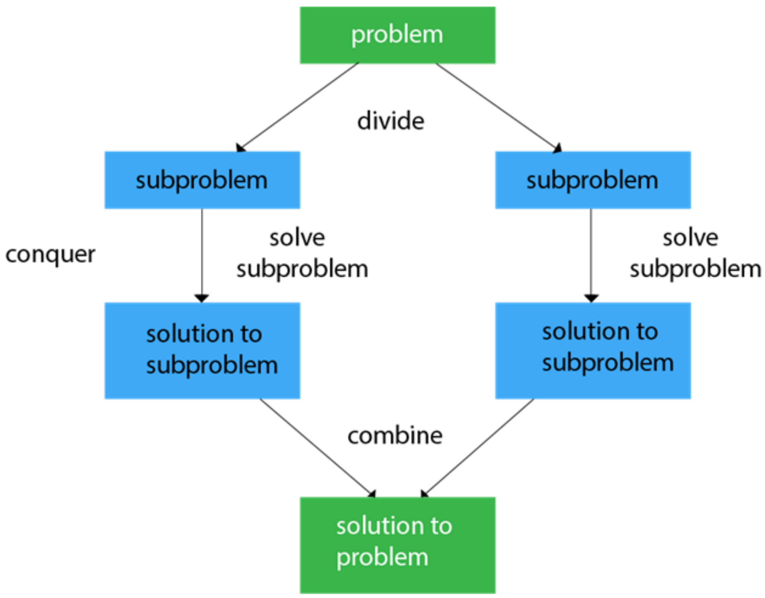
Divide and Conquer Concepts

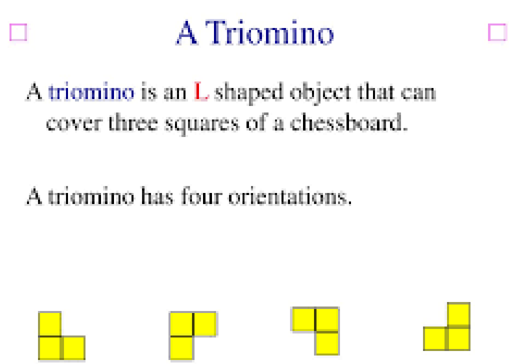
General Method

In divide and conquer approach, a problem is divided into smaller problems, then the smaller problems are solved independently, and finally the solutions of smaller problems are combined into a solution for the large problem.



Defective Chessboard

A defective chessboard is **a chessboard that has one unavailable (defective) position**. A triomino is an L shaped object that can cover three squares of a chessboard. A triomino has four orientations.



The Defective Chessboard Problem

GIVEN CONDITIONS:-

1. We have a chessboard of size n x n , where $n = 2^k$
2. Exactly on square is defective in the chessboard.
3. The tiles(trominoes) are in L-shape i.e. 3 squares.

