National Workshop 6

Schedule

Session 1	Algorithms III: Sorting
11.00 – 11.30	Tea/Coffee
Session 2	Testing and evaluation
13.00 – 14.00	Lunch
Session 3	Digital portfolios and CWA video

Key Messages for NW6

There are many ways to use the LCCS specification.

The study of Computers and Society is one of the overarching principle of LCCS

LCCS can be mediated through a constructivist pedagogical approach.

ALTs

ALTs provide an opportunity to teach theoretical aspects of LCCS.

Critical reflection will be a central component of the student experience and the LCCS teacher's PD journey.

Digital technologies can be used to enhance collaboration, learning and reflection.

- Focus of the session

Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

Session 1

Algorithms III - Sorting

Overview of the Session

Part 1	Introduction to algorithms (revisited)
Movement Break	
Part 2	Sorting algorithms: Selection Sort, Insertion Sort and Bubble Sort
Movement Break	
Part 3	Quicksort

By the end of this session participants will have:

reflected on the **definition** and **characteristics** of algorithms, as well as the ubiquitous nature of algorithms in today’s society.

developed their conceptual understanding of a variety of **sorting algorithms**.

participated in activities to **facilitate** the **effective learning** of algorithms in their own classrooms.

Section I

Introduction to Algorithms (revisited)

Algorithms and the Specification

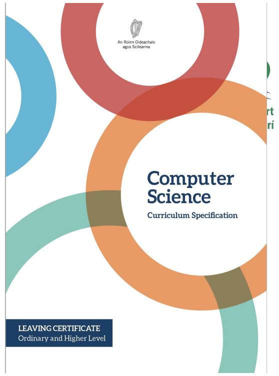
"Computer science is the study of computers and algorithmic processes. Leaving Certificate Computer Science includes how programming and computational thinking can be applied to the solution of problems, and how computing technology impacts the world around us. "

NCCA Curriculum specification, Page 1

Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
Computers and society	Abstraction	Applied learning task 1 - Interactive information systems
Computational thinking	Algorithms	Applied learning task 2 - Analytics
Design and development	Computer systems	Applied learning task 3 - Modelling and simulation
	Data	Applied learning task 4 - Embedded systems
	Evaluation/Testing	

NCCA Curriculum specification, Page 11

LCCS Learning Outcomes



2.5 use pseudo code to outline the functionality of an algorithm

2.6 construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement

2.7 implement algorithms using a programming language to solve a range of problems

2.8 apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm

2.9 assemble existing algorithms or create new ones that use functions (**including recursive**), procedures, and modules

2.10 explain the common measures of algorithmic efficiency using any algorithms studied

S2: Algorithms

Programming concepts

Sorting: Simple sort, Insert sort, Bubble sort, **Quicksort**

Search: Linear search, Binary search

Algorithmic complexity

See also learning outcomes 1.6, 1.7 1.14, 1.22, 2.3, 3.4 and 3.7 … plus others

Activity: Features of algorithms

“What features of an algorithm are demonstrated in the video?”

In what other contexts do you think the Gale-Shapley algorithm could be used?

Reflection: Features of algorithms

What features of an algorithm are demonstrated in the video?

Go to menti.com and enter the code:

7317 4422

OR

Scan the QR code on the right.

What is an algorithm?

"An algorithm is a set of rules for getting a specific output from a specific input. Each step must be so precisely defined that it can be translated into computer language and executed by machine"

Source: Knuth, D The Art of Computer Programming (Vol. 1, Fundamental Algorithms, 3rd ed.)

Donald Knuth

According to Knuth, an algorithm has five important features:

Finiteness	An algorithm must always terminate after a finite number of steps.
Definiteness	Each step must be precisely defined.
Input	An algorithm has zero or more inputs.
Output	An algorithm has one or more outputs, which have a specified relation to the inputs.
Effectiveness	All operations to be performed must be sufficiently basic that they can in principle be done exactly and in finite length of time by someone using pencil and paper.

What is an algorithm?

A step-by-step procedure for solving a problem or accomplishing some end especially by a computer

Merriam-Webster

- ✔ A sequence of instructions
- ✔ A way of capturing intelligence and sharing it with others
- ✔ Provide general solutions to problems
- ✔ Some problems are so hard that they cannot be solved by algorithms (Computability)
- ✔ Can be expressed in a variety of different ways
- ✔ Common elements include input, processing, output
- ✔ Close relationship between algorithms and data structures
- ✔ Essential features are correctness and effectiveness
- ✔ Rule-based algorithms vs. Machine learning algorithms (AI)

2 minute movement break

Section II

Sorting algorithms: Selection Sort, Insertion Sort and Bubble Sort

Sorting and LCCS

2.8 apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm

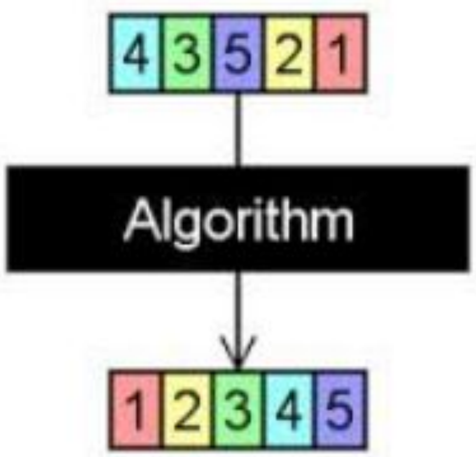
S2: Algorithms
Programming concepts
Sorting: Simple sort, Insert sort, Bubble sort, **Quicksort**
Search: Linear search, Binary search
Algorithmic complexity

Sorting algorithms

An algorithm that maps the following input/output pair is called a sorting algorithm:

Input: A list (or array), L, that contains n orderable elements: $\mathrm{L}[0, 1, \ \ldots \ , \mathrm{n}-1]$.

Output: A sorted permutation of L, called S, such that $S[0] \leq S[1] \leq \cdots \leq S[\mathrm{n}-1]$.



Simple (selection) Sort

Insertion Sort

Bubble Sort

Quicksort

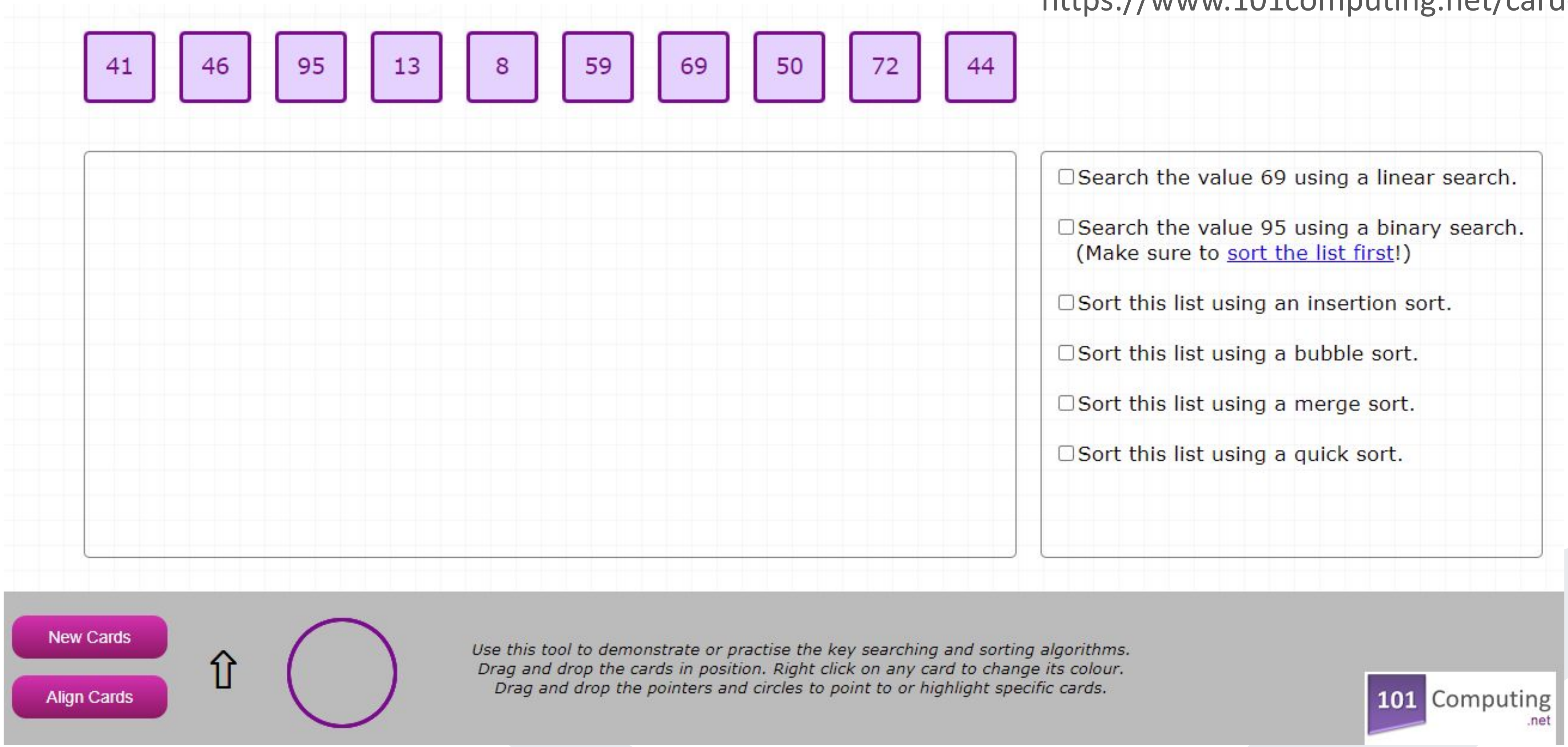
Simple Sort Demonstration

Input

Required Output:

Simple Sort Demonstration

https://www.101computing.net/card-sort/



Group Activity/Breakout

Group Activity / Breakout

Instructions :

1. Individuals read the algorithm provided and develop their own understanding (5 mins)

2. Each group then discusses and agrees a common understanding of their assigned algorithm (5 mins)

3. Groups prepare a demonstration/explanation which they will use to teach others after the breakout (5 mins)

(https://www.101computing.net/card-sort/)
https://deck.of.cards/

Selection Sort

Insertion Sort

Bubble Sort

Group Activity/Breakout

Groups 1,4,7	Selection Sort	Pages 20-24
Groups 2,5,8	Insertion Sort	Pages 25-31
Groups 3,6,9	Bubble Sort	Pages 32-39

Appoint a chair, a timekeeper, a notetaker and a spokesperson

Feedback

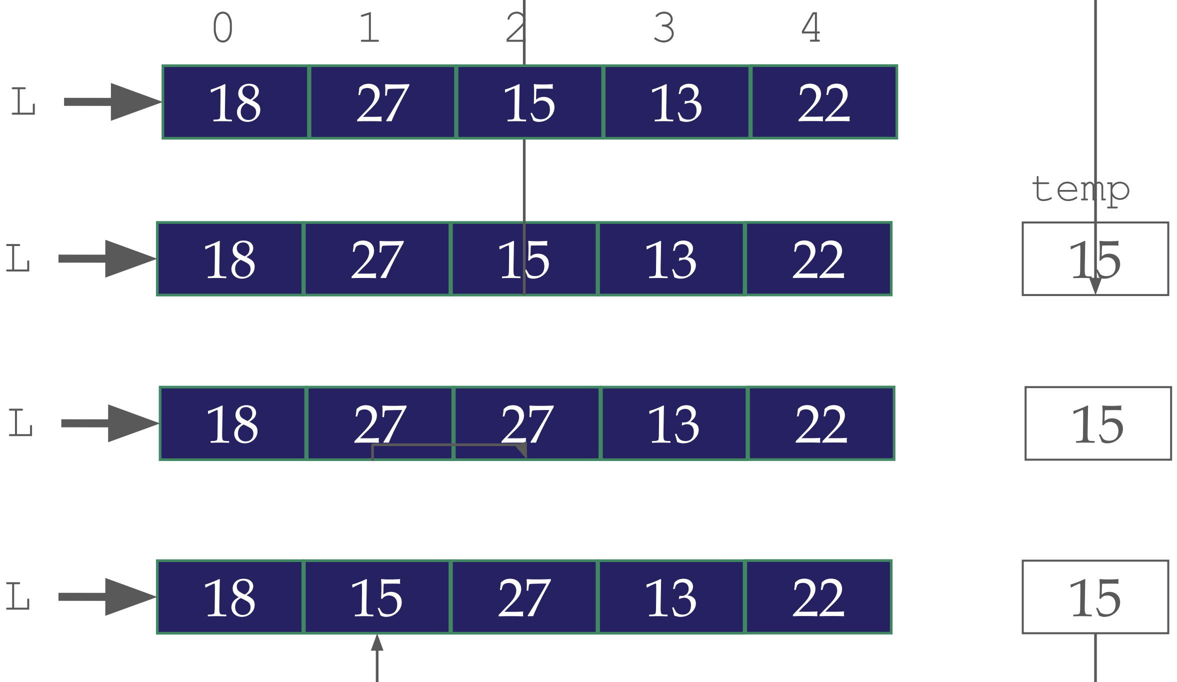
Sorting … key skills

- ✔ Lists
- ✔ For loops
- ✔ Find min/max (if statements/comparison operators)
- ✔ Swap operation (assignment)
- ✔ Functions
- ✔ sorted (built in function) vs sort (list method)

