Types of Algorithms and algorithm examples – illustrated

https://www.lavivienpost.com/algorithms-types-and-algorithm-examples/

An algorithm is a set of rules that instruct the computer how to perform a task. This post lists the types of algorithms and their examples, such as Binary search, sorting, Divide and conquer, Two pointers, Greedy, Recursion, Backtracking, and Dynamic programming. The algorithms illustrated provide a glimpse of algorithms in different types.

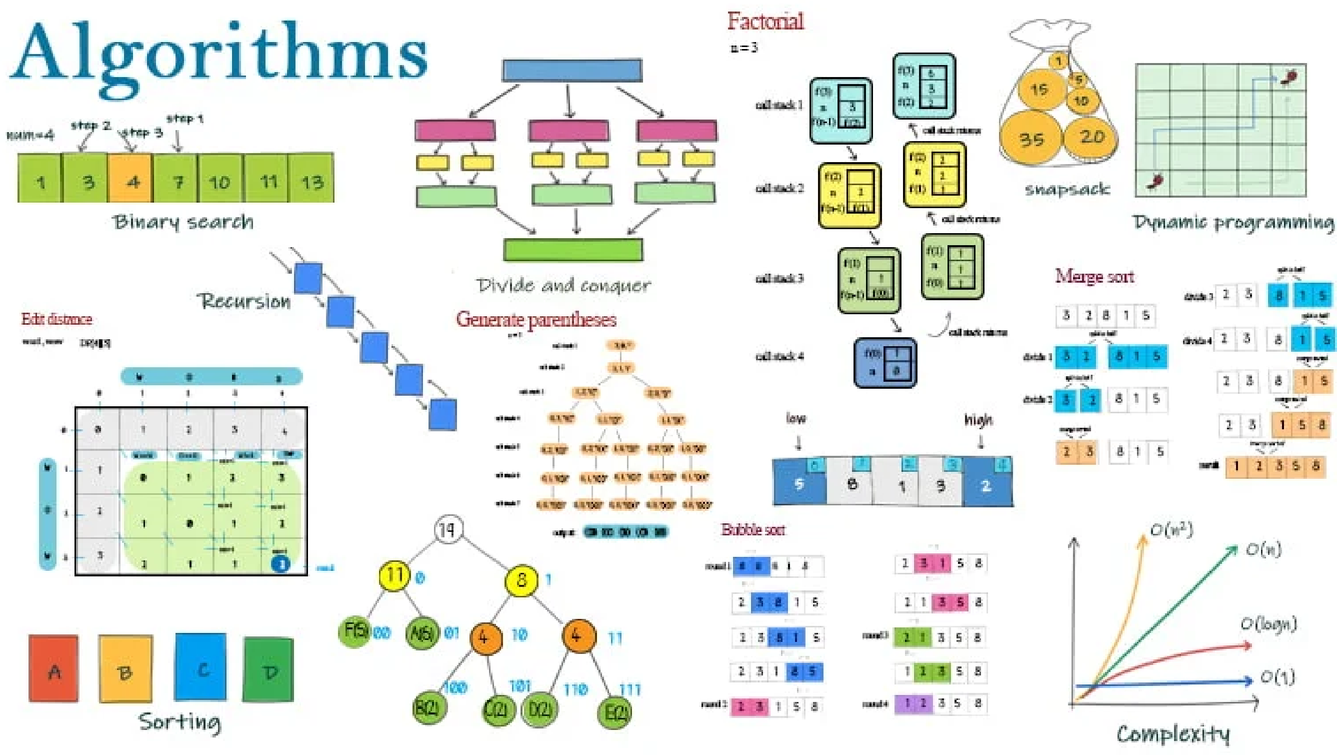
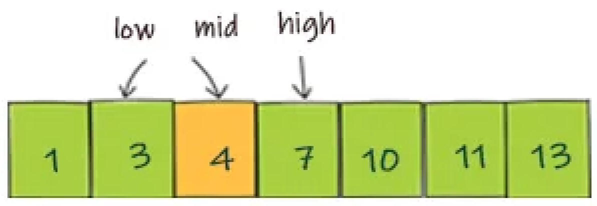
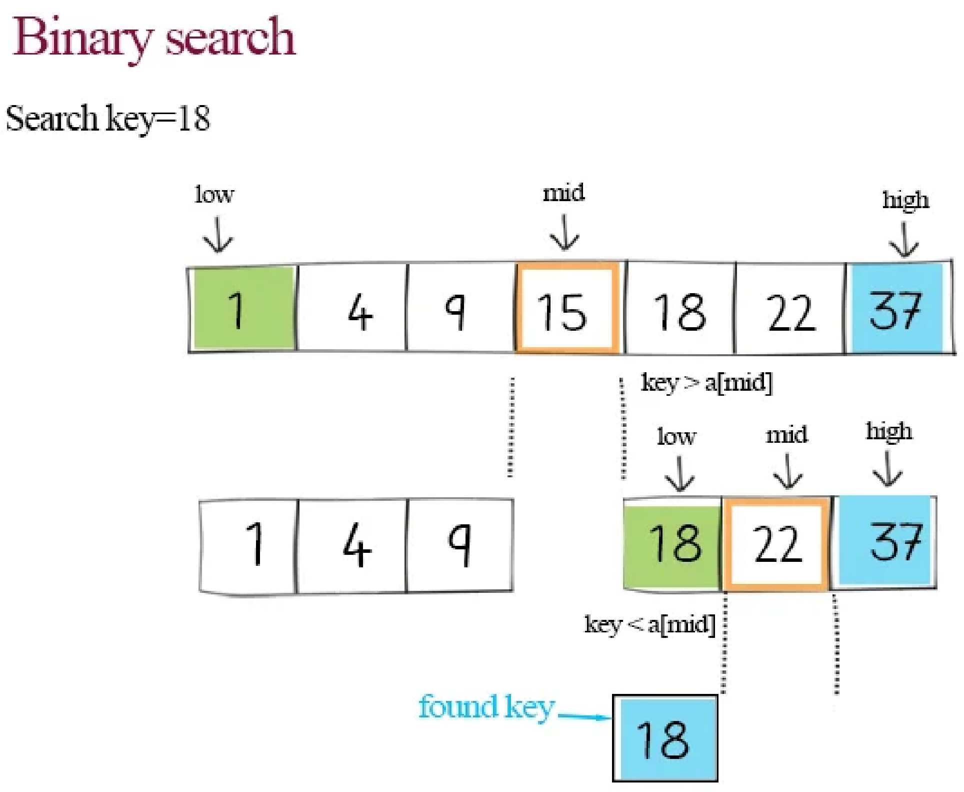


Table of Content

1. Algorithm examples – Binary search

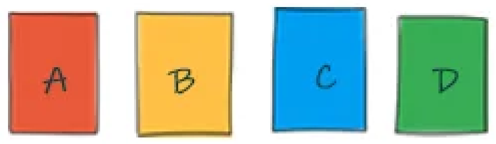


Binary search is an efficient algorithm for finding an item from an ordered list of items. It works by repeatedly dividing in half the portion of the list, until narrowing down the possible locations to just one. The time complexity reduces from O(n) to O(logn).



