k\ th$$

Given two arrays, find the "i" in the smaller array, then there should be $j = (m+n+1)/2 - i$ elements in another array.

How to validate the correct "i" and "j"?

$$\boxed{\begin{array}{l} \boldsymbol{num_1[i-1] \le num_2[j]} \\ \boldsymbol{num_2[j-1] \le num_1[i]} \end{array}}$$

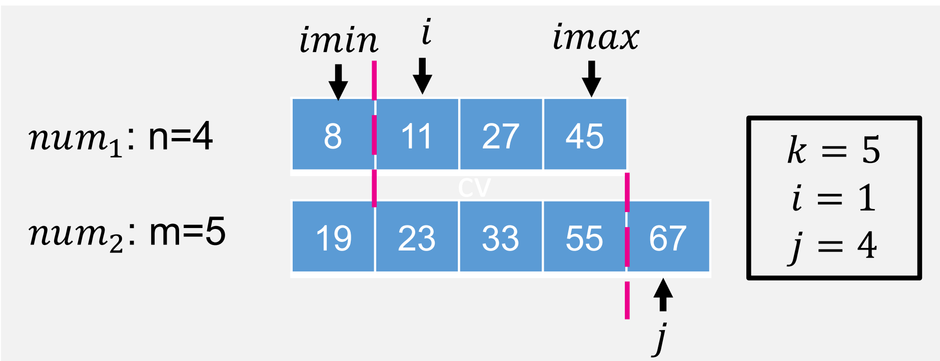
Then we can definitely say that these are the elements on the left of the median.

Otherwise, we have to reduce the number of elements taken from any one array.

- If $num_1[i] < num_2[j-1]$, consider more elements from the first array
- Otherwise, if $num_2[j] < num_1[i-1]$, consider more elements from the second array

Q4 Find median of two sorted arrays

$$Median \ = i + j = \frac{m+n+1}{2} = k\ th$$

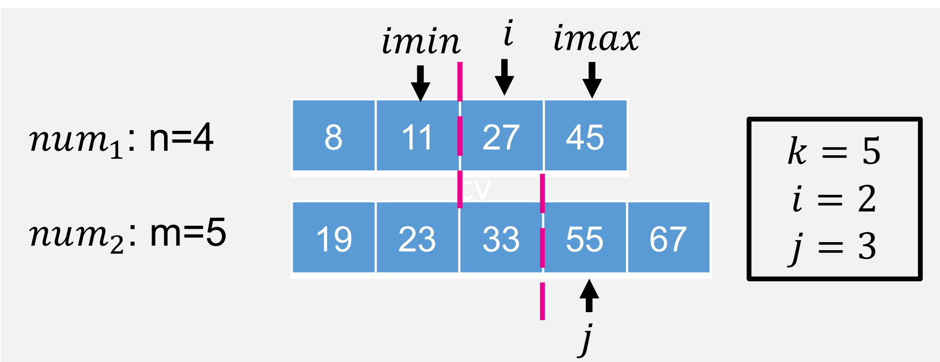


$num_1[i] \geq num_2[j-1]$
$num_2[j] \ \geq num_1[i-1]$

Because $num_1[i] \ < num_2[j-1]$ (11 < 55), consider more elements from the first array

$$\boldsymbol{imin = i + 1}$$

8	19	23	33	55	11	27	45	67



Because $num_1[i] \ < num_2[j-1]$ (27 < 33), consider more elements from the first array

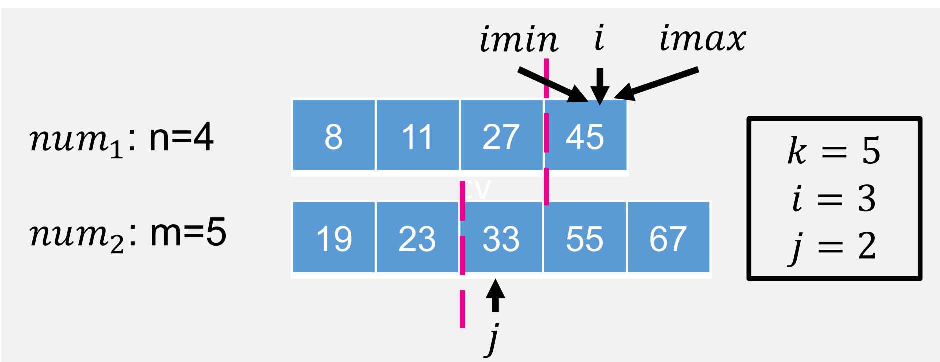
$$\boldsymbol{imin = i + 1}$$

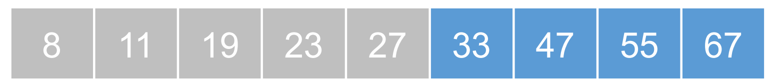
8	11	19	23	33	27	45	55	67

Q4 Find median of two sorted arrays

$$Median \; = i + j = \frac{m+n+1}{2} = k\ th$$

$$\boxed{\begin{array}{l} num_1[i] \geq num_2[j-1] \\ num_2[j] \; \geq num_1[i-1] \end{array}}$$





$$num_1[i-1] \; = \; 27$$
$$num_2[j-1] = \; 23$$
$$\max(num_1[i-1], num_2[j-1]) = 27$$