The Swap Operation

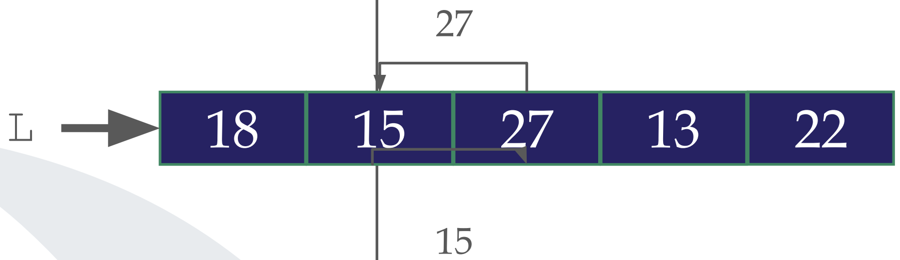
Let's say we wanted to exchange `L[2]` and `L[1]`

```
L = [18, 27, 15, 13, 22]

temp = L[2]

L[2] = L[1]

L[1] = temp
```



Python supports a single statement swap:

```
L[2], L[1] = L[1], L[2]
```



2 minute movement break

Section III

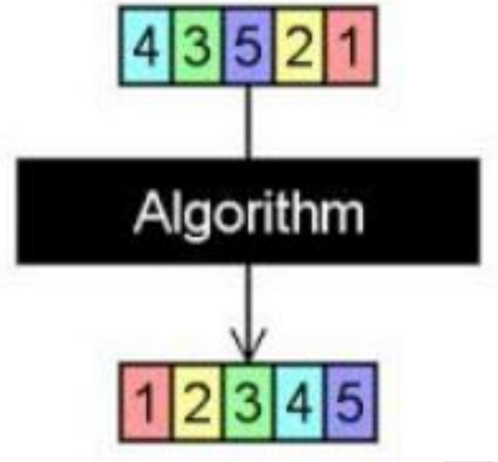
Quicksort

Quicksort

An algorithm that maps the following input/output pair is called a sorting algorithm:

Input: A list or array, A, that contains n orderable elements: $\mathrm{A}[0, 1, \ldots, \mathrm{n}-1]$.

Output: A sorted permutation of A, called B, such that $\mathrm{B}[0] \leq \mathrm{B}[1] \leq \cdots \leq \mathrm{B}[\mathrm{n}-1]$.



A general sorting algorithm devised by Tony Hoare in the late 1950s.

Quicksort: the basic idea

<u>**DIVIDE**</u>

1. Pick some number p from the list – called the pivot

2. Partition all the data into:

A. The values less than the pivot (call this the left list)
B. The pivot (call this the middle list)
C. The values greater than the pivot (call this the right list)

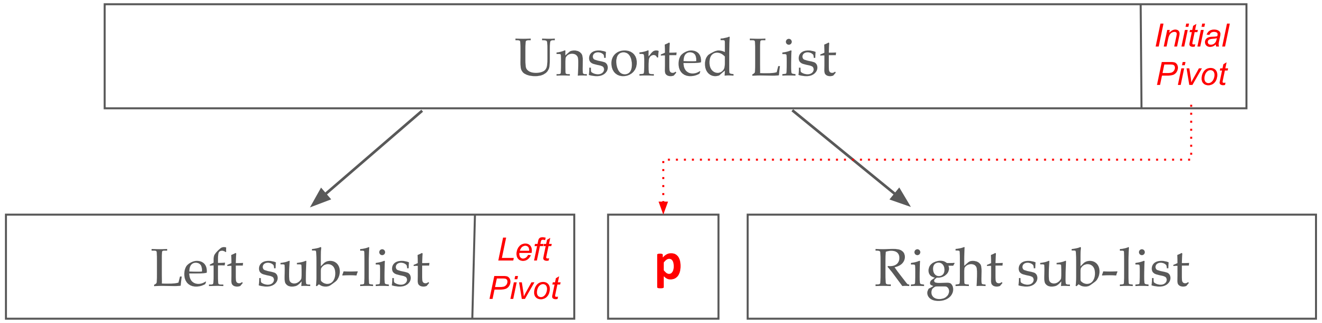
<u>**CONQUER**</u>

3. Quicksort the left list (A)

4. Quicksort the right list (B)

5. The answer is left list + middle list + right list

Partitioning



STEP 1. Choose the rightmost element in the list as the `pivot`

STEP 2. Create three empty lists called `left_list`, `middle_list` and `right_list`

STEP 3. for each `key` (item) in the list

- if `key` is < `pivot` add it to `left_list`
- if `key` is == `pivot` add it to `middle_list`
- if `key` is > `pivot` add it to `right_list`

```python
def quick_sort(L):
    left_list = []
    middle_list = []
    right_list = []

    # Base case
    if len(L) <=1:
        return(L)

    # Set pivot to the last element in the list
    pivot = L[len(L)-1]

    # Iterate through all elements (keys) in L
    for key in L:
        if key < pivot:
            left_list.append(key)
        elif key == pivot:
            middle_list.append(key)
        else:
            right_list.append(key)

    # Repeat the quicksort on the sub-lists and combine the results
    return quick_sort(left_list) + middle_list + quick_sort(right_list)
```

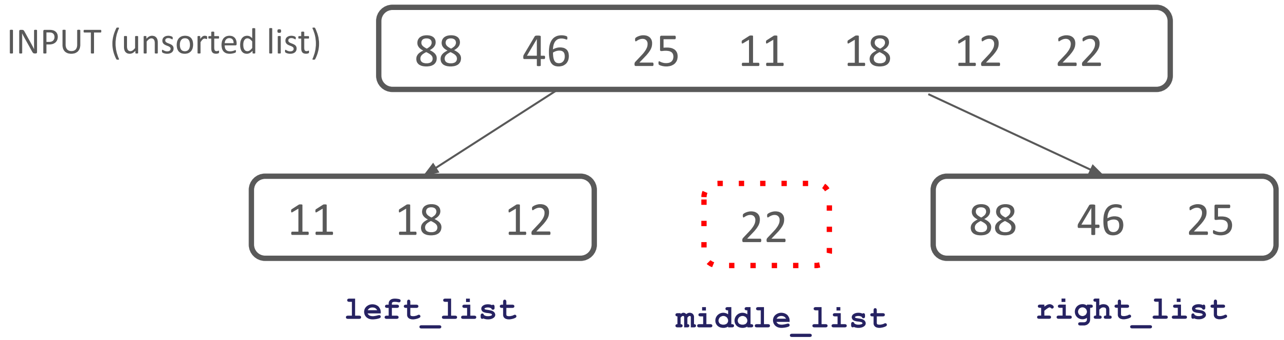
What features of recursion can you can see in this code?

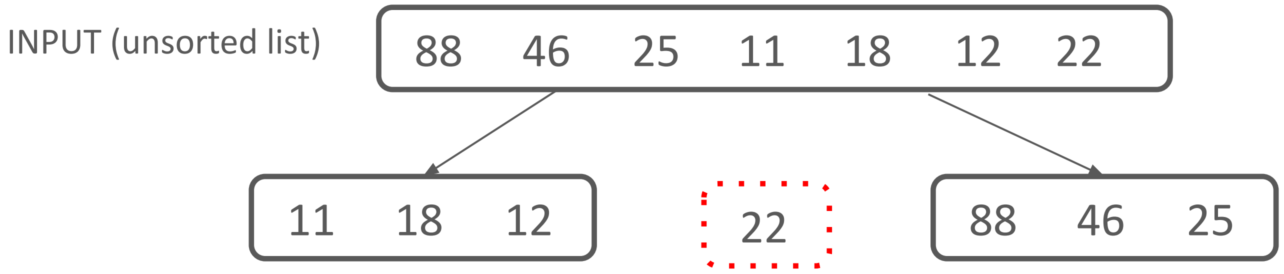
Example

88	46	25	11	18	12	22

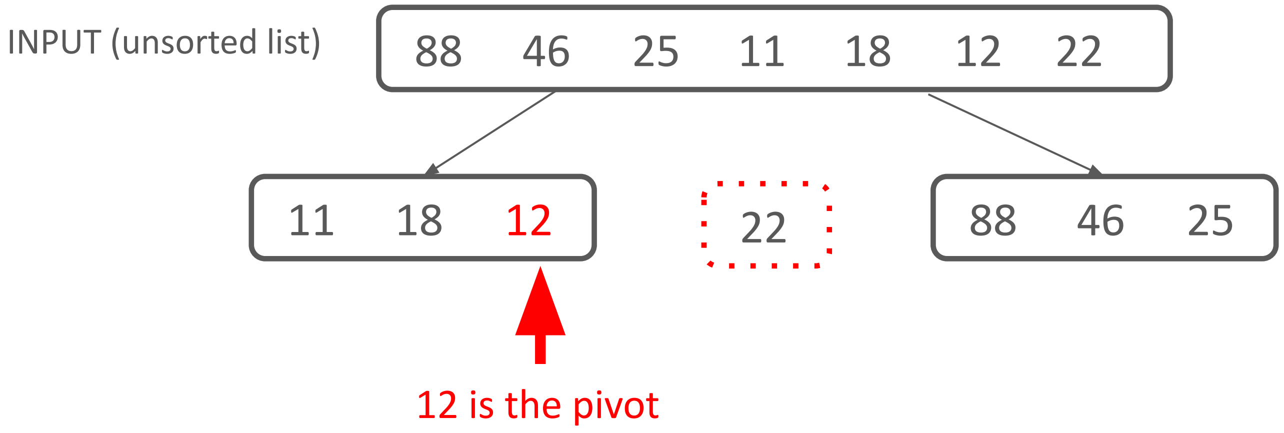
← 22 is the pivot

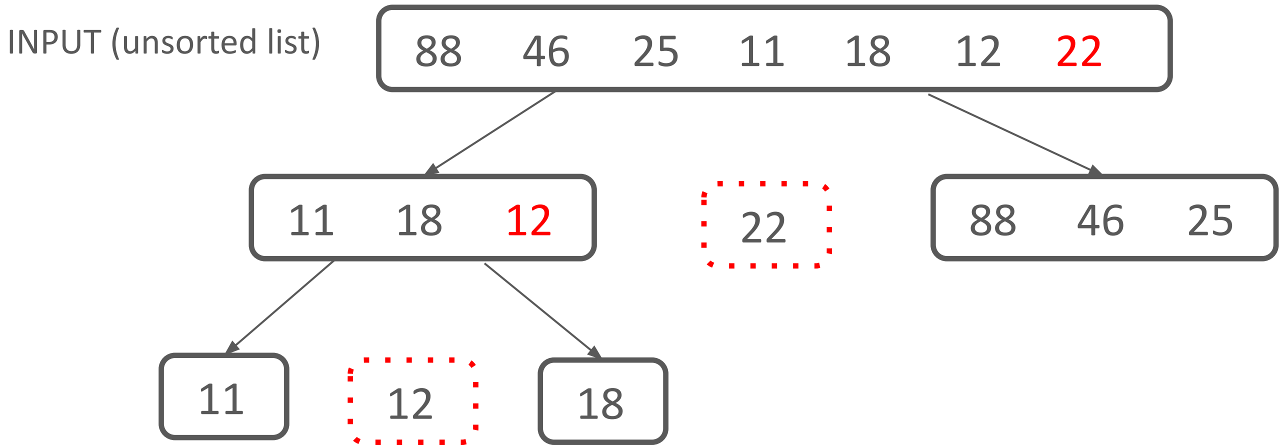
```
pivot = L[len(L)-1]
```

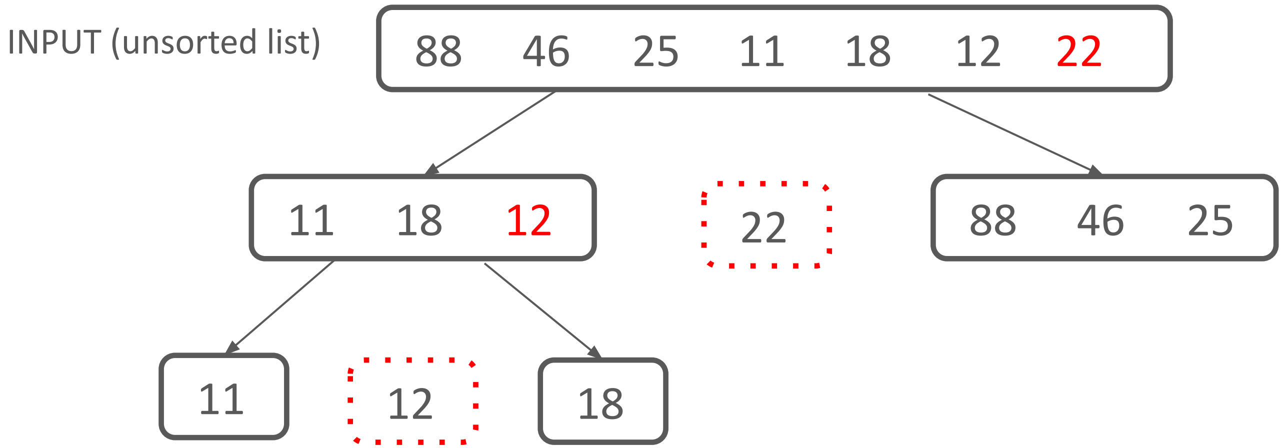

INPUT (unsorted list)
88 46 25 11 18 12 22
11 18 12
22
88 46 25
left_list
middle_list
right_list