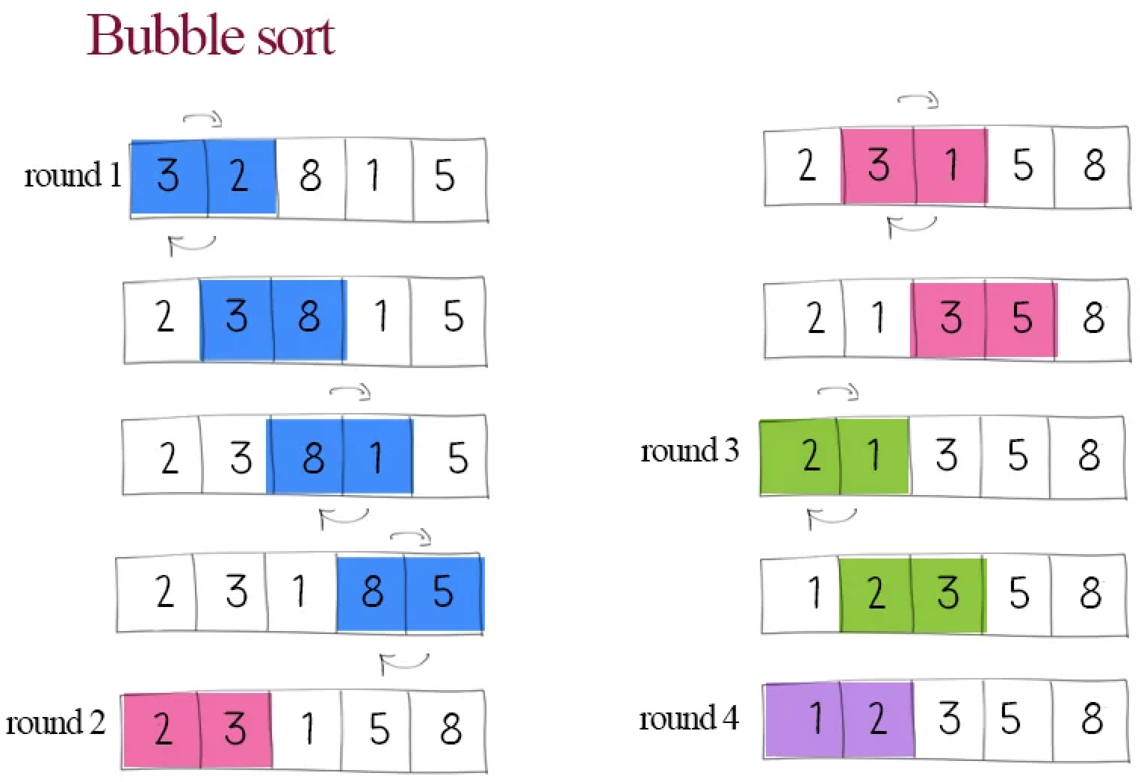
Binary search in Java, JavaScript, Python and Doodle

2. Algorithm examples – Simple sorting



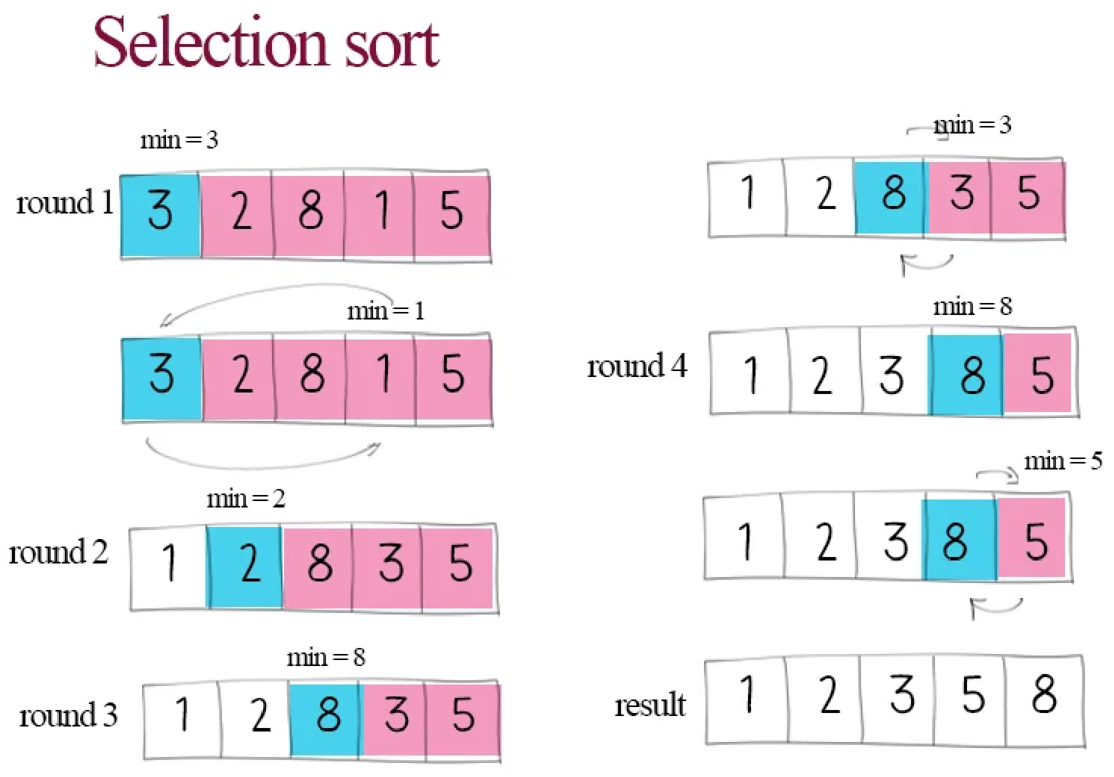
Sorting is probably one of the most studied algorithm examples. It is a process that takes an array or strings as input, performs specified operations, and outputs a sorted order of arrays or strings. Simple sorting algorithms use two nested loops to compare two elements and change position if they are not ordered. They are not efficient as the time complexity is O(n^2).

Example 1: **Bubble sort **– compares adjacent elements and swaps them if they are in the wrong order.



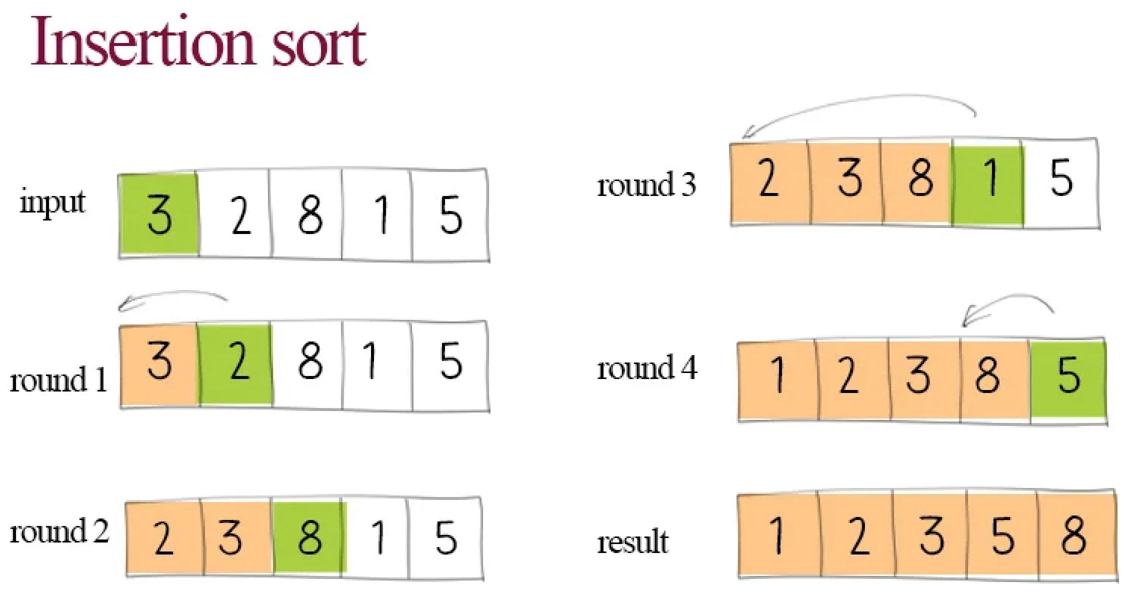
Bubble sort in Java, JavaScript, Python and Doodle

Example 2: **Selection sort** – repeatedly finds the minimum element from the unsorted part and puts it at the beginning.



