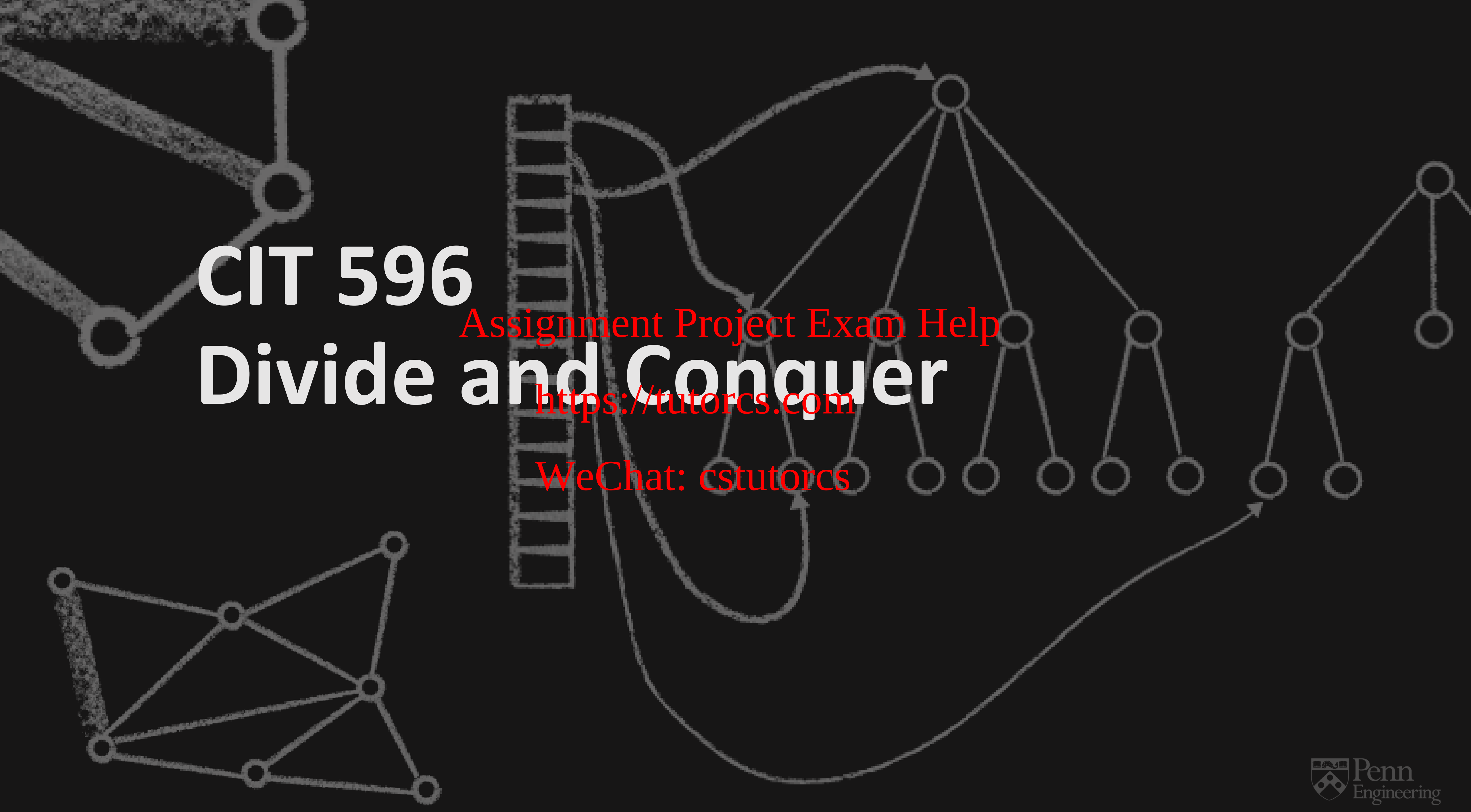


CIT 596

Divide and Conquer

DIVIDE AND CONQUER TECHNIQUE

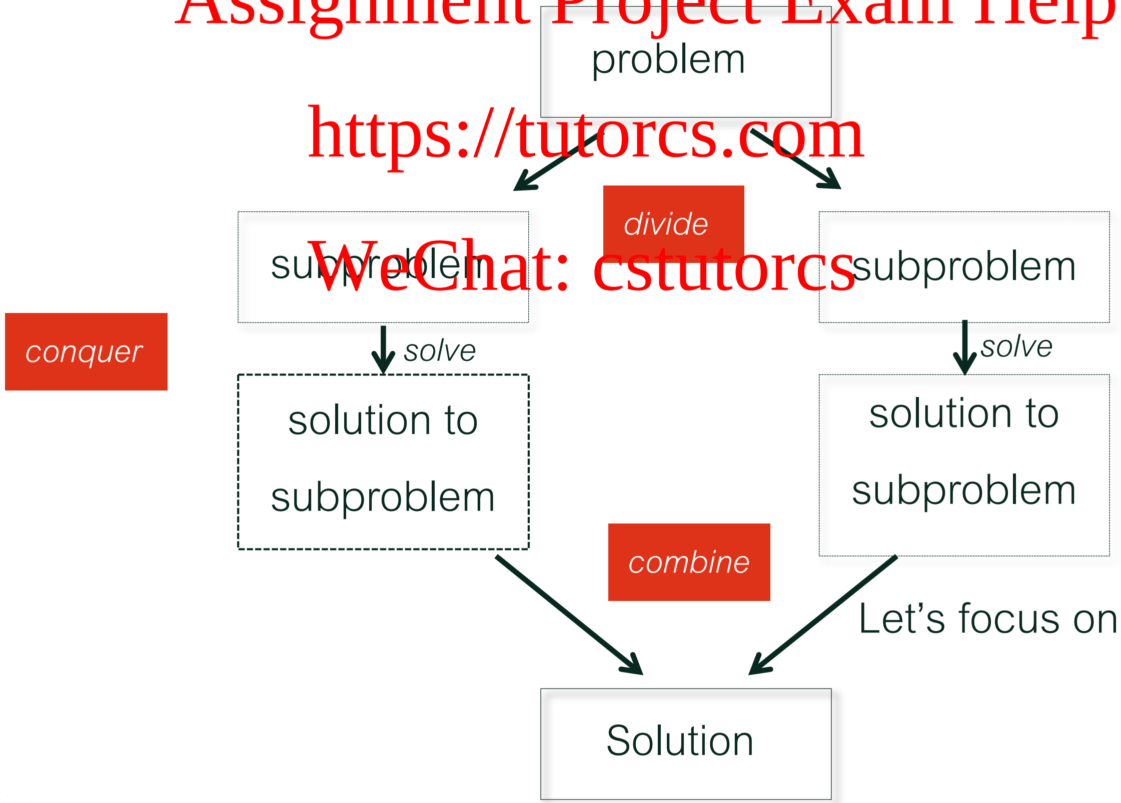
- Break input into roughly equal halves (Divide)
- Solve the problem in each of the halves (Conquer)
- Put together solution to the whole problem (Combine)

Merge sort: Sorting algorithm designed as above

- Split input array into two halves
- Sort each half separately
- Combine the sorted arrays