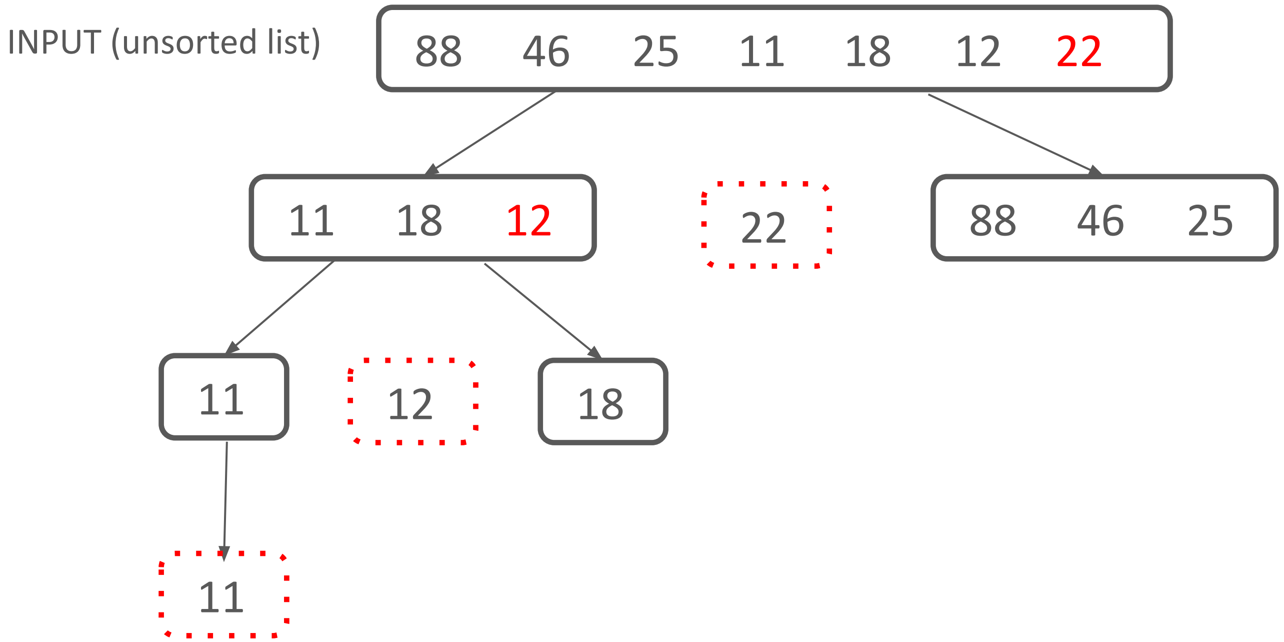
Now quicksort `left_list`


INPUT (unsorted list)
88 46 25 11 18 12 22
11 18 12
22
88 46 25
12 is the pivot


INPUT (unsorted list)
88 46 25 11 18 12 22
11 18 12
22
88 46 25
11
12
18

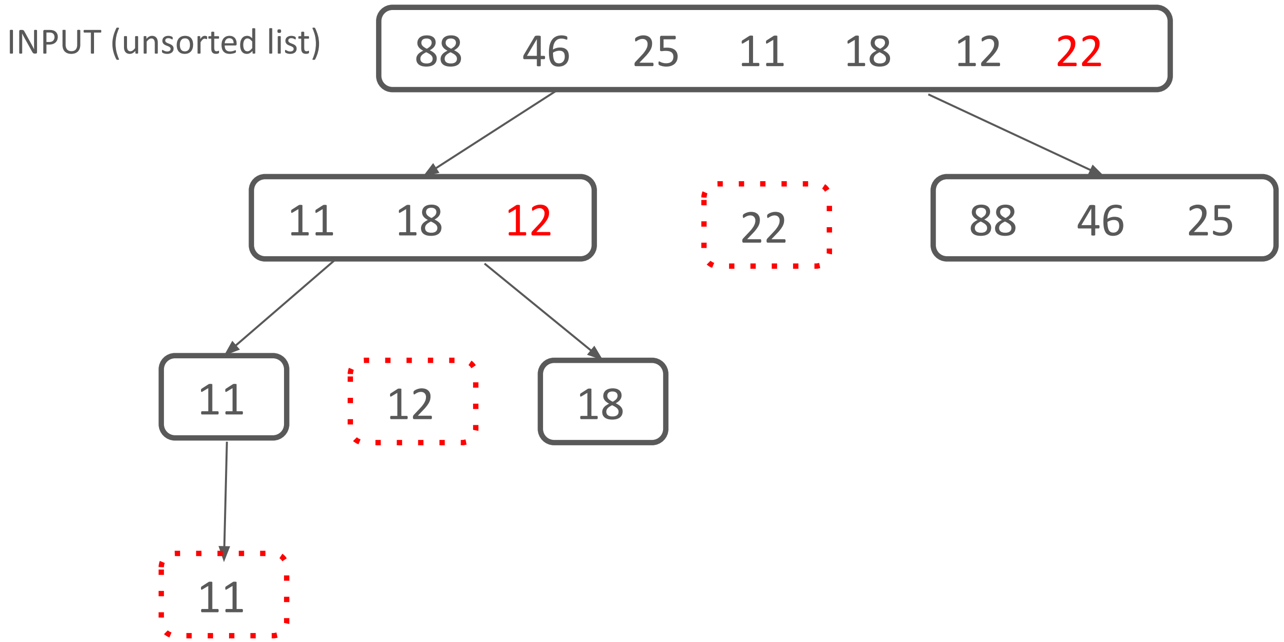


Now quicksort `left_list`

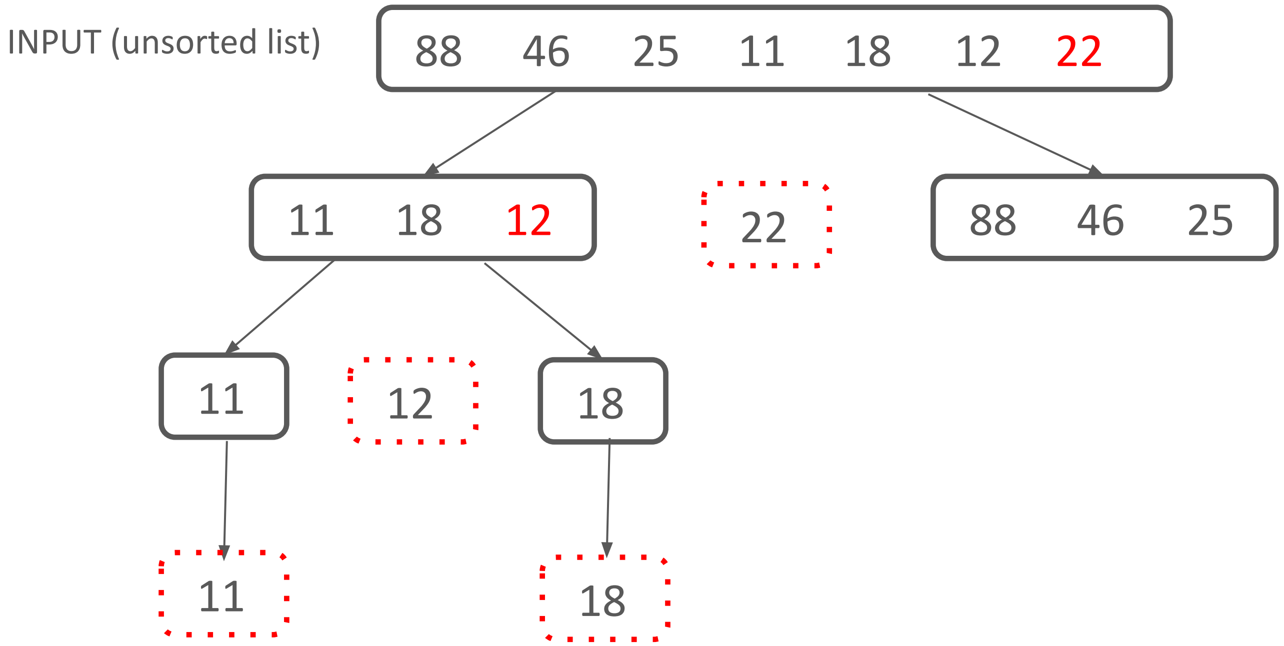


Now quicksort `left_list`

Base case: `len(L) <= 1` so return 11

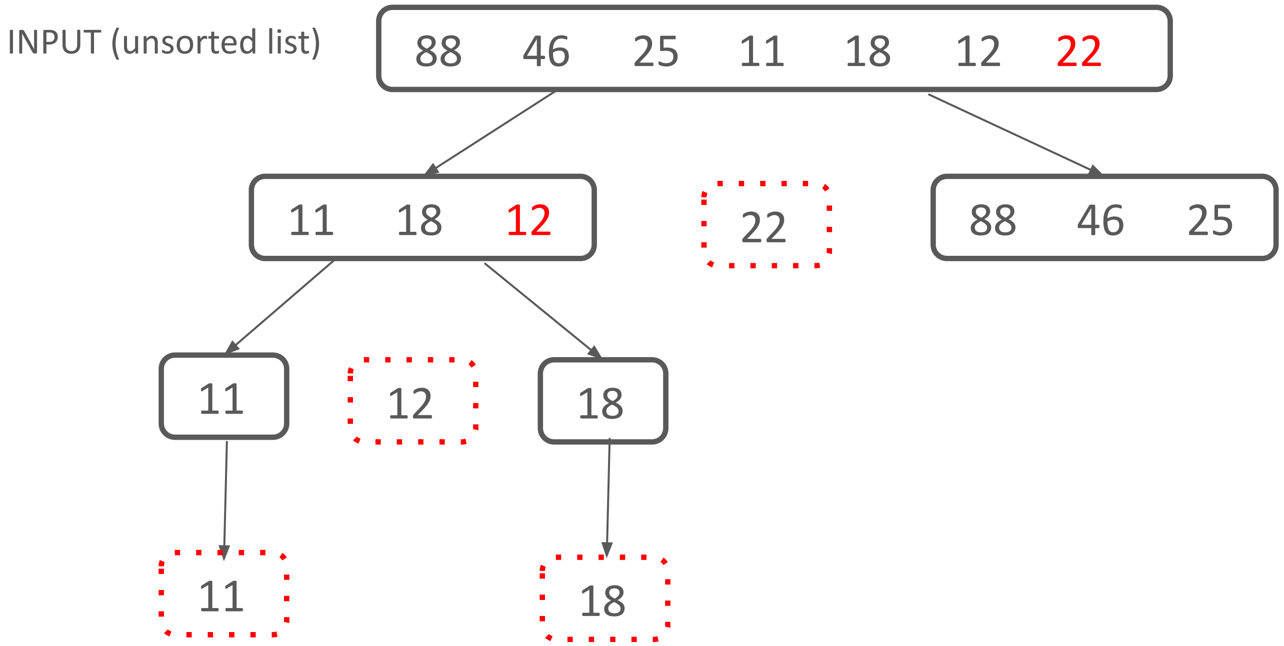