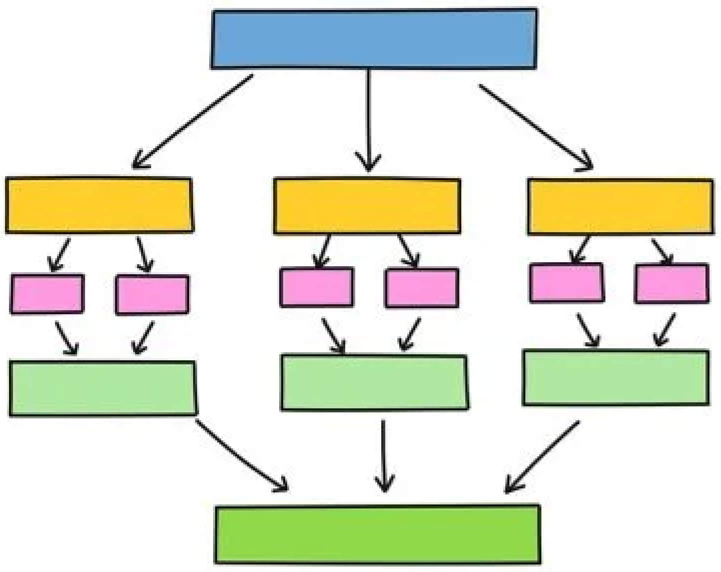
Selection sort in Java, JavaScript, Python and Doodle

Example 3: **Insertion sort** – repeatedly takes an element from the input data and inserts it into the position so that its value is between the previous and the next element.



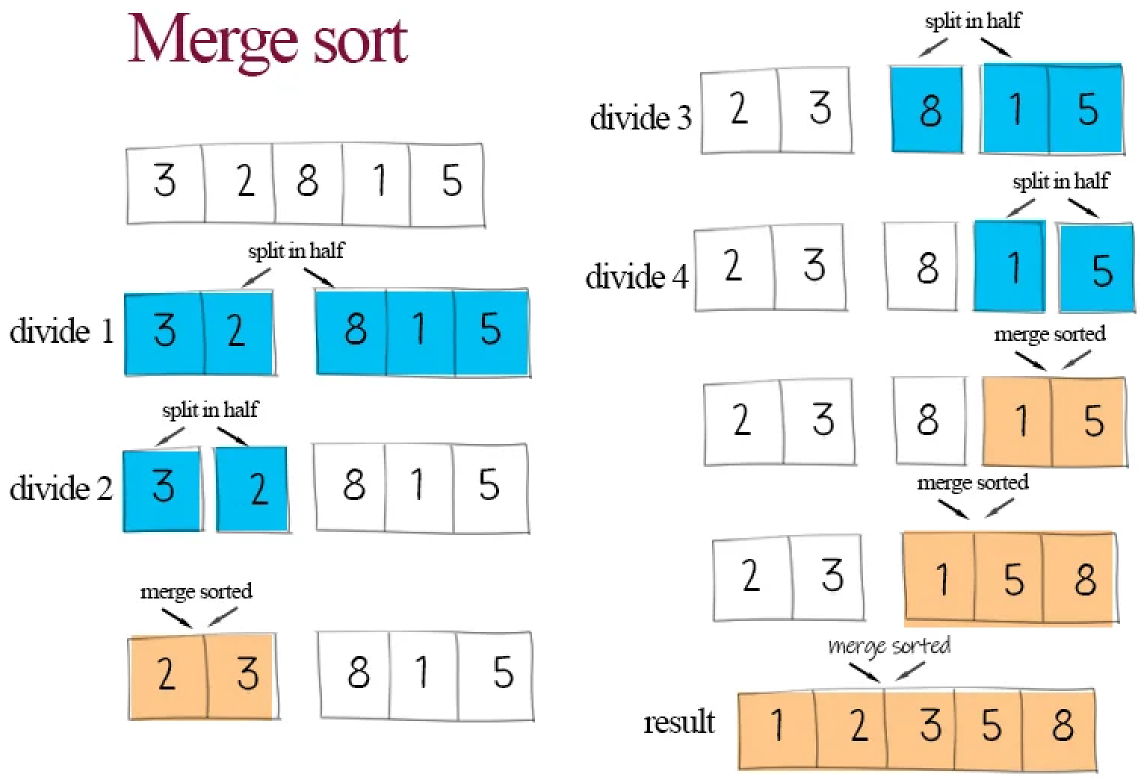
Insertion sort in Java, JavaScript, Python and Doodle

3. Algorithm examples – Divide and conquer



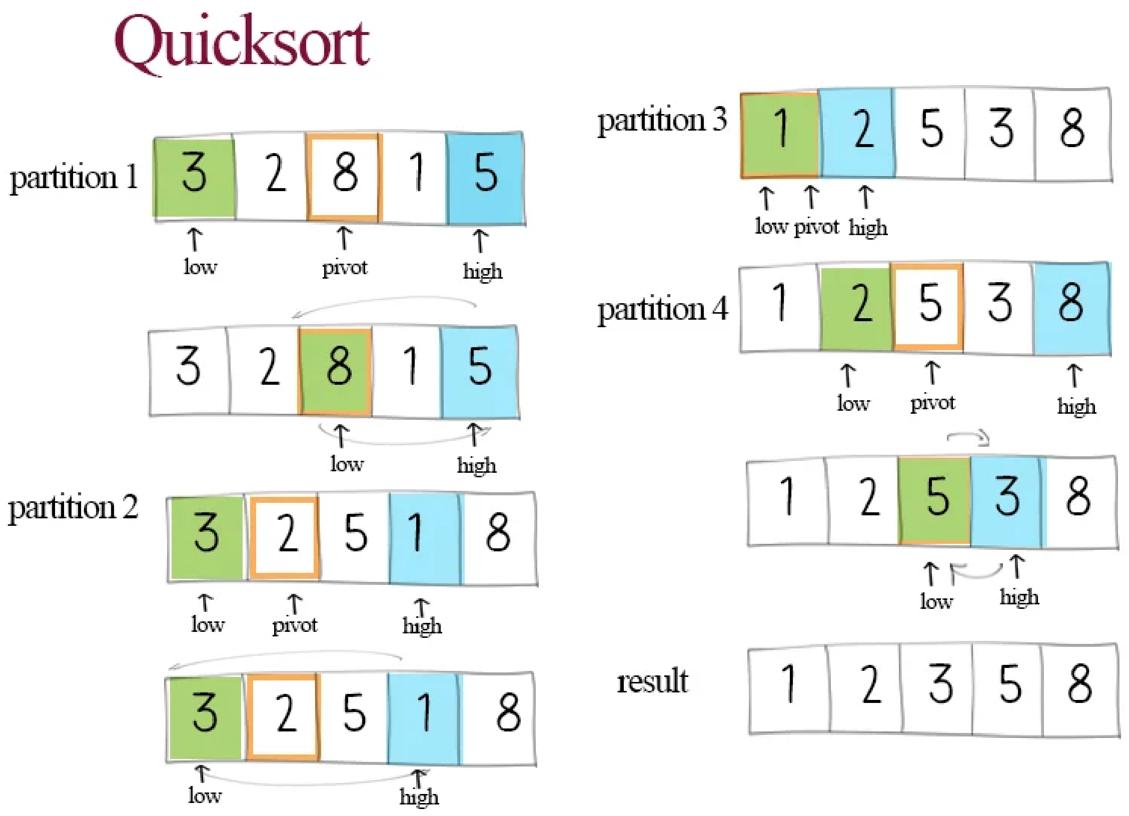
The divide-and-conquer technique works by recursively breaking down a problem into two or more sub-problems of the same or related type until these become simple enough to be solved directly.

Example 1: **Merge sort** – divides the array in half, sorts each of those halves, and then merges them together.



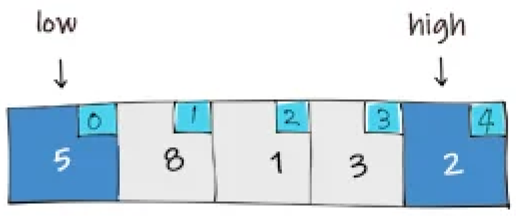
Merge sort in Java, JavaScript, Python and Doodle

Example 2: **Quicksort** – partitions the array into two subarrays based on the pivot, moving the larger ones to the right, and smaller ones to the left.