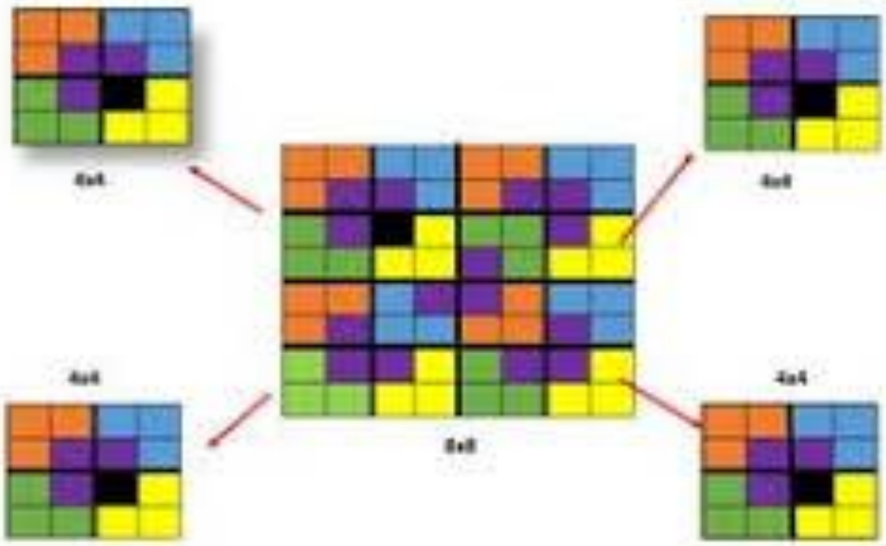
OBJECTIVE

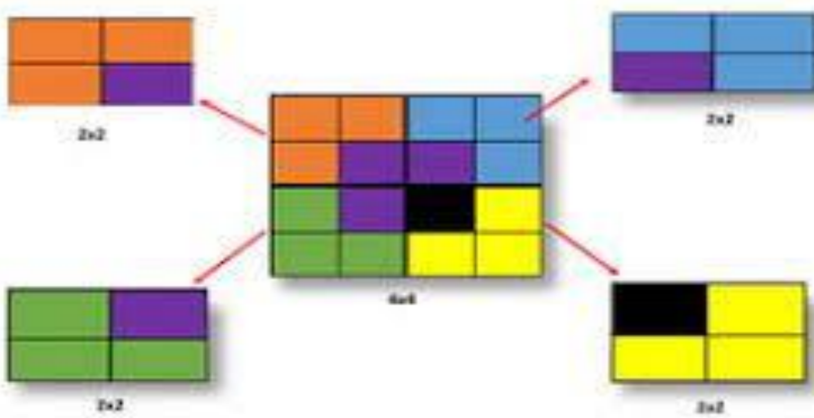
Cover all the chessboard with L-shape tiles(trominoes), except the defective squ

Is it possible to solve this?

Absolutely, it is possible to cover all non-defective squares.
Let's see how

- As the size of the chessboard is n x n and $n=2^k$
- Therefore, Total no. of squares $=2^k \times 2^k = 2^{2k}$
- No. of non-defective squares $= 2^{2k}-1$
- Now, for the value of K, $2^{2k}-1$ is divisible by 3.
- For E.g. K=1 , $2^{2(1)}-1 = 3$ is divisible by 3.
- $K=2$, $2^{2(2)}-1 = 15$ is divisible by 3.


4x4
4x4
4x4
4x4
8x8


2x2
2x2
4x4
2x2
2x2

Binary Search

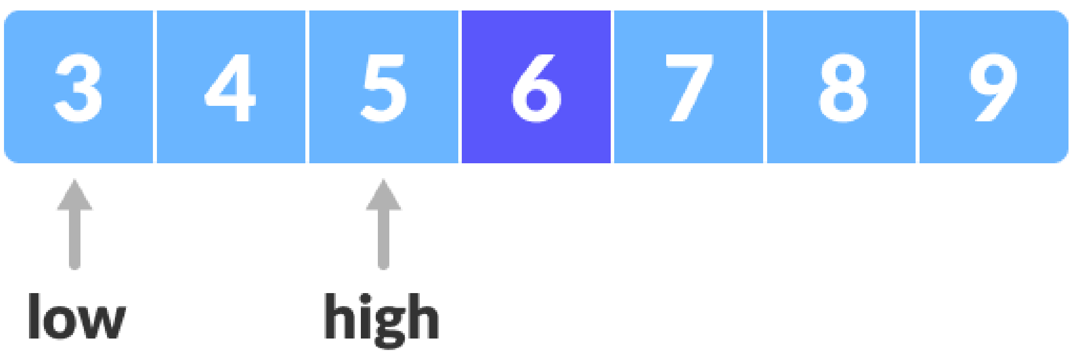
Binary search **looks for a particular item by comparing the middle most item of the collection**. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item.

1. The array in which searching is to be performed is:

2. Set two pointers low and high at the lowest and the highest positions respectively.

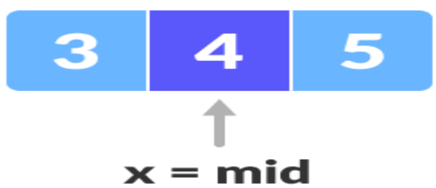
3. Find the middle element mid of the array ie. arr[(low + high)/2] = 6.

4. If x == mid, then return mid.Else, compare the element to be searched with m.
5. If x > mid, compare x with the middle element of the elements on the right side of mid. This is done by setting low to low = mid + 1.
6. Else, compare x with the middle element of the elements on the left side of mid. This is done by setting high to high = mid - 1.