Now quicksort `right_list`

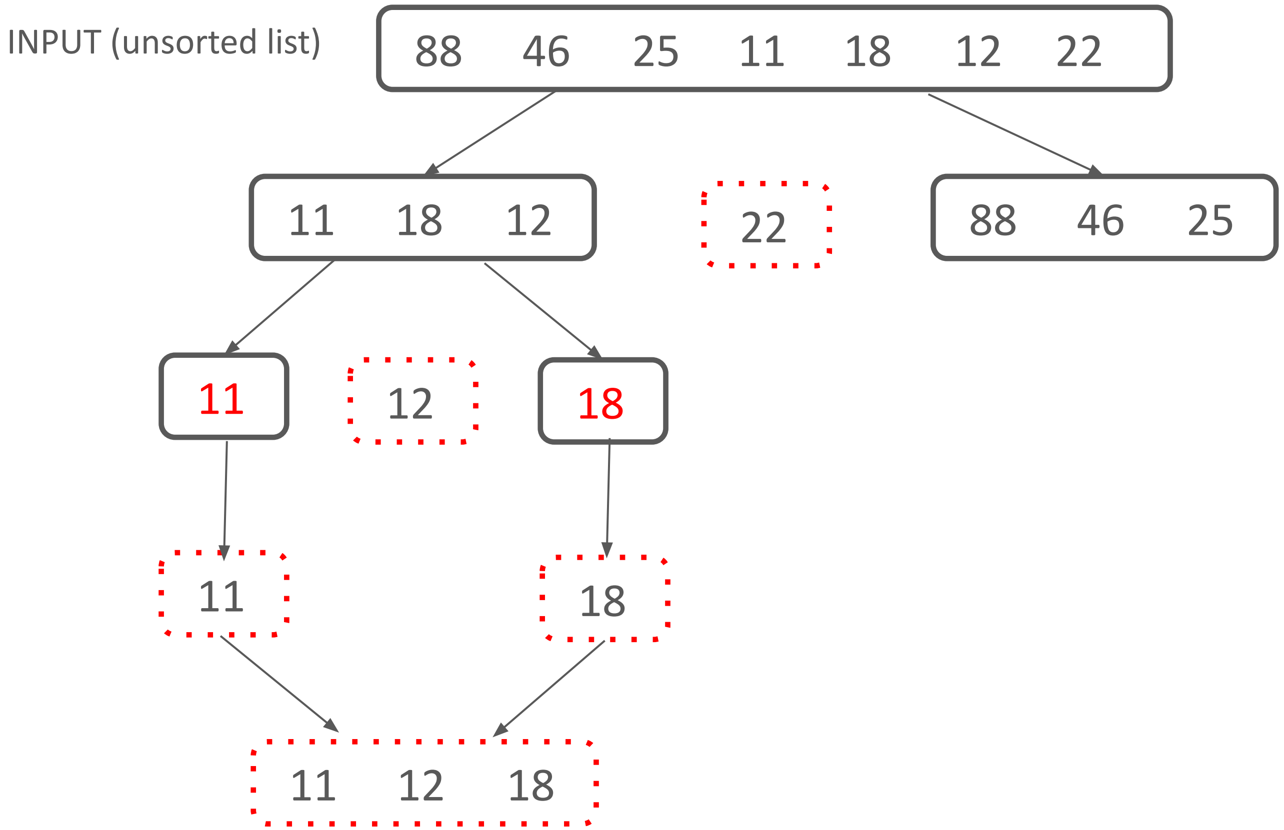


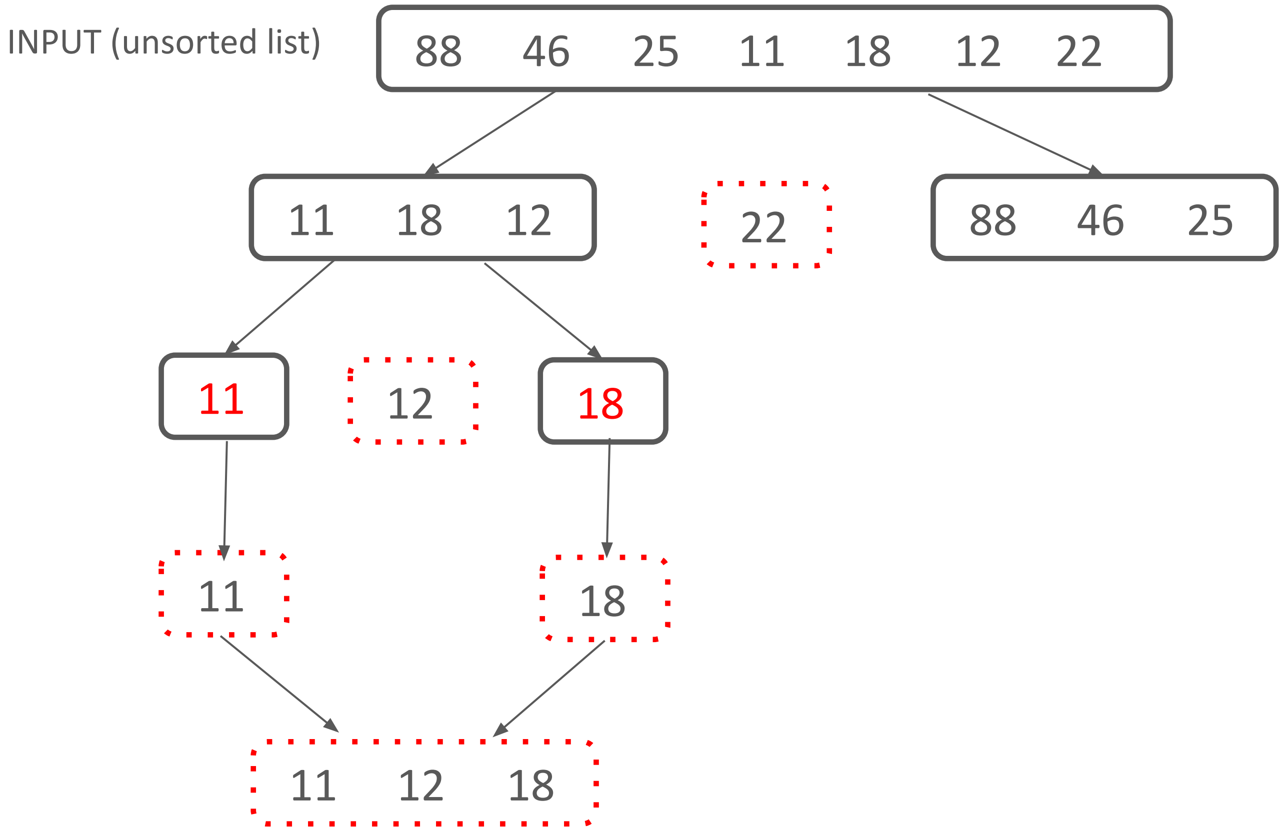
Now quicksort `right_list`

Base case: `len(L) <= 1` so return 18

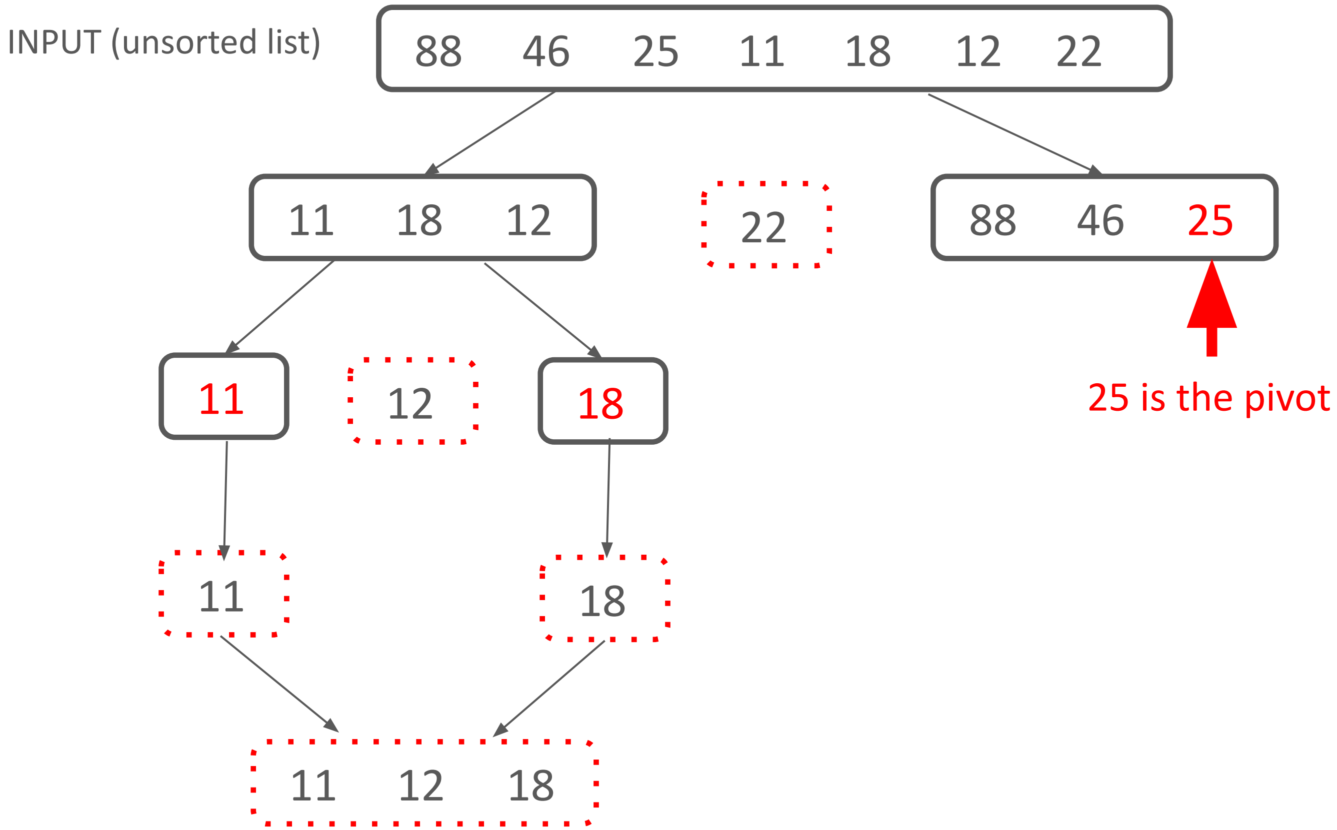


Result is `left + middle + right` so return 11 12 18

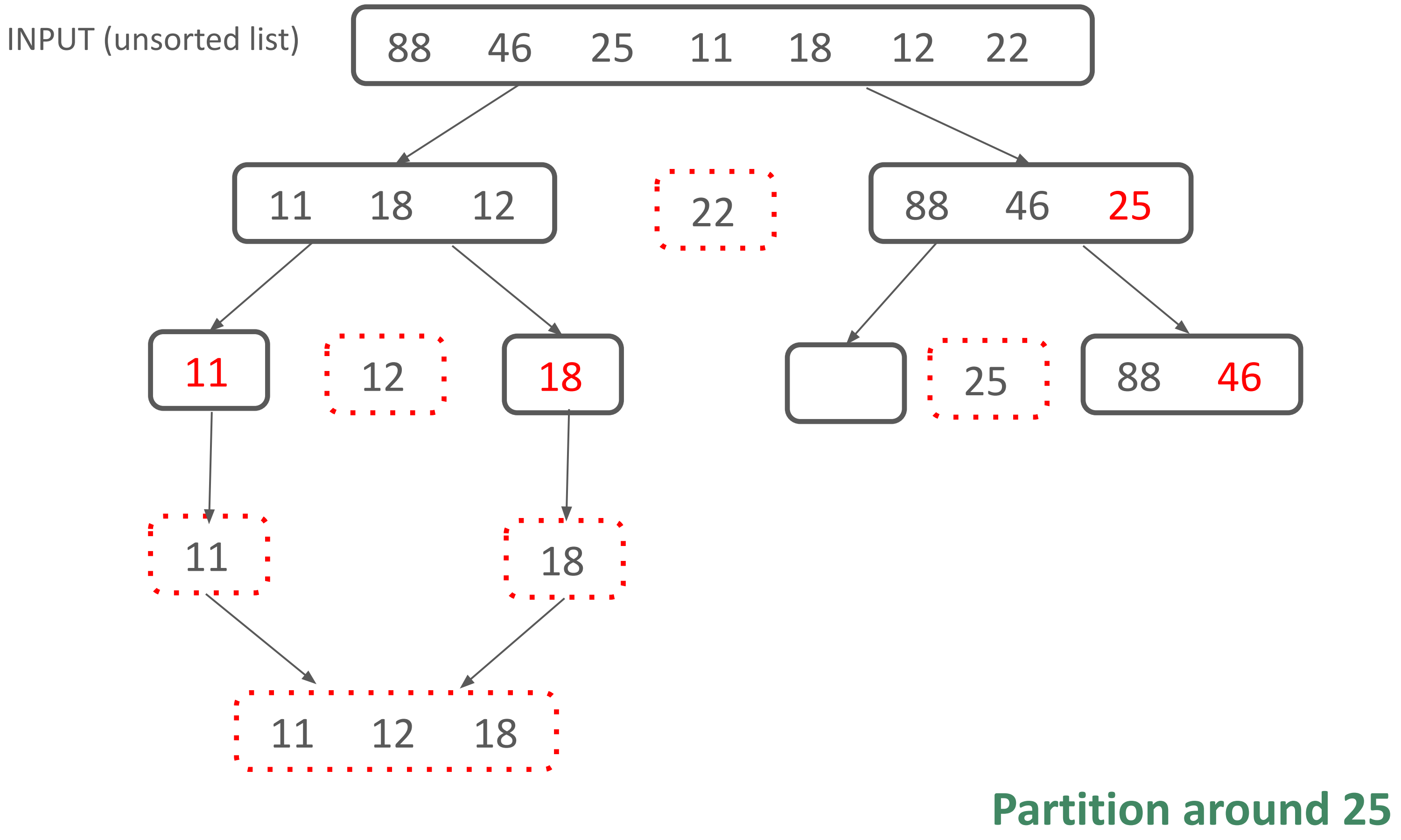

INPUT (unsorted list)
88 46 25 11 18 12 22
11 18 12
22
88 46 25
11
12
18
11
18
11 12 18