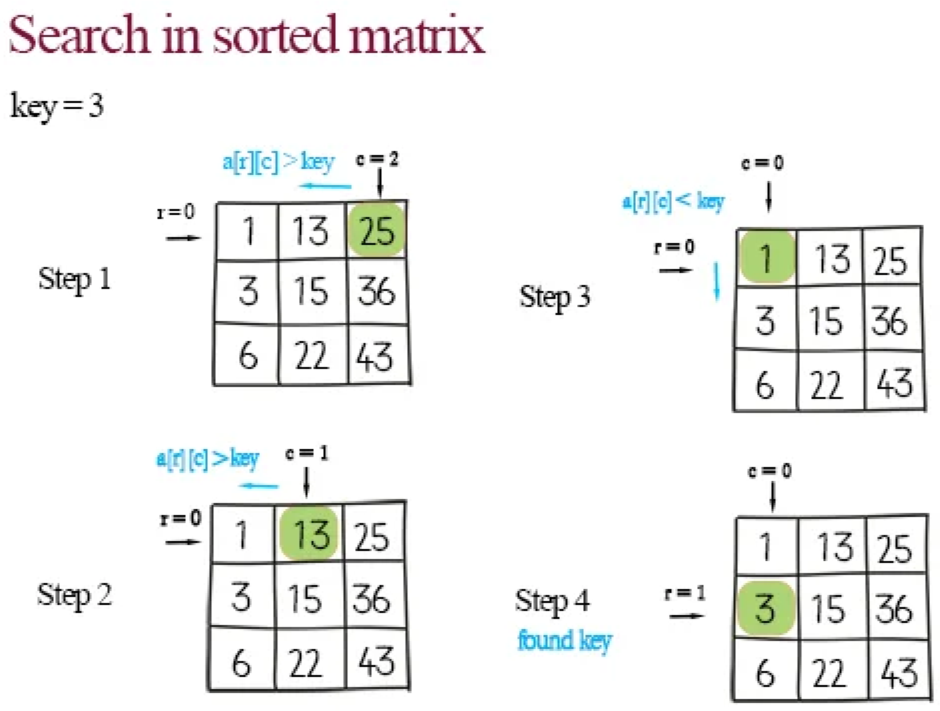
Quicksort in Java, JavaScript, Python and Doodle

4. Two pointers



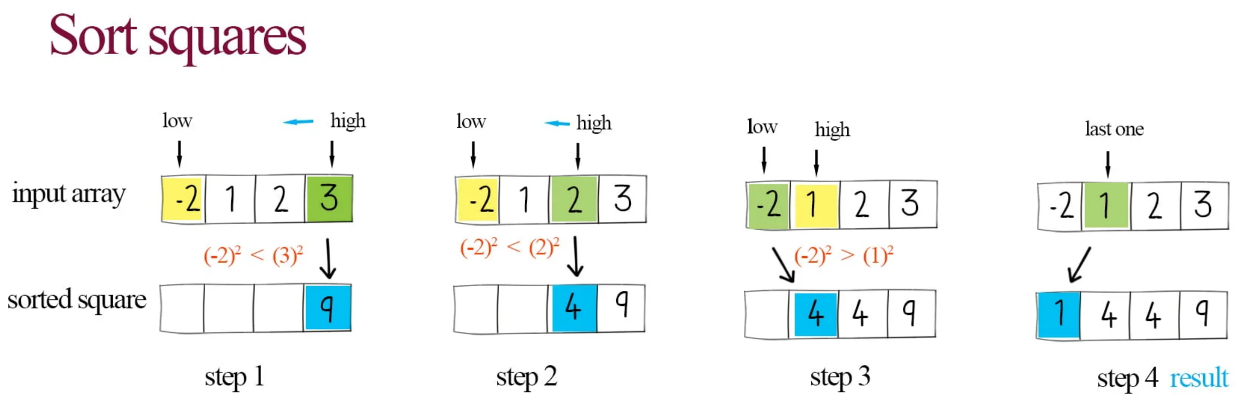
Two pointers are two indices in an array, pointing to either start and end or slower and faster. They move in different directions or paces in each iteration. By using two pointers in the array, two elements are processed per loop. This helps to reduce the time with fewer iterations.

Example 1: **Search in a sorted 2d array** – search when the matrix is sorted horizontally and vertically (Time complexity should be O(n)).



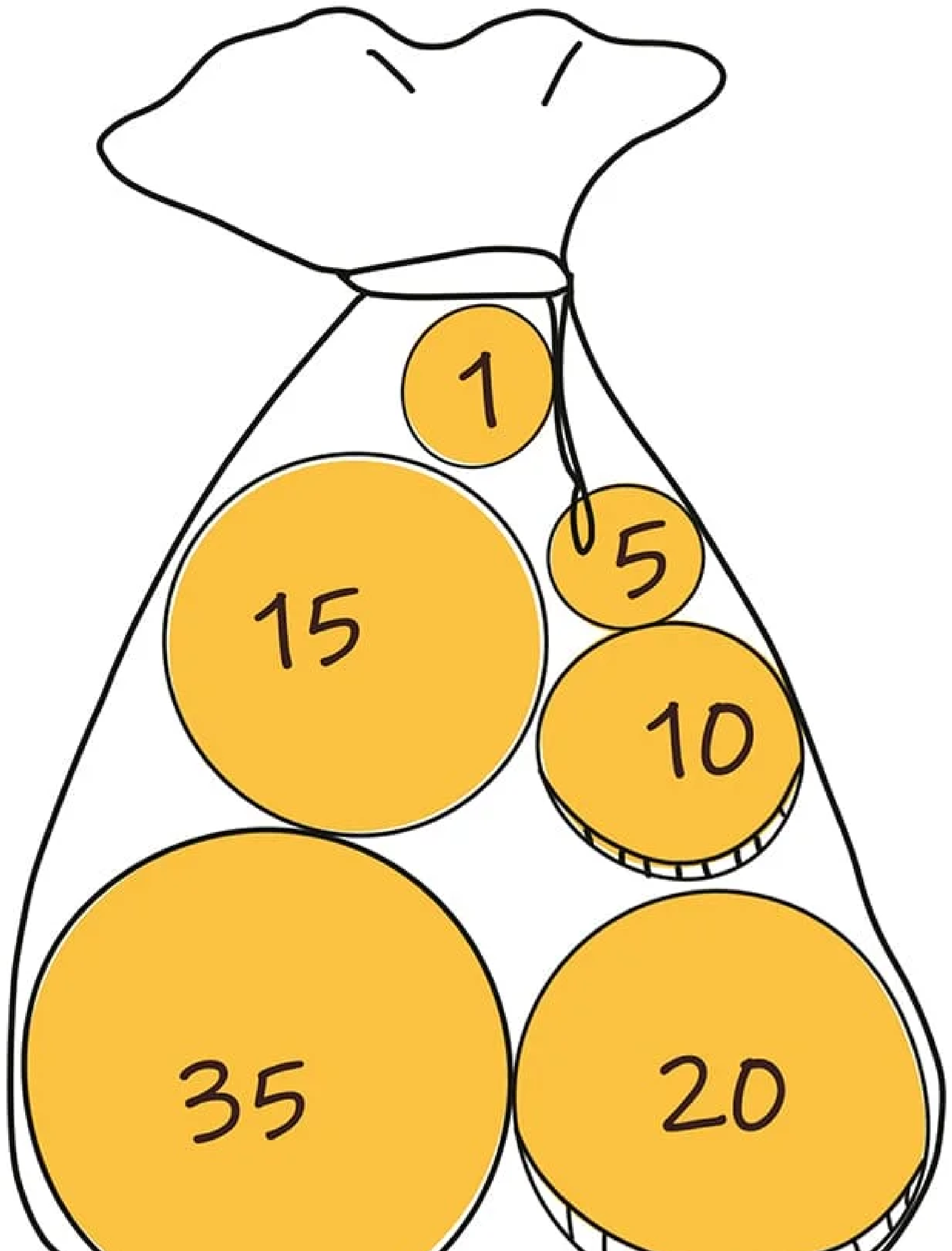
Matrix operations in Java, JavaScript, Python

Example 2: **Sort squares** – sort the squares of elements in a sorted array in one pass (O(n) time).