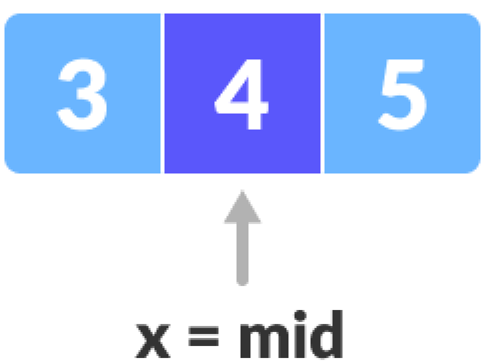


7. Repeat steps 3 to 6 until low meets high.

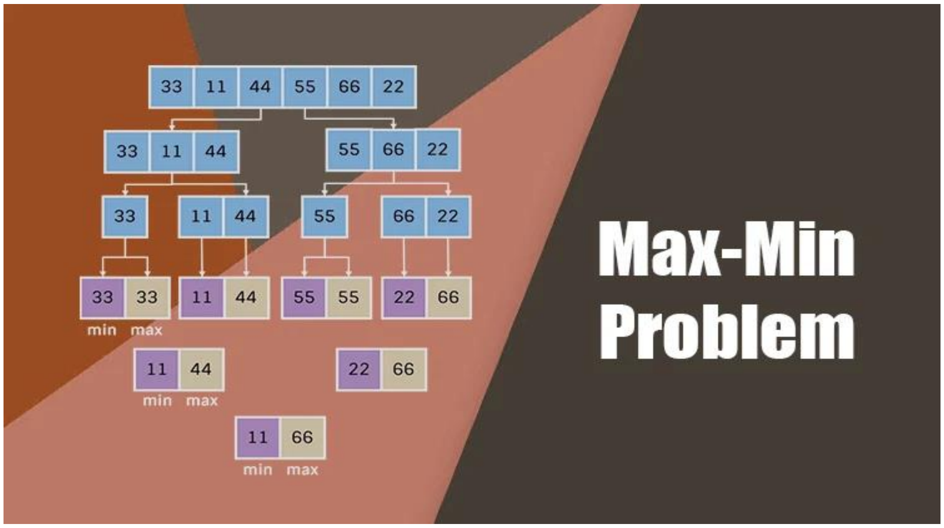
8.