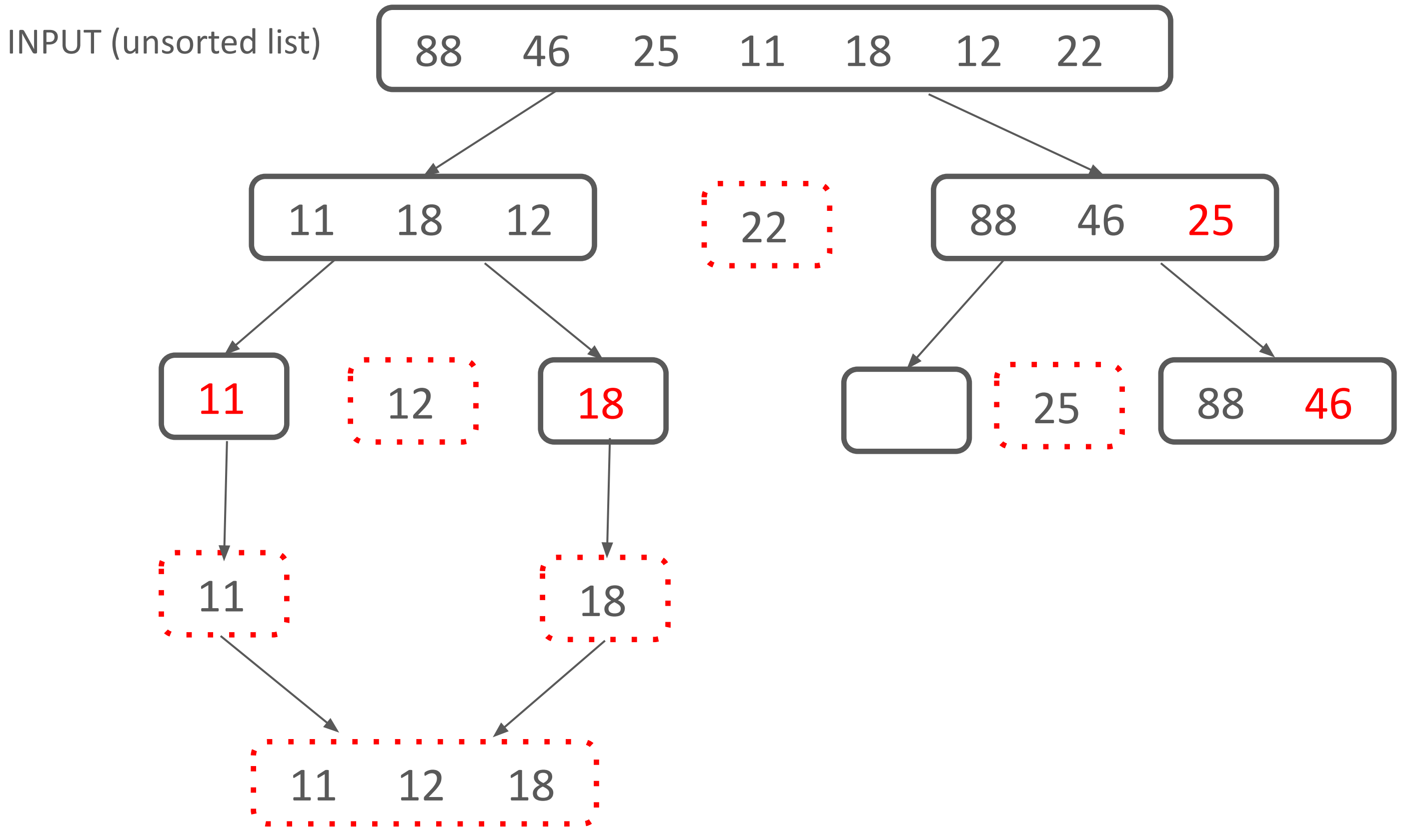


Now quicksort `right_list`

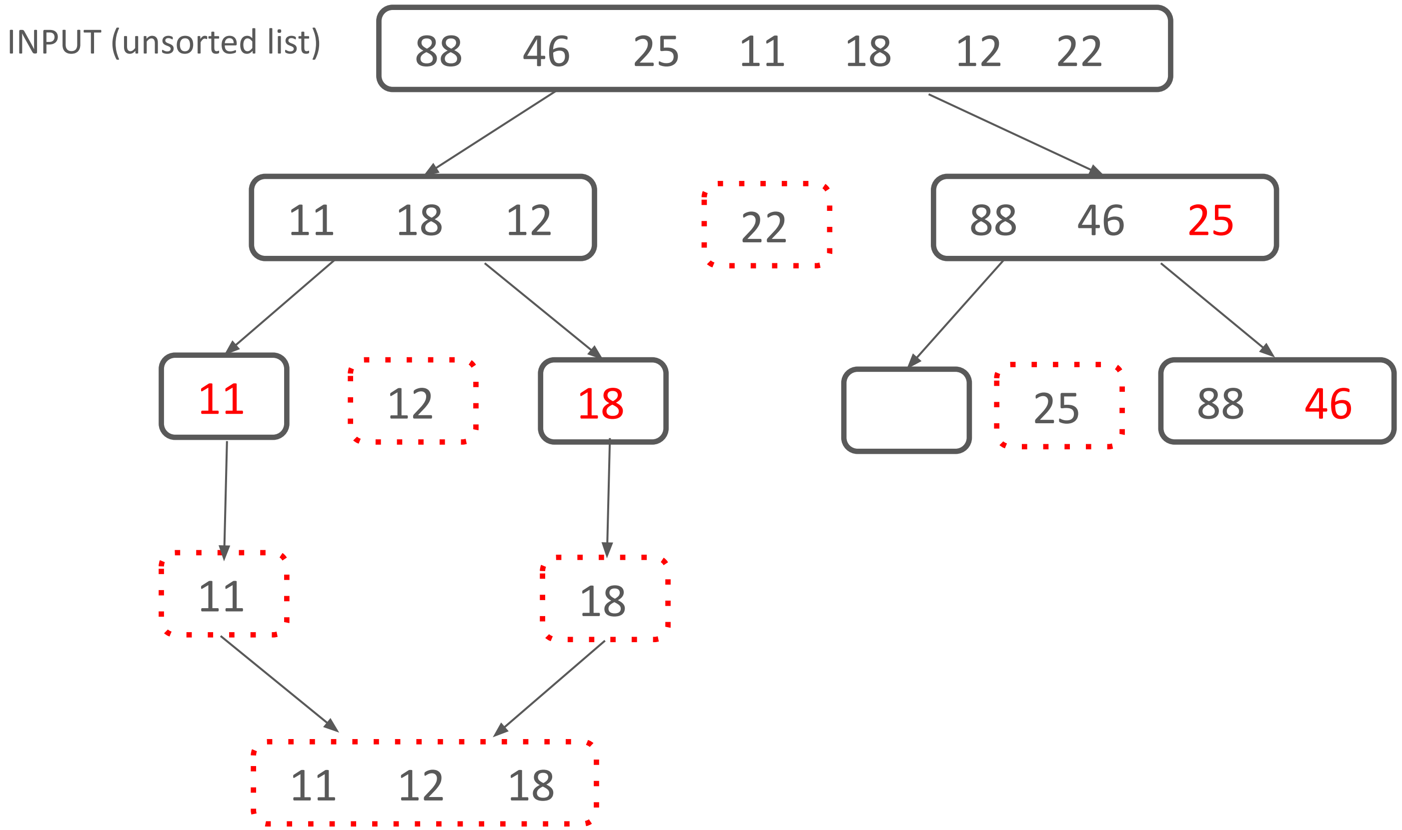


Now quicksort `right_list`

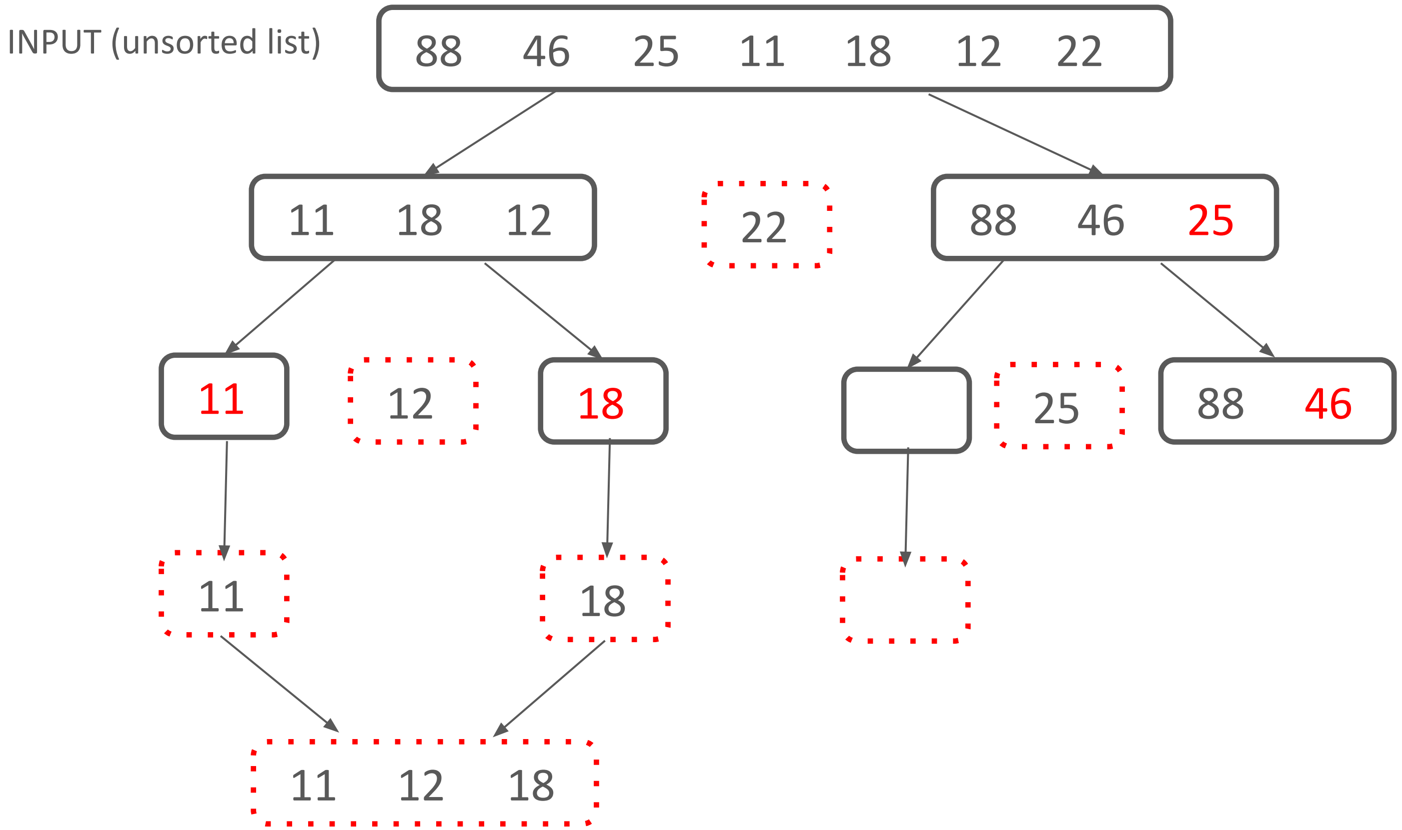

INPUT (unsorted list)
88 46 25 11 18 12 22
11 18 12
22
88 46 25
11
12
18
25
88 46
11
18
11 12 18
Partition around 25