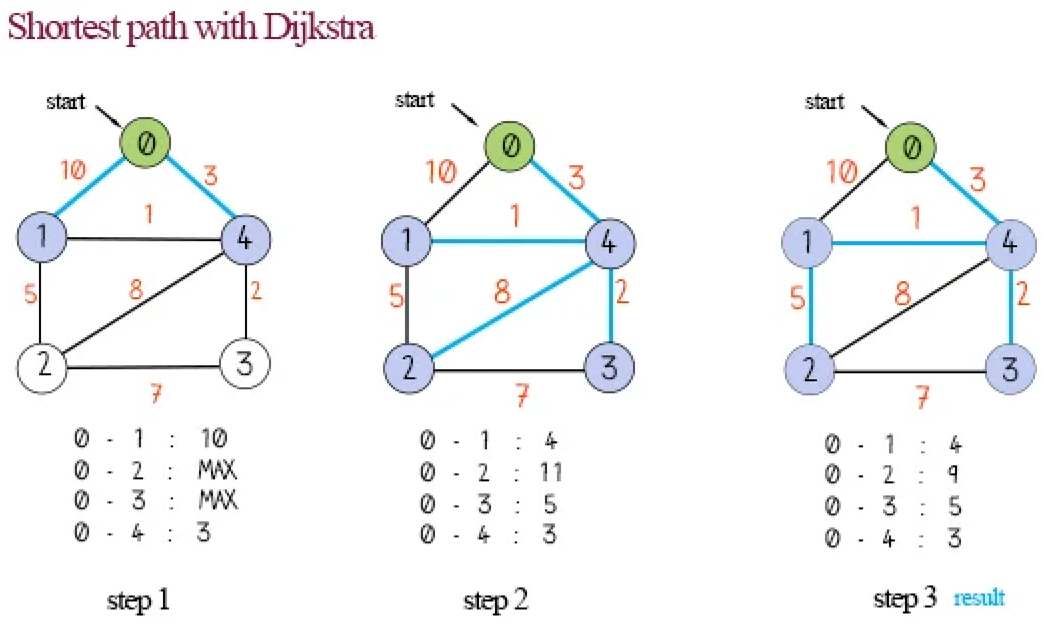
Sort squares with optimization

5. Greedy


1
15
5
10
35
20

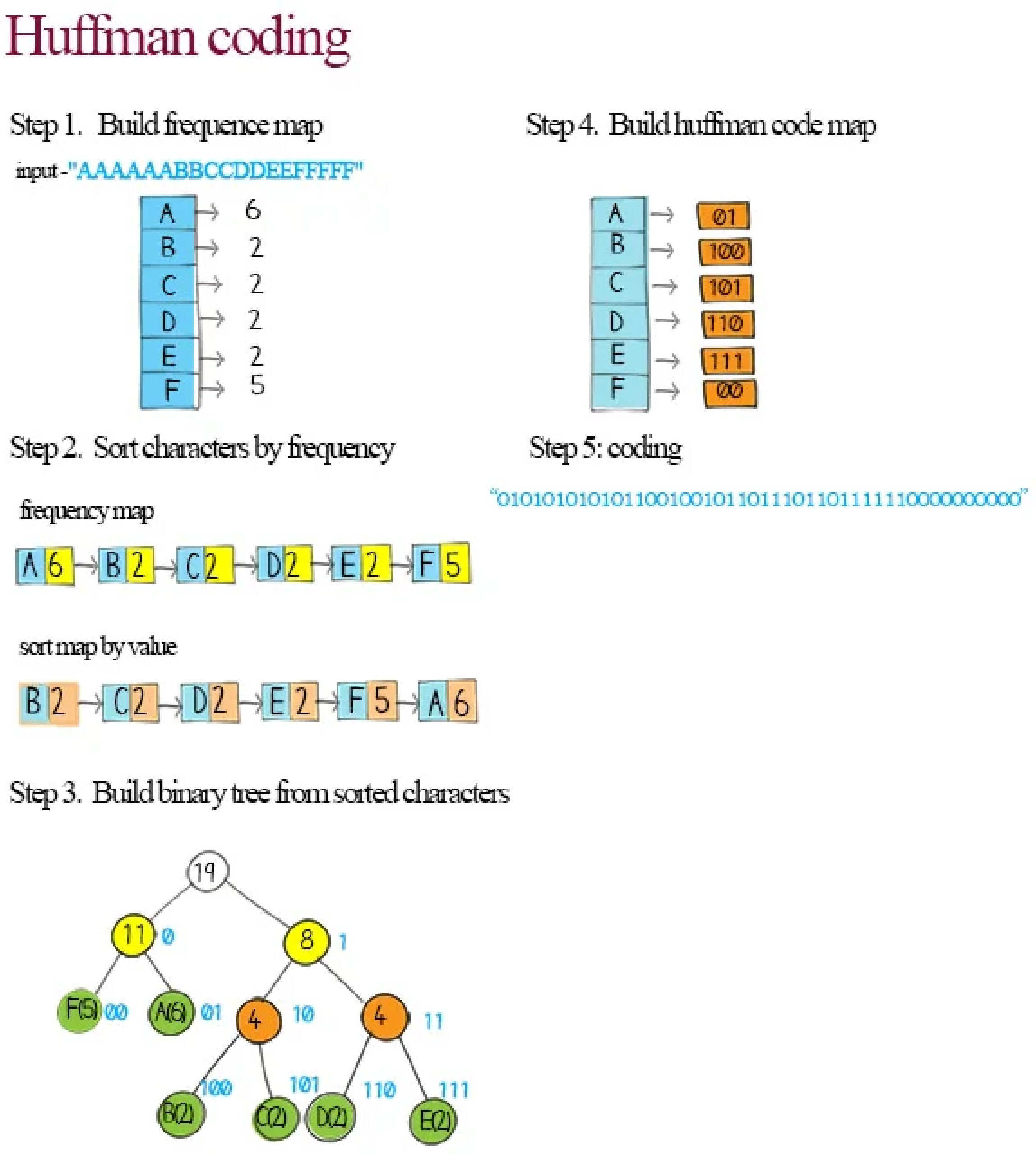
The greedy algorithm chooses the most obvious and immediate benefit item from the list so that the locally optimal choice leads to a globally optimal solution. It is usually implemented by sorting or partially sorting (Priority queue).

Example 1: **Find the shortest path with Dijkstra**– repeatedly picks the un-visited vertex with the lowest distance. When all vertices have been evaluated, the result is the shortest path.



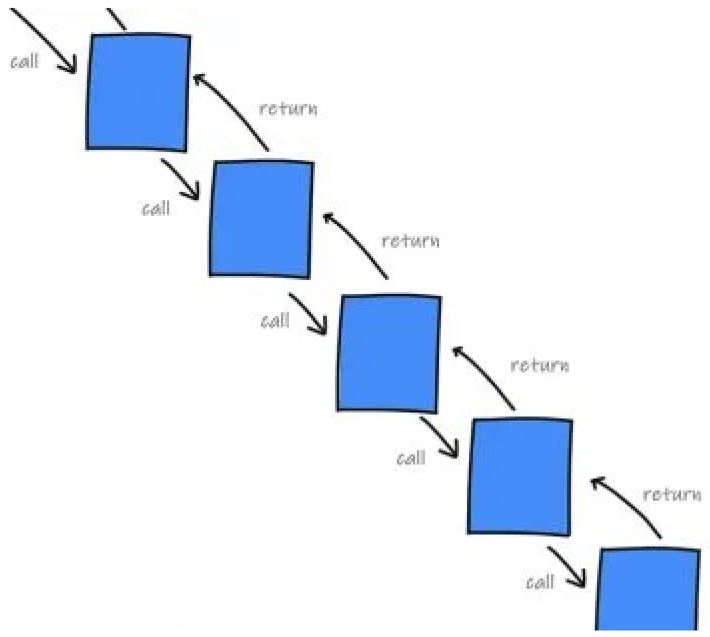
Dijkstra in Java, JavaScript, Python

Example 2: **Huffman coding** – generate the binary code based on the frequencies of corresponding characters in the input string.