9. x = 4 found

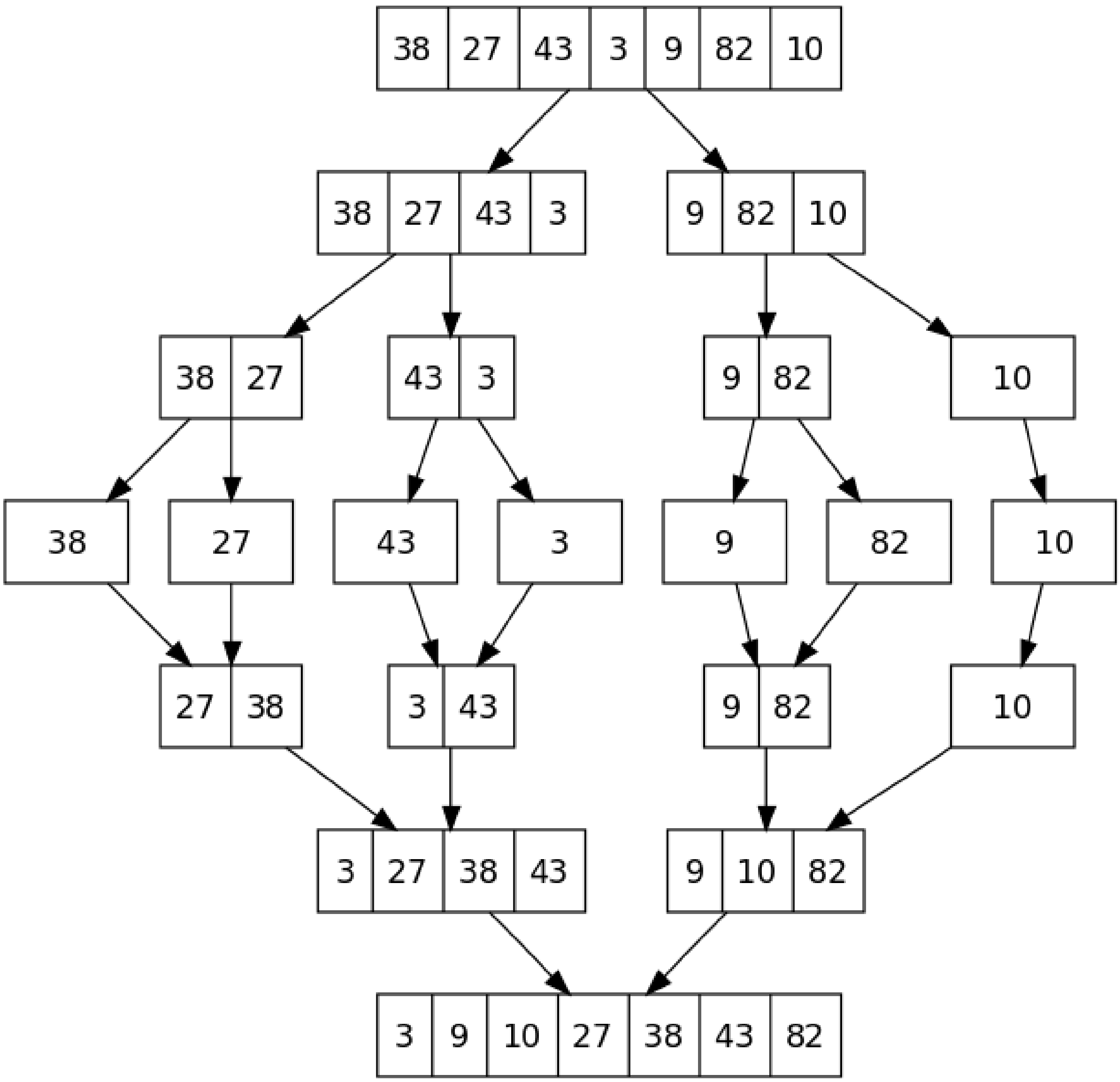


Max-Min Problem

In this approach, the array is divided into two halves. Then using recursive approach maximum and minimum numbers in each halves are found. Later, return the maximum of two maxima of each half and the minimum of two minima of each half. 