Now quicksort `left_list`

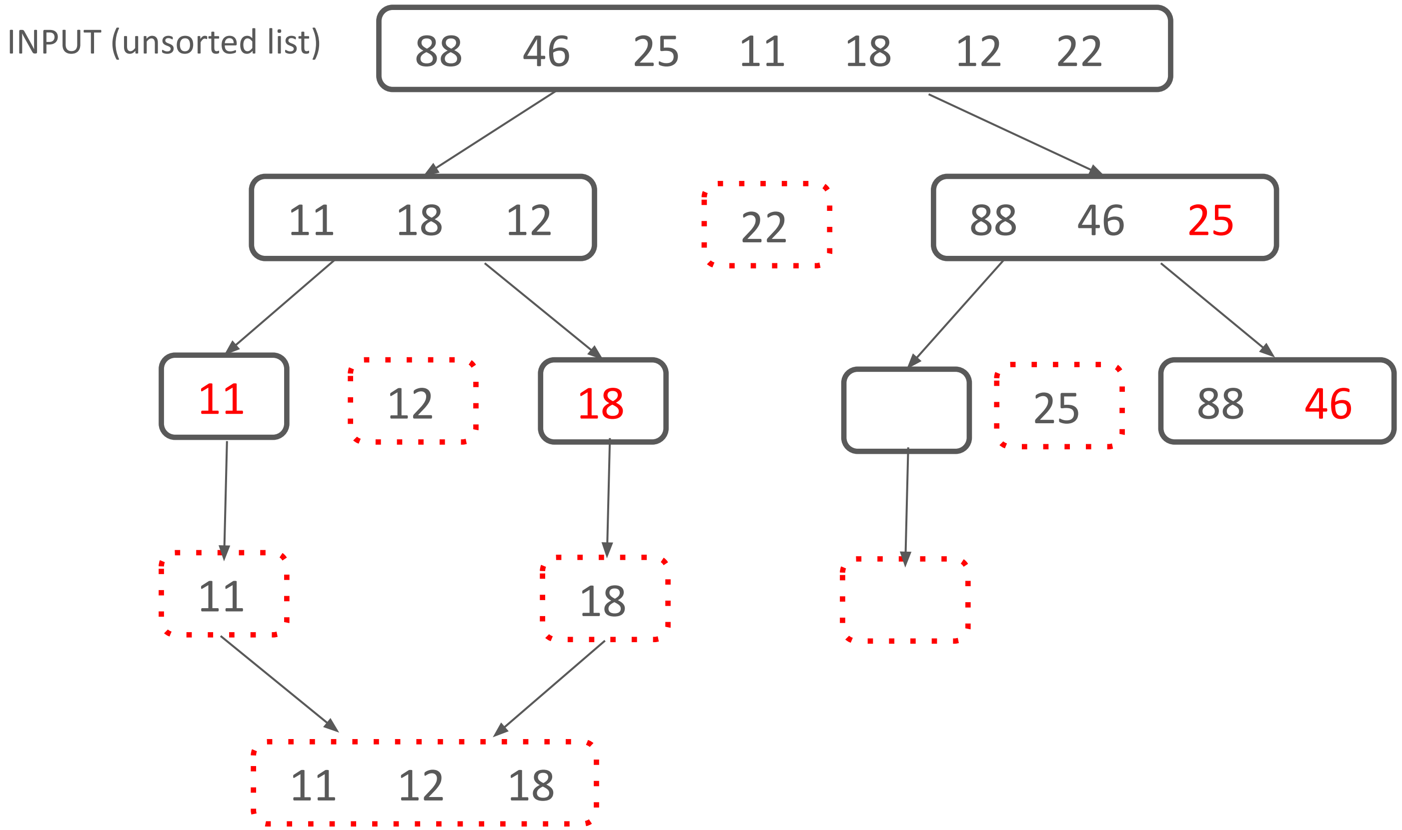


Now quicksort `left_list`

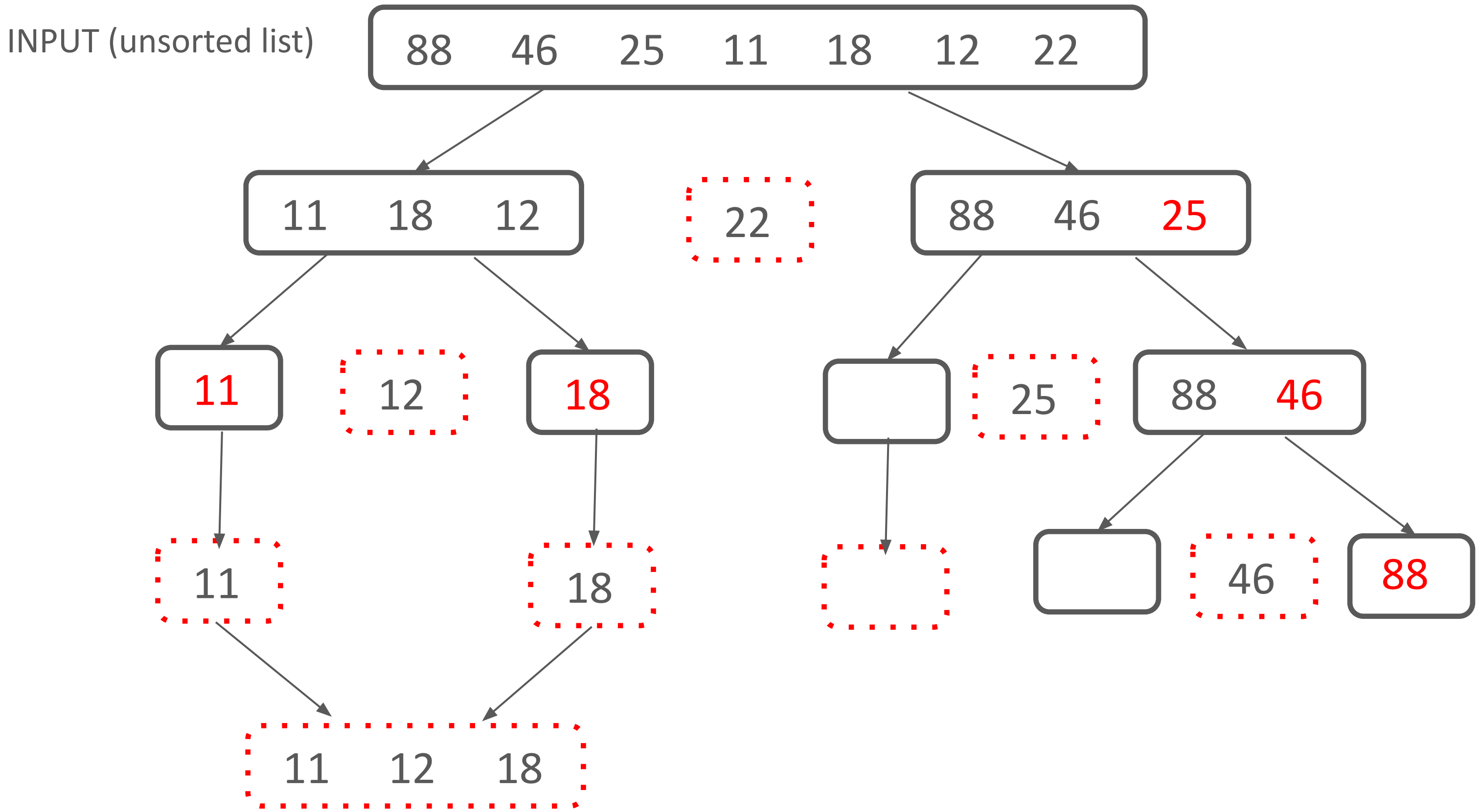


Now quicksort `left_list`

Base case: `len(L) <= 1` so return []