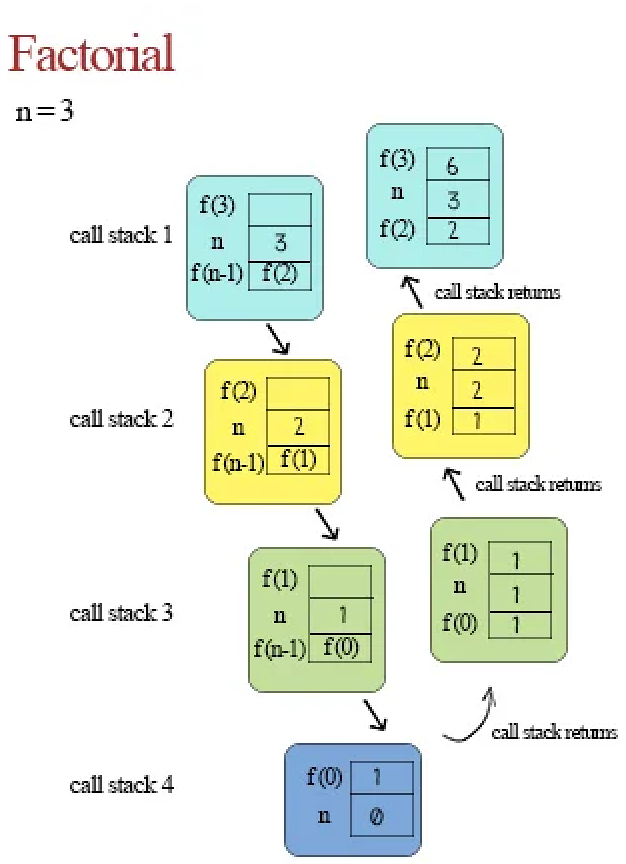
Huffman coding in Java, JavaScript, Python

6. Recursion



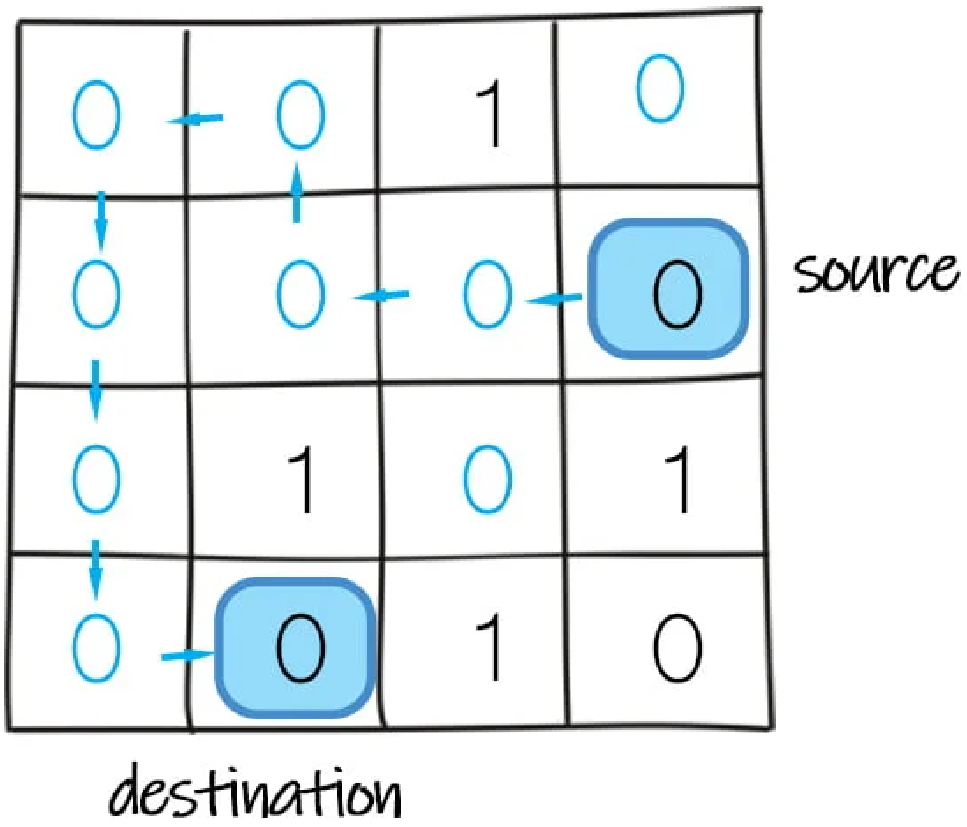
Recursion is a technique that a function or an algorithm calls itself. The termination condition should be defined so that when the condition is met, the rest of the call stacks return from the last call to the first.

Example 1: **Factorial numbers** – denoted as n!, is the product of all integers between n and 1. n! = n x (n-1) x (n-2) … x1.



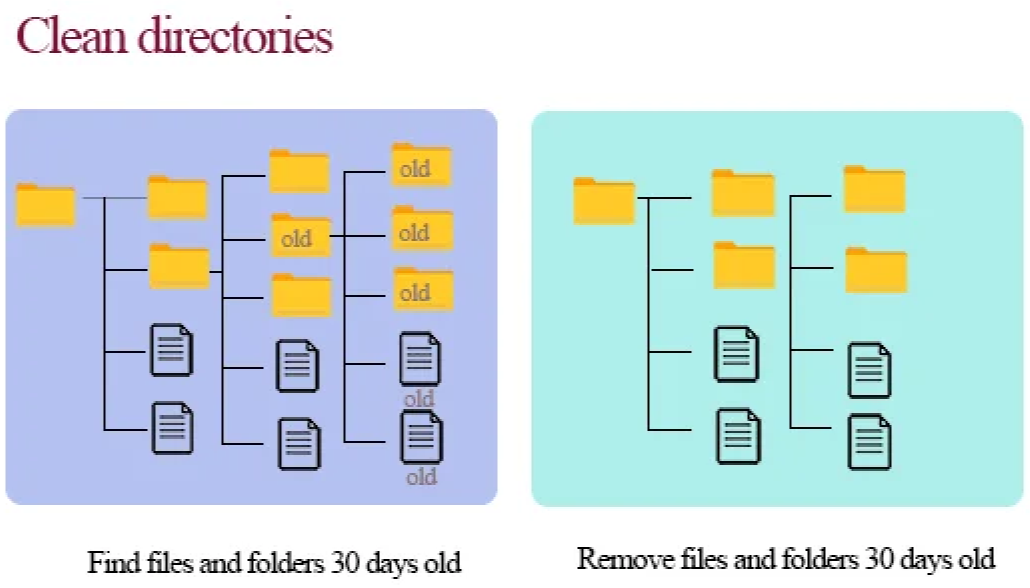
Factorial number in Java, JavaScript, Python and Doodle

Example 2: **Depth-first** search and matrix – Depth-first search (DFS) is used to traverse or search in a matrix that represents a graph. DFS can be implemented with recursion.



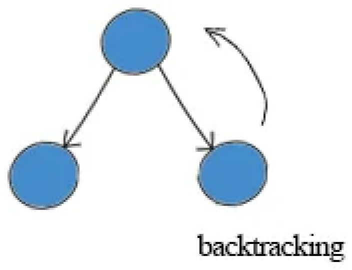
Depth first search in matrix using recursion

Example 3: **Clean directories in the file system** – clean out files that are older than certain dates, and remove the empty directories after the files are deleted.