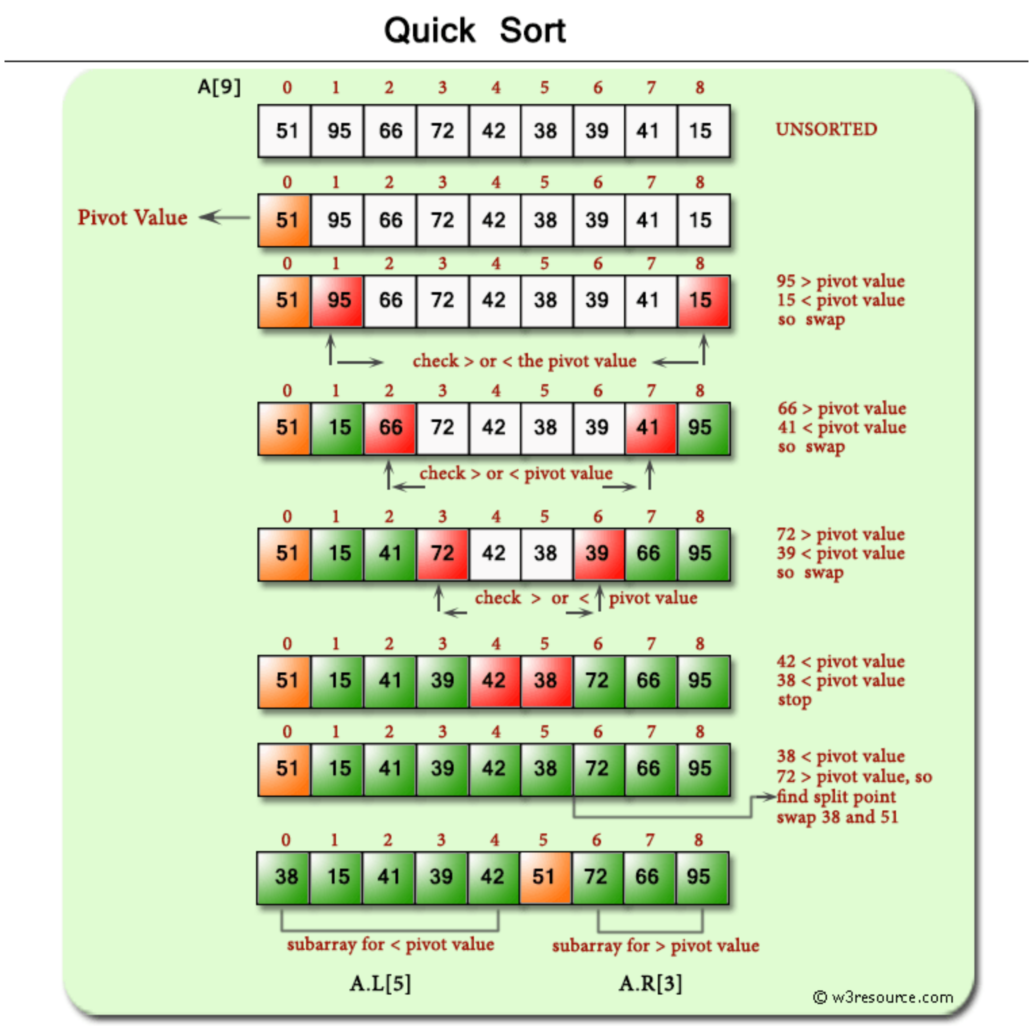
Merge Sort

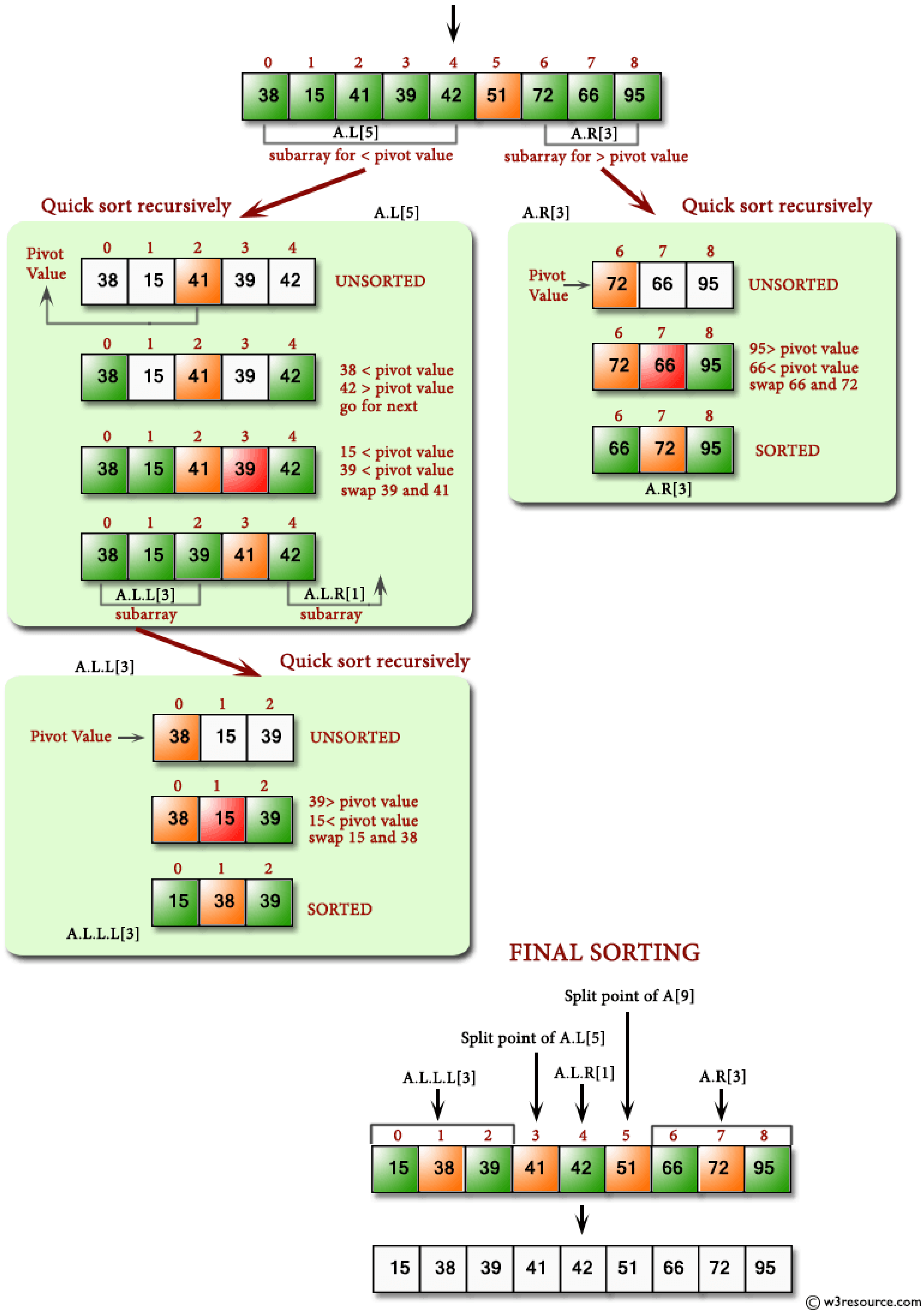
The MergeSort function keeps on splitting an array into two halves until a condition is met where we try to perform MergeSort on a subarray of size 1, i.e., p == r. And then, it combines the individually sorted subarrays into larger arrays until the whole array is merged.