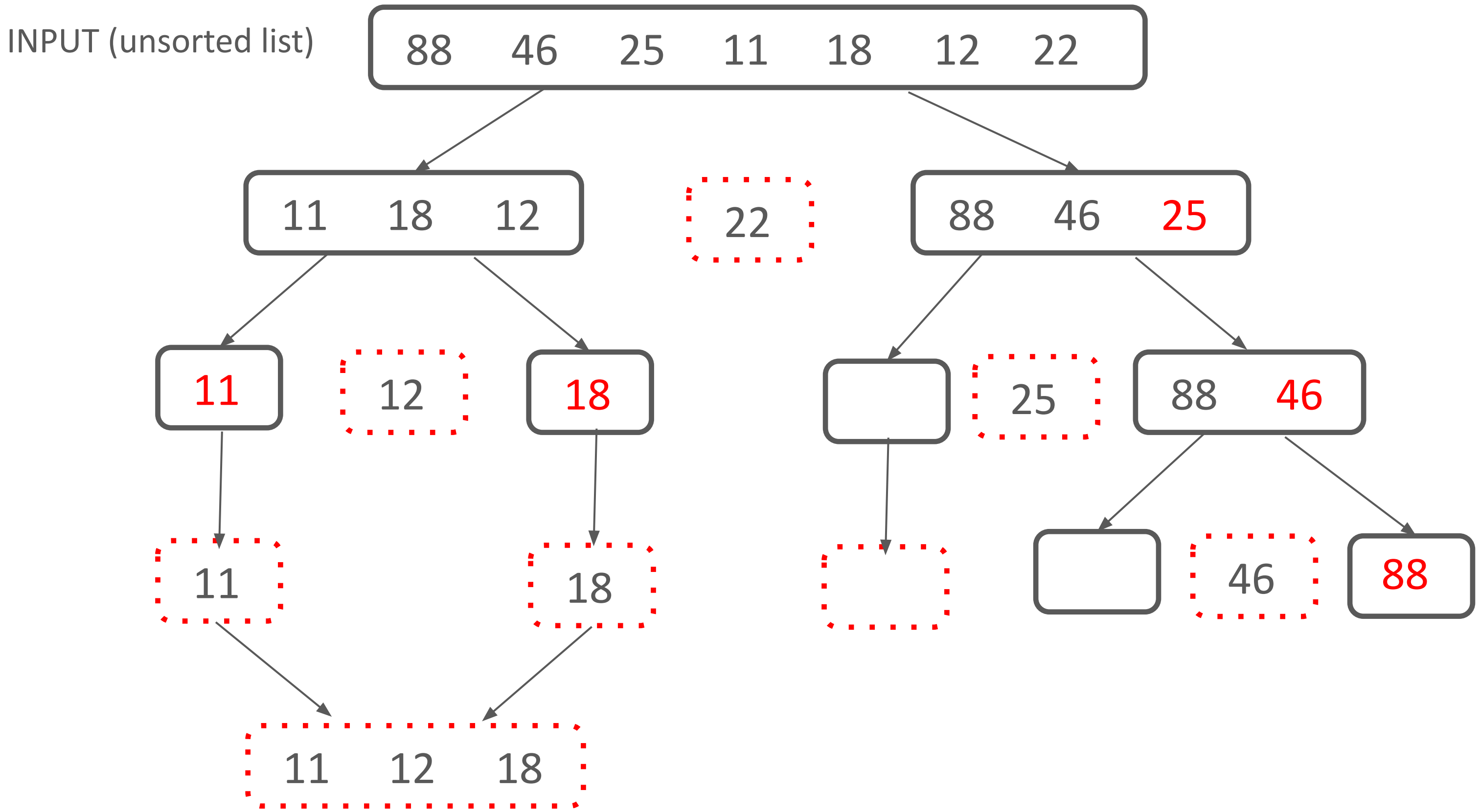


Now quicksort `right_list`



Now quicksort `right_list`

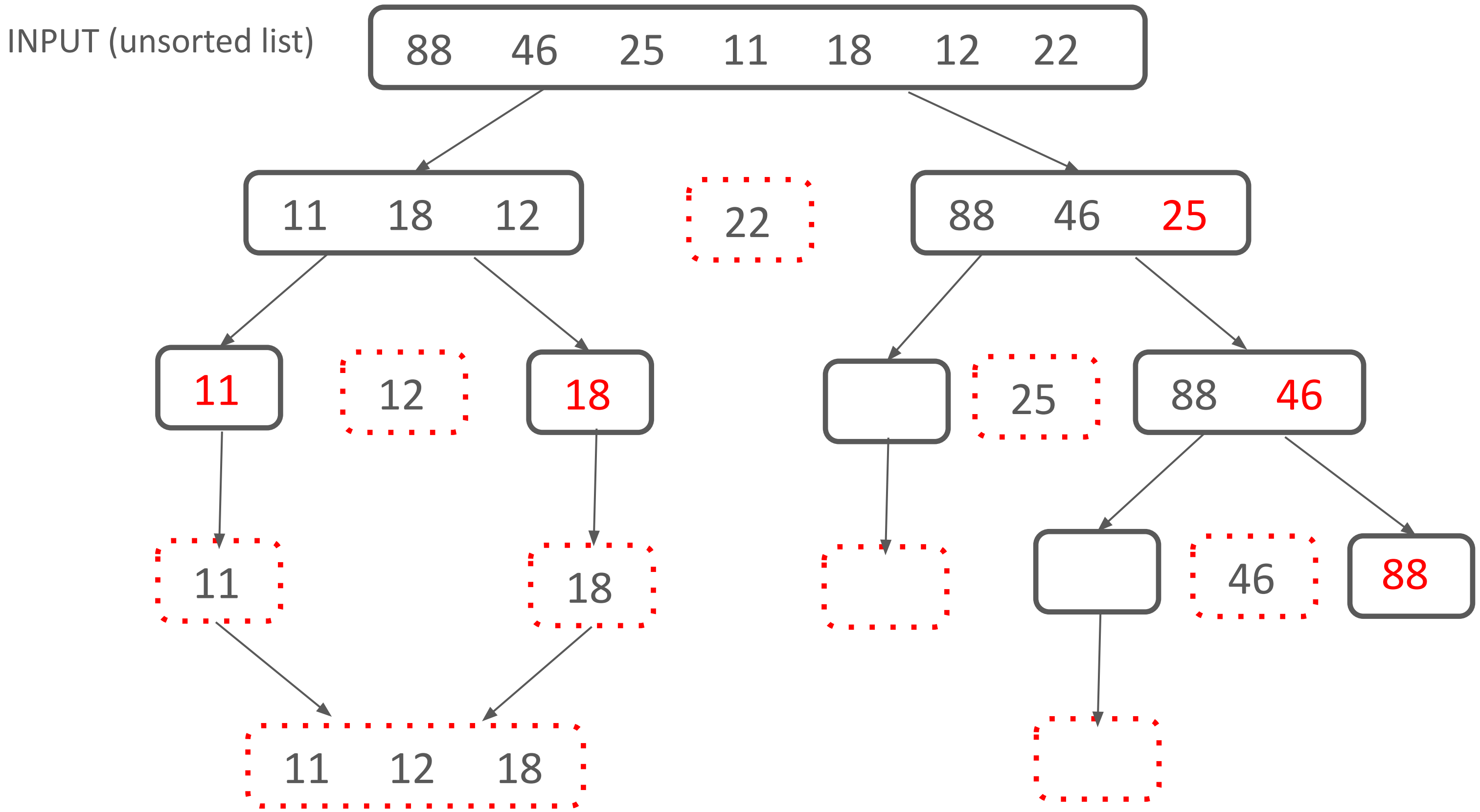
Partition around 46



Now quicksort `right_list`

Partition around 46

Now quicksort `left_list`

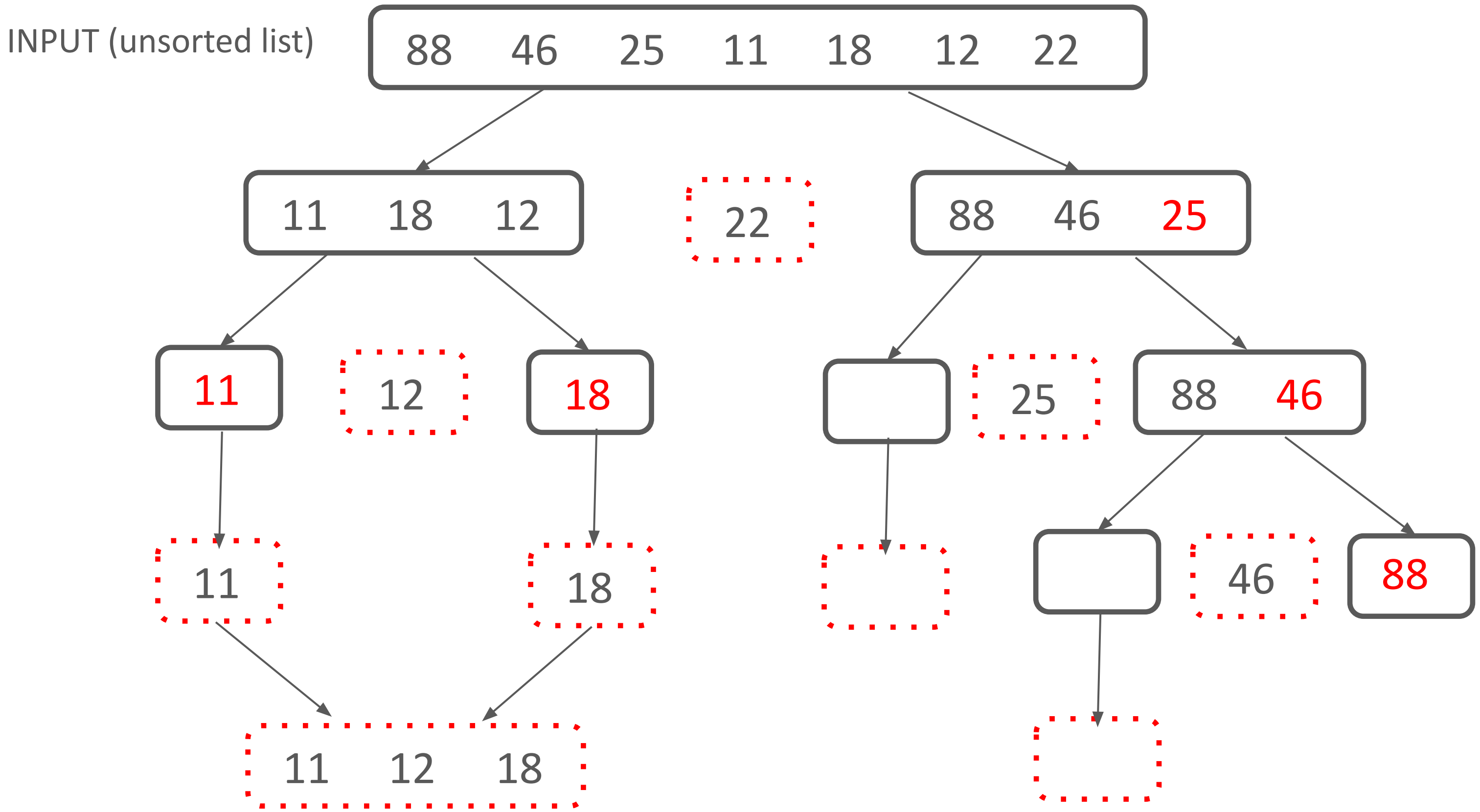


Now quicksort `right_list`

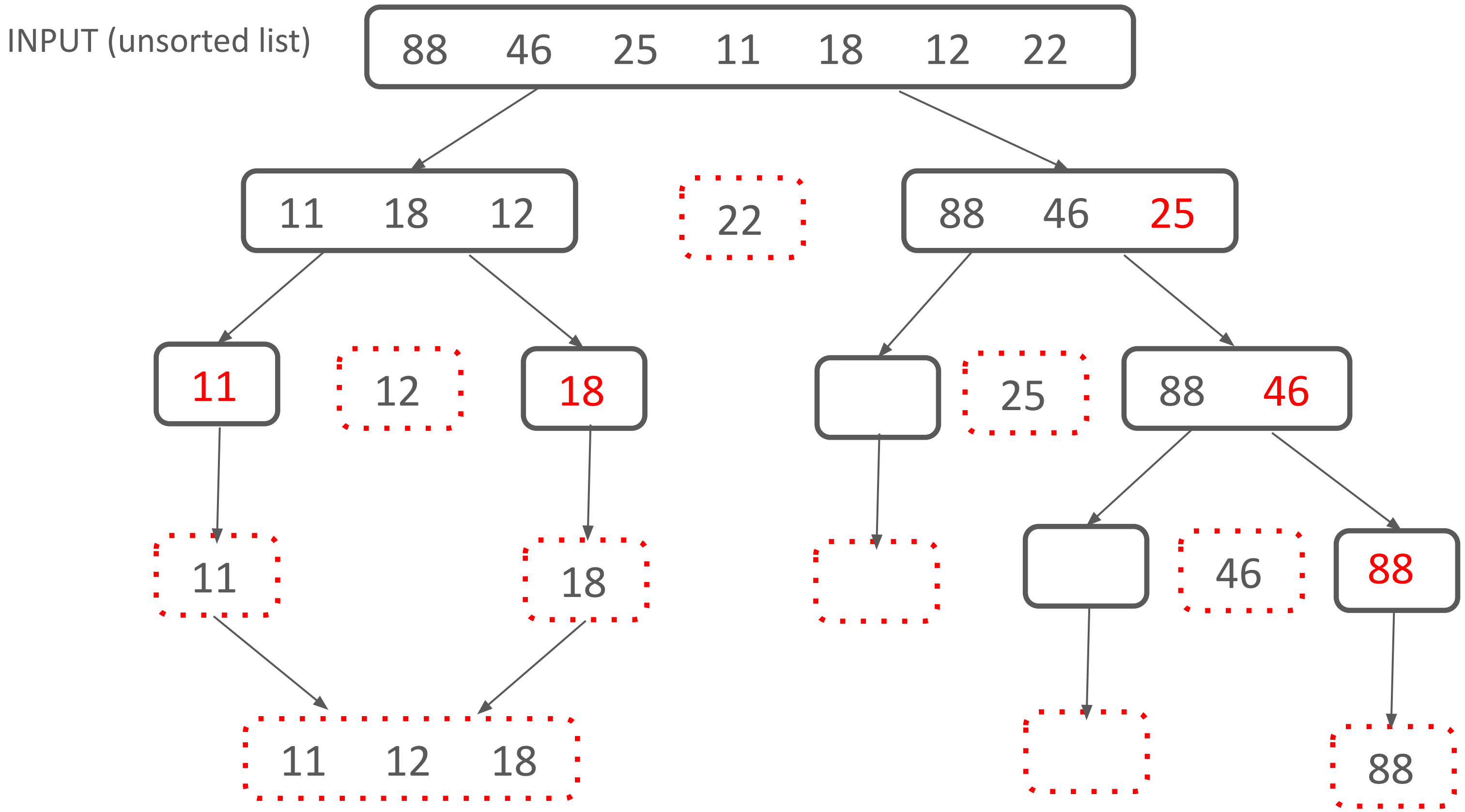
Partition around 46

Now quicksort `left_list`

Base case: `len(L) <= 1` so return []