Clean directories
old
old
old
old
old
old
Find files and folders 30 days old
Remove files and folders 30 days old

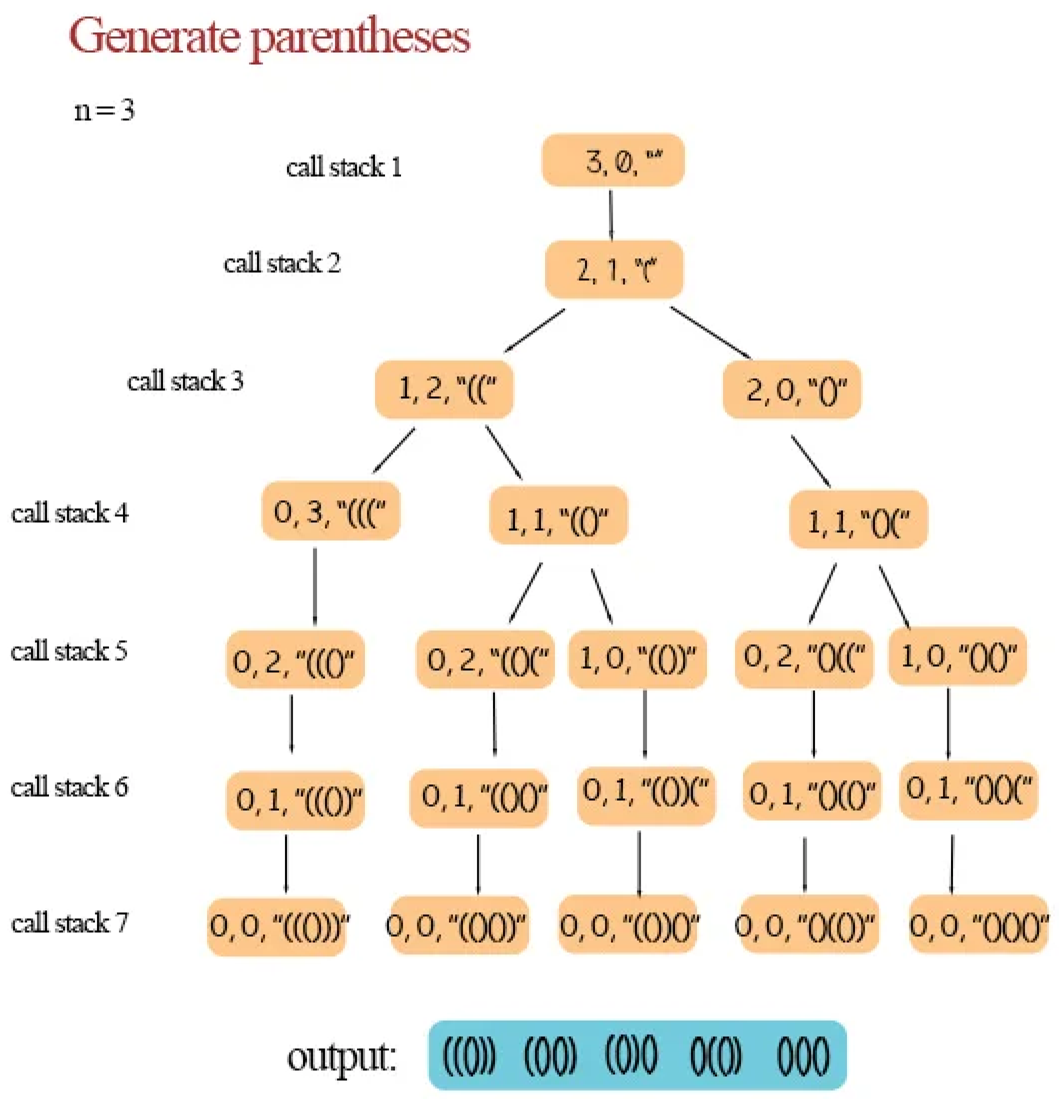
Clean directories in file system

7. Backtracking



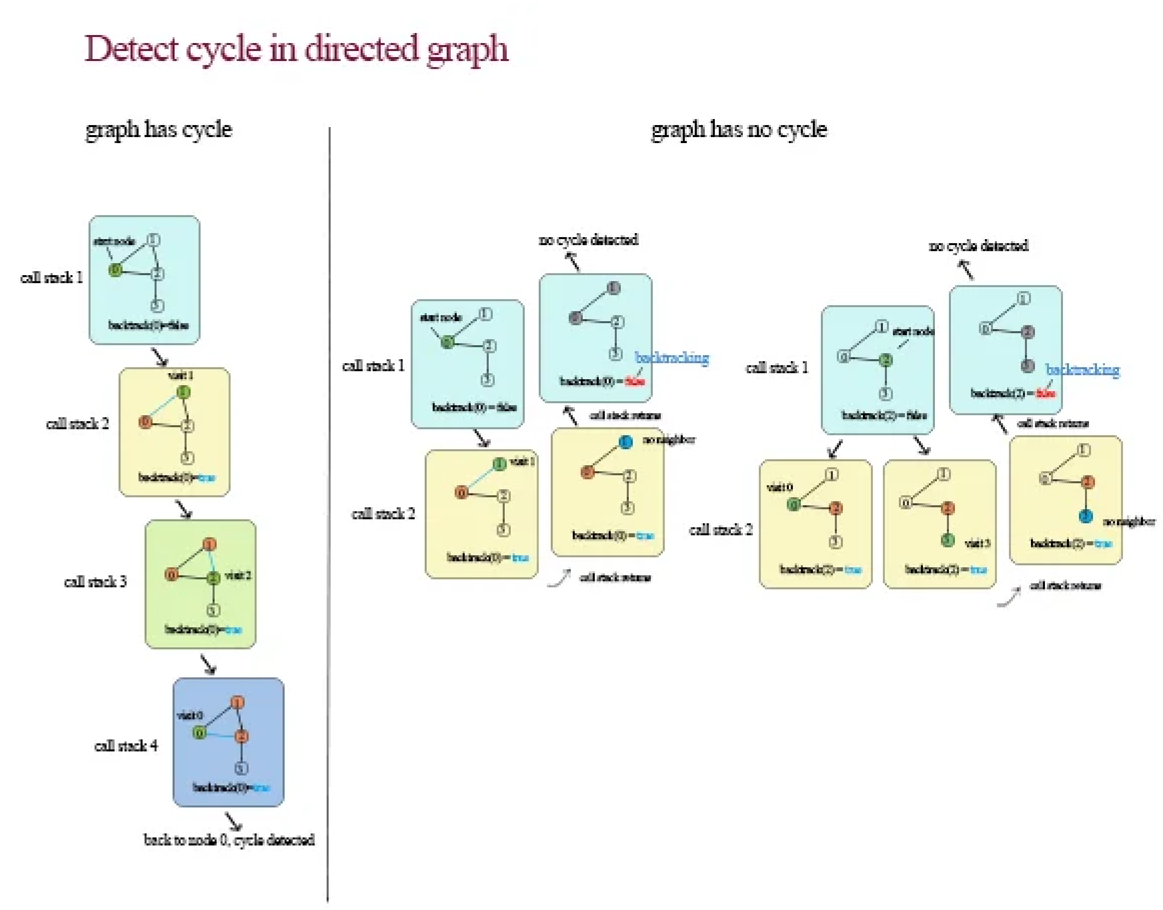
Backtracking is a method for solving problems recursively. It incrementally builds candidates to the solutions and removes the candidates (“backtracks”) that fail to satisfy the constraints of the problems.

Example 1: **Generate valid parentheses** – generate all possible expressions that contain n pairs of valid parentheses.