Quick Sort

It is an algorithm of Divide & Conquer type. Divide: Rearrange the elements and split arrays into two sub-arrays and an element in between search that each element in left sub array is less than or equal to the average element and each element in the right sub- array is larger than the middle element.

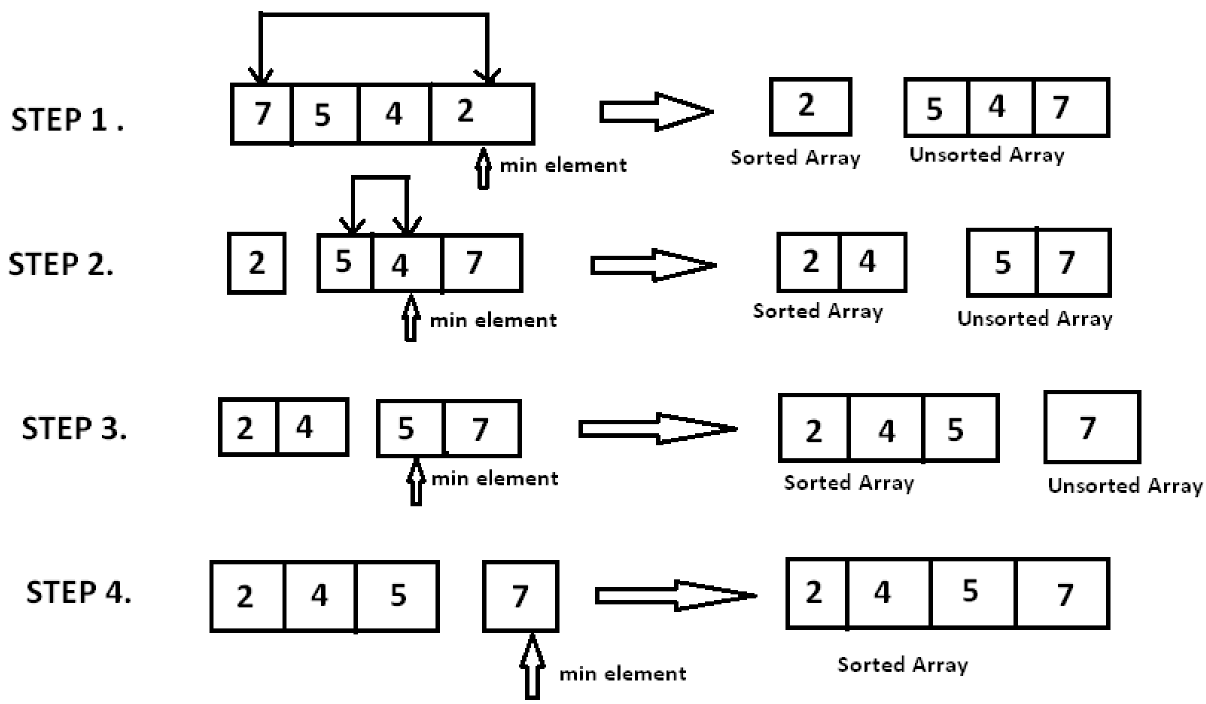




0 1 2 3 4 5 6 7 8
38 15 41 39 42 51 72 66 95
A.L[5]
subarray for < pivot value
A.R[3]
subarray for > pivot value
Quick sort recursively
A.L[5]
Pivot Value
0 1 2 3 4
38 15 41 39 42
UNSORTED
38 < pivot value
42 > pivot value
go for next
15 < pivot value
39 < pivot value
swap 39 and 41
38 15 39 41 42
A.L.L[3] subarray
A.L.R[1] subarray
Quick sort recursively
A.R[3]
Pivot Value
6 7 8
72 66 95
UNSORTED
95> pivot value
66< pivot value
swap 66 and 72
66 72 95
SORTED
A.R[3]
Quick sort recursively
A.L.L[3]
Pivot Value
0 1 2
38 15 39
UNSORTED
39> pivot value
15< pivot value
swap 15 and 38
15 38 39
SORTED
A.L.L.L[3]
FINAL SORTING
Split point of A[9]
Split point of A.L[5]
A.L.L.L[3]
A.L.R[1]
A.R[3]
0 1 2 3 4 5 6 7 8
15 38 39 41 42 51 66 72 95
15 38 39 41 42 51 66 72 95
© w3resource.com

Selection Sort

Selection sort is an effective and efficient sort algorithm based on comparison operations. It adds one element in each iteration. You need to select the smallest element in the array and move it to the beginning of the array by swapping with the front element.



Strassen's Matrix Multiplication

Strassen, is **an algorithm for matrix multiplication**. It is faster than the standard matrix multiplication algorithm for large matrices, with a better asymptotic complexity, although the naive algorithm is often better for smaller matrices.

Strassen's Matrix multiplication **can be performed only on square matrices where n is a power of 2**. Order of both of the matrices are n × n. Divide X, Y and Z into four (n/2)×(n/2) matrices as represented below − Z=[IJKL] X=[ABCD] and Y=[EFGH]

$$P = (A_{11}+A_{22}) * (B_{11}+B_{22})$$

$$Q = (A_{21}+A_{22}) * B_{11}$$

$$R = A_{11} * (B_{12} - B_{22})$$

$$S = A_{22} * (B_{21} - B_{11})$$

$$T = (A_{11}+A_{12}) * B_{22}$$

$$U = (A_{21} - A_{11}) * (B_{11} + B_{12})$$

$$V = (A_{12} - A_{22}) * (B_{21} + B_{22})$$

$$C_{11} = P+S-T+V$$

$$C_{12} = R+T$$

$$C_{21} = Q+S$$

$$C_{22} = P+R-Q+U$$

Multiply the matrix using strassen's Matrix Multiplication.

$$A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} 6 & 7 \\ 3 & 8 \end{bmatrix}$$

A11 =1 , A12 =3 , A21 = 7 , A22 = 5

B11 = 6 , B12 = 7 , B21 = 3 , B22 = 8

APPLY STRASSEN'S FORMULA WE GET

P = (1+5)(6+8) = 6* 14 =84

Q = 6 (7+5) = 6*12 = 72

S = 5 (3-6) = -15

T = 8 (1+3) = 32

U = (6+7) (7-1) = 13 * 6 =78

V = (3+8)(3-5) = 11 * -2 = -22

NOW,

C11 = P + S - T + V

= 84 + (-15) - 32 + (-22)

R = 1 (7-8) = -1

= 15

C12 − R + T

= -1 + 32

= 31

C21 = Q + S

= 72 ı (15)

= 57

C22 – P + R - Q + U

= 84 + (-1) - 72 + 78

= 89

SO MATRIX C WILL BE:

$$C = \begin{bmatrix} 15 & 31 \\ 57 & 89 \end{bmatrix}$$