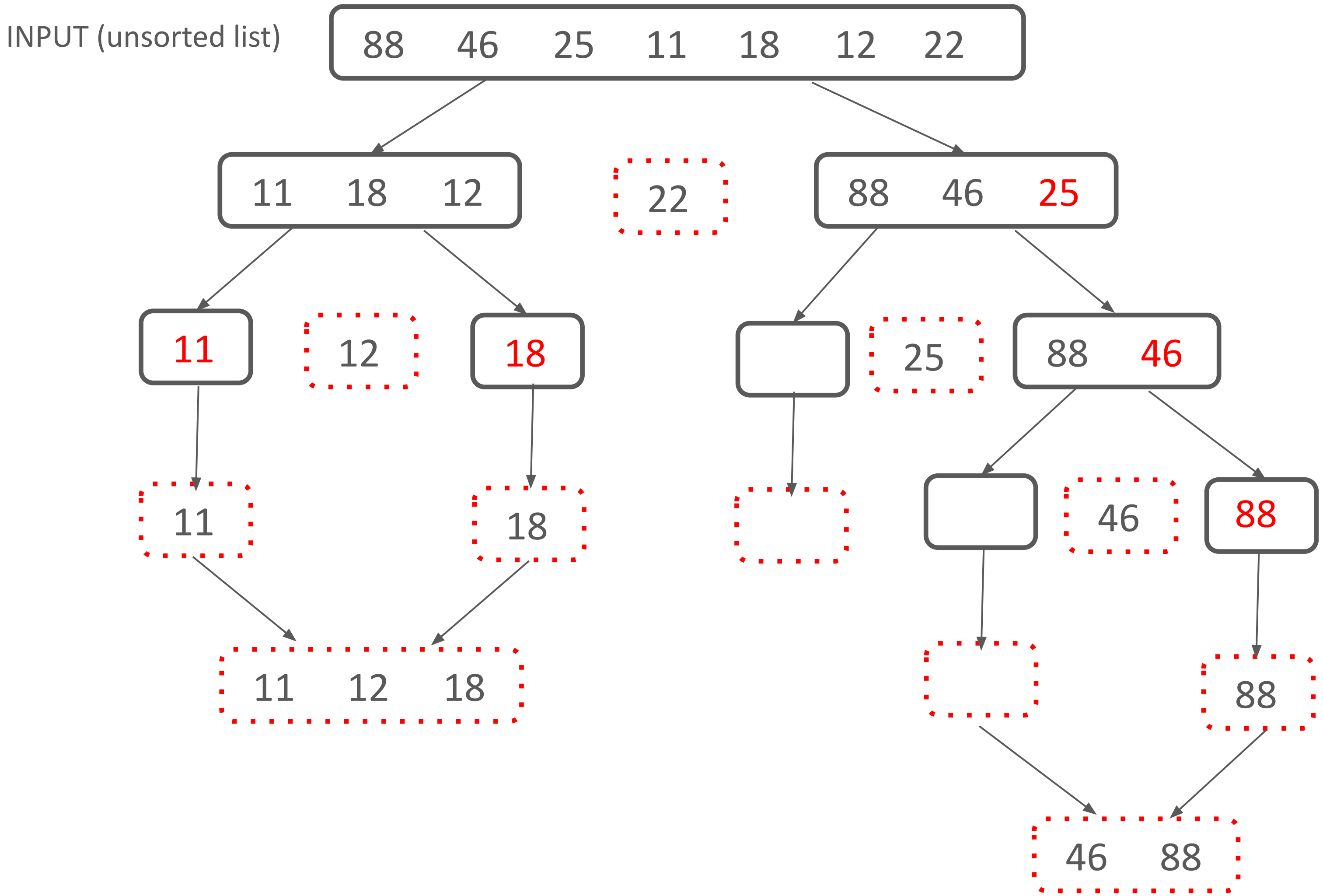


Now quicksort `right_list`

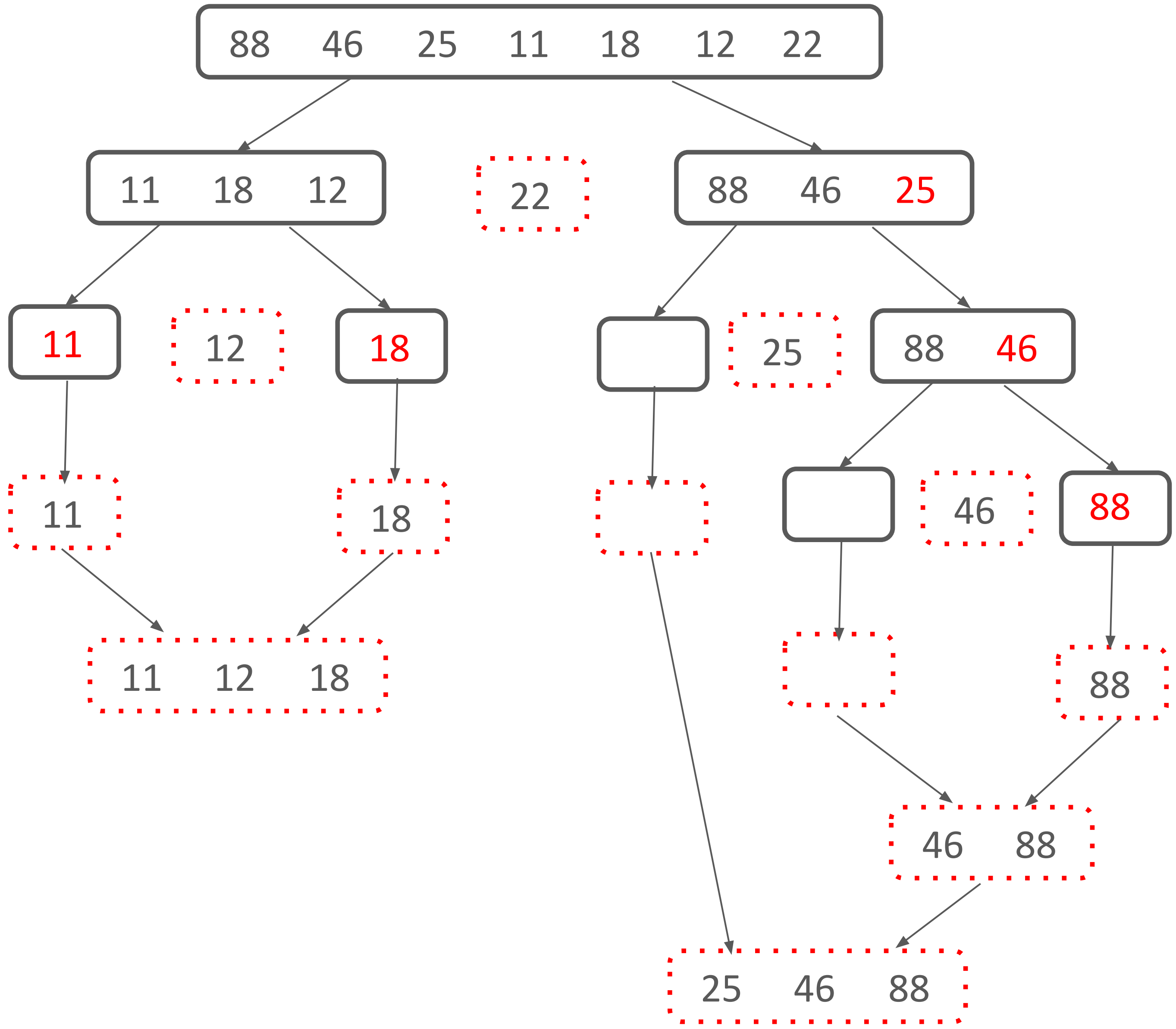


Now quicksort `right_list`

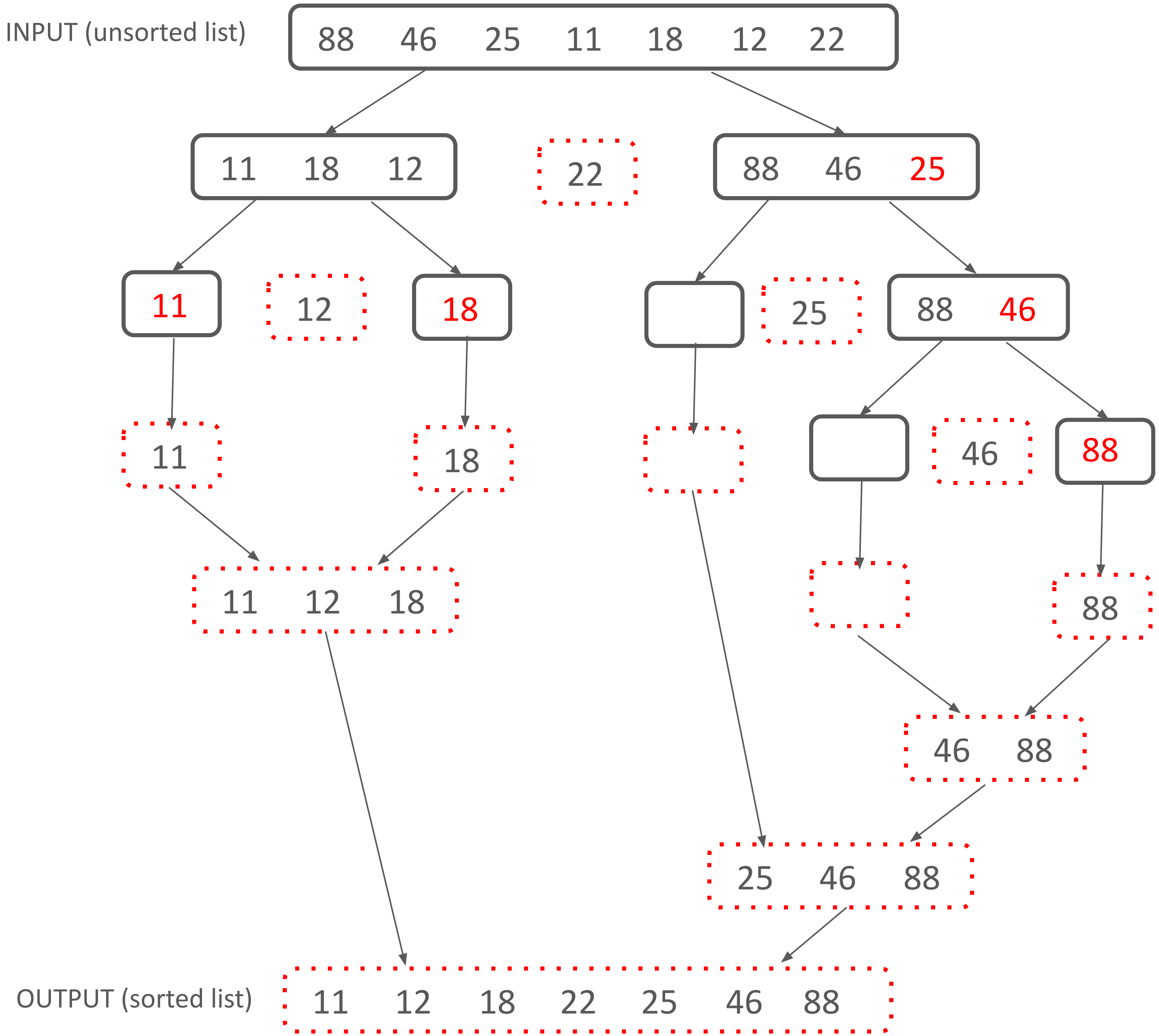
Base case: `len(L) <= 1` so return 88



Result is `left + middle + right` so return 46 88



Result is `left + middle + right` so return 25 46 88



INPUT (unsorted list)
88 46 25 11 18 12 22
11 18 12
22
88 46 25
11
12
18
25
88 46
11
18
46
88
11 12 18
88
46 88
25 46 88
OUTPUT (sorted list)
11 12 18 22 25 46 88

Perform a quicksort on the following:

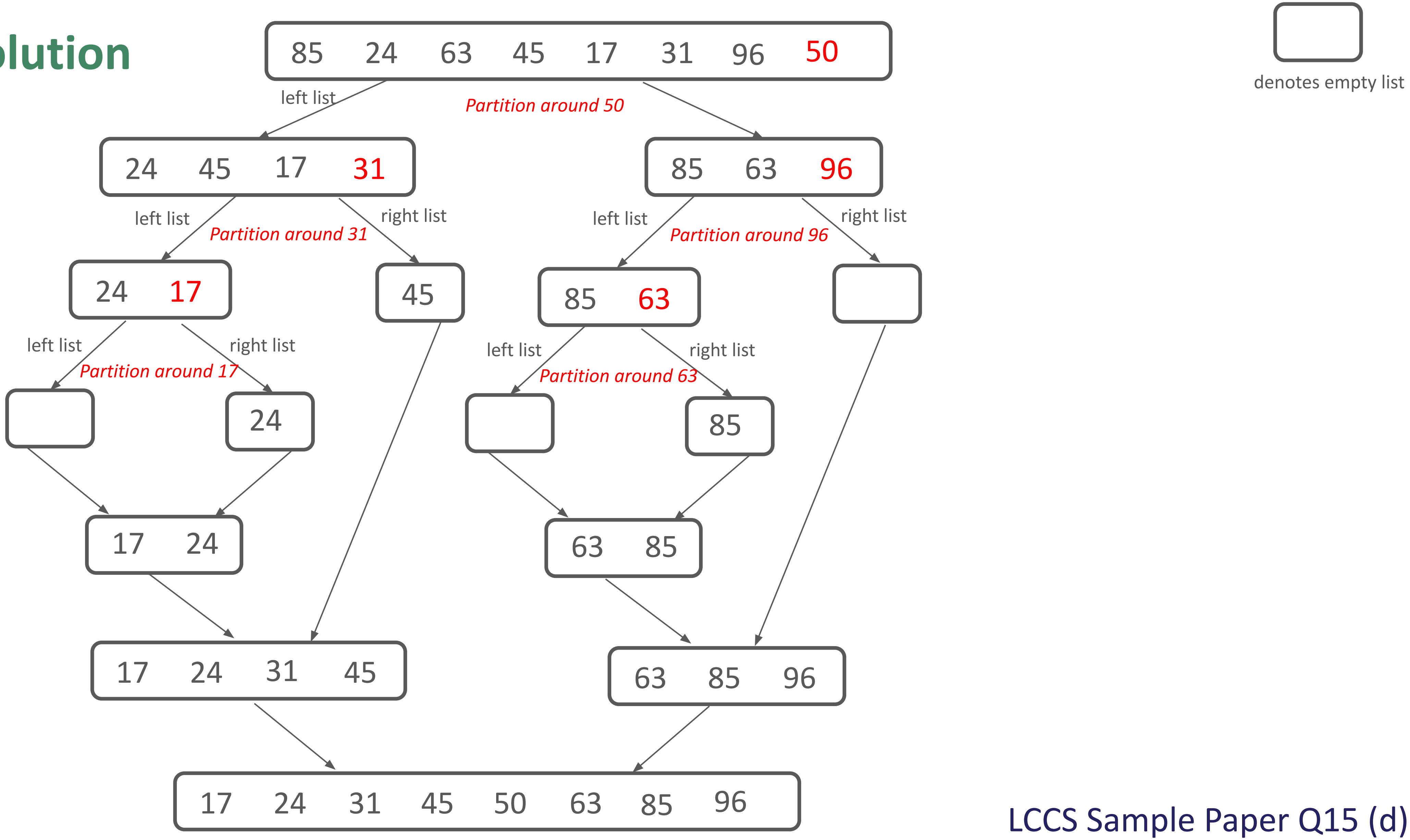
85	24	63	45	17	31	96	50

left list

Partition around 50

right list

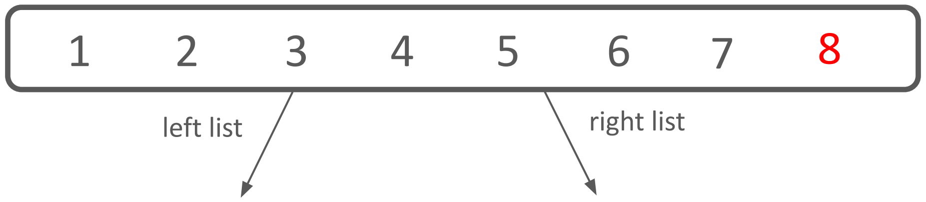
Sample Solution



LCCS Sample Paper Q15 (d)

Exercise

Investigate why this scenario leads to the worst case performance for the quicksort



Tea/Coffee

An Roinn Oideachais
Department of Education