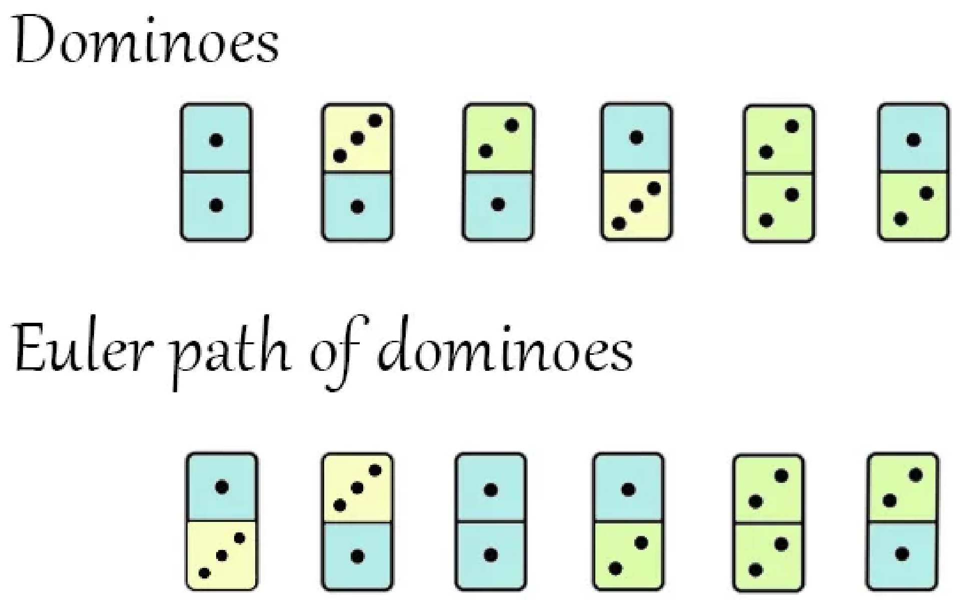
Generate valid parentheses

Example 2: **Detect cycle in the** directed graph – detect whether the graph comprises a path that starts from a node and ends at the same node.



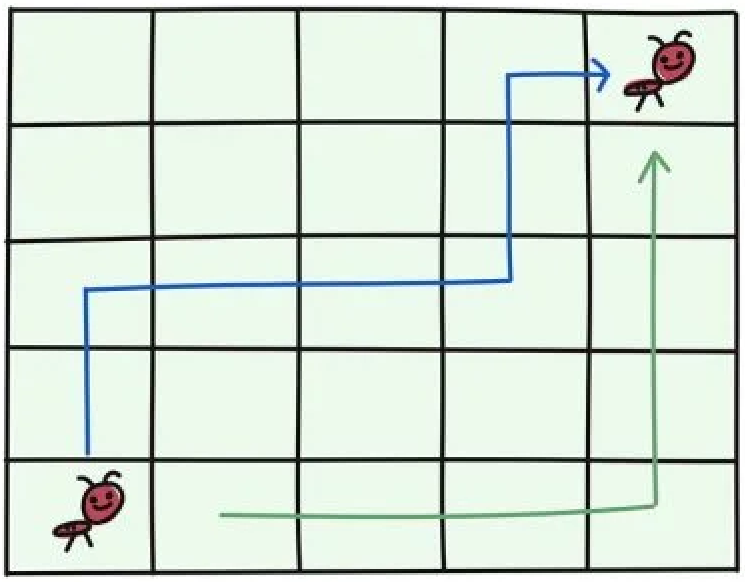
Detect cycle in directed graph in Java, JavaScript, Python

Example 3: **Domino Eulerian path** – Given a set of dominoes, order them so that the number on the bottom of the domino in front is equal to the number on top of the domino behind.



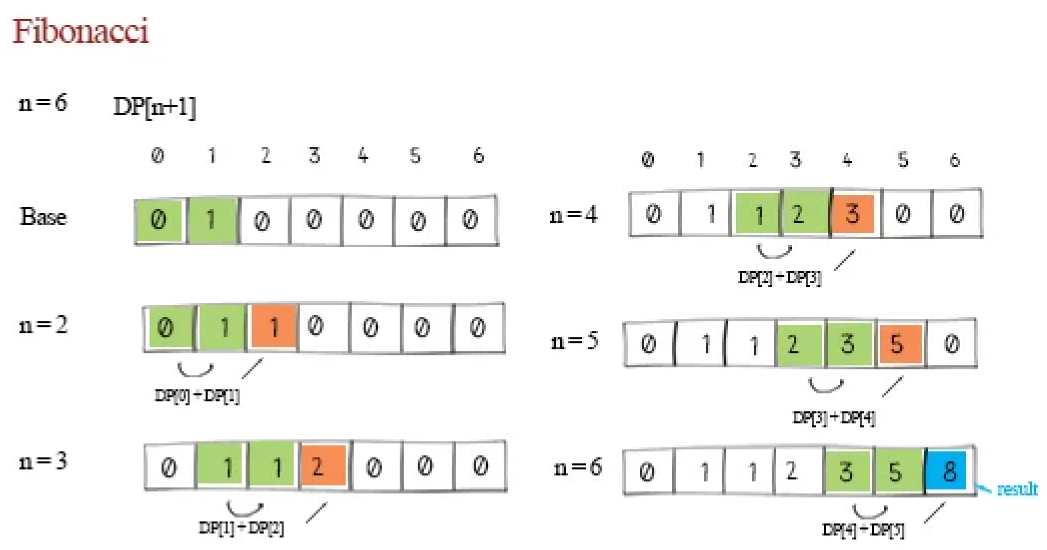
Domino Eulerian path

8. Dynamic programming



Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler sub-problems, solving each of those sub-problems just once, and storing their solutions using data structure, such as arrays, matrices, or maps. So the next time the same sub-problem occurs, one simply looks up the previously computed solution, thereby saving computation time. The intuition behind dynamic programming is that we trade space for time.

Example 1: **Fibonacci numbers** – a sequence of numbers, in which each number is the sum of the two preceding ones. Dynamic programming is one of the solutions.

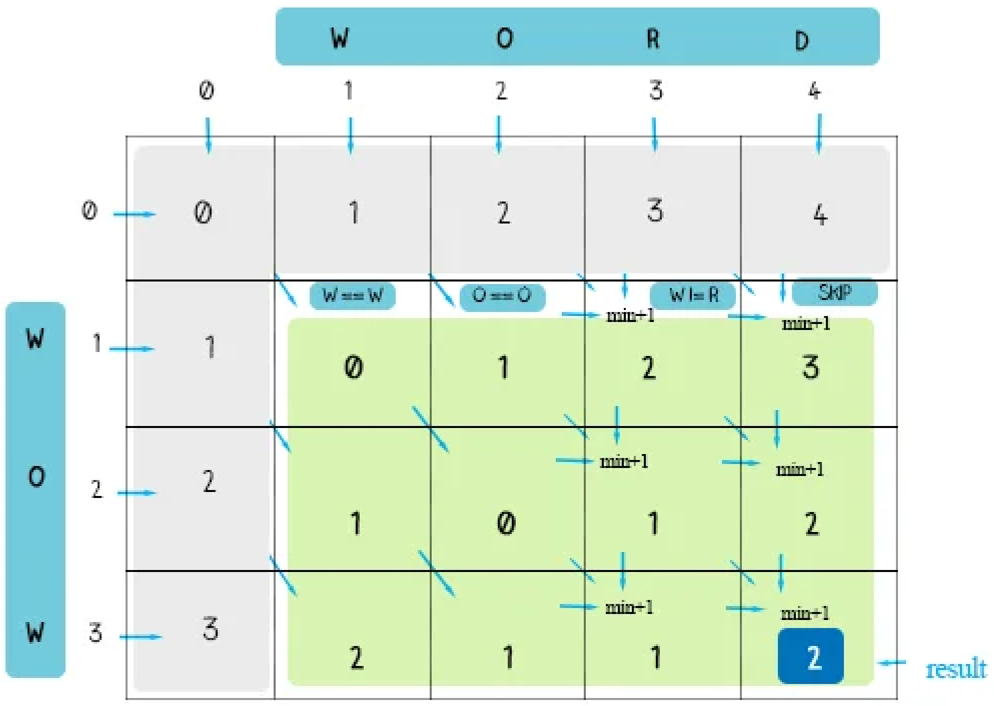


Fibonacci sequence 4 solutions and their complexity

Example 2: **Edit distance** – Find the number of actions (insert, delete, and update) to convert one word to another word.

Edit distance

word , wow DP[4][5]



Edit distance and autocorrect in Java

What are common techniques of algorithms?1. Recursion 2. Greedy 3. Divide and conquer 4. Backtracking 5. Dynamic programming 6. Two pointers

What are examples of algorithms?1. Binary search 2. Bubble sort 3. Merge sort 4. Quick sort 5. Depth-first search 6. Breadth-first search 7. Dijkstra’s algorithm 8. Huffman coding