ALGORITHM DESIGN:
DIVIDE AND CONQUER PARADIGM

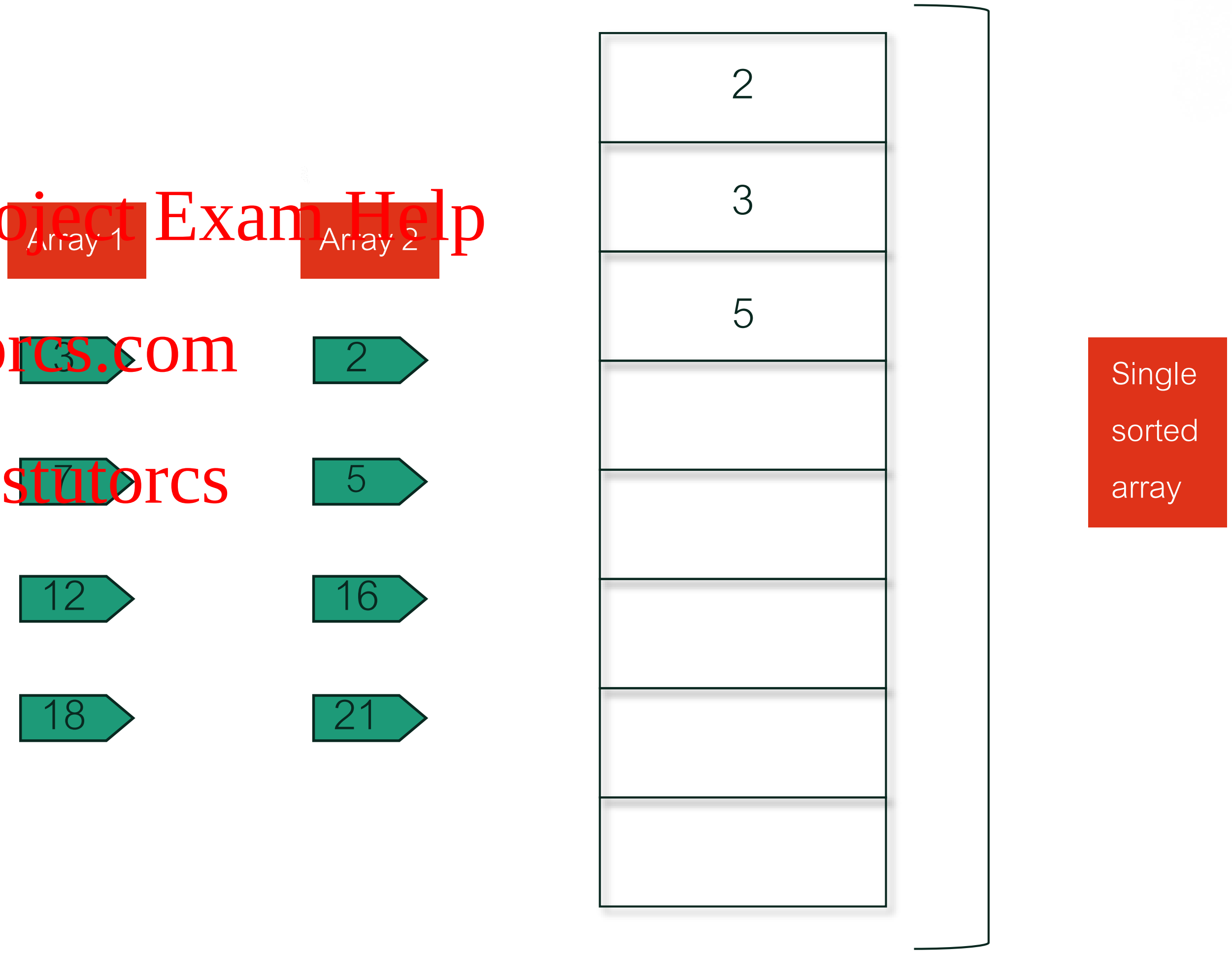
- Idea: solve a problem by splitting it into pieces, solving those pieces recursively, and merging them to solve the larger problem



Let’s focus on the combine step first.

MERGING TWO SORTED LISTS

- Input: two sorted arrays of size n and m
- Output: a single sorted array of size $n + m$
- Continue to fill all 8 positions in output array
- How many steps?
  - Each comparison places one input
  - Total number of inputs $m + n$
  - Steps = $O(m + n)$



MergeSort Pseudocode

```
merge(A, B):
   C = new array[len(A) + len(B)]
   i, j, k <- 0
   while i < len(A) and j < len(B):
      if A[i] < B[j]:
         C[k] <- A[i]
         i++, k++
      else:
         C[k] <- B[j]
         j++, k++
   while i < len(A):
      C[k++] <-A[i++]
   while j < len(B):
      C[k++] <- B[j++]
   return C
```

```
MergeSort(A)

    rec-mergesort(A, 0,len(A)-1)

rec-mergesort(A, lo, hi):
    if (hi - lo <= 0) return
    mid = (lo + hi) / 2
    rec-mergesort(A, lo, mid)
    rec-mergesort(A, mid+1,hi)
    C = merge(A[lo:mid],A[mid+1:hi])
    copy elements from C back into A
```

BASE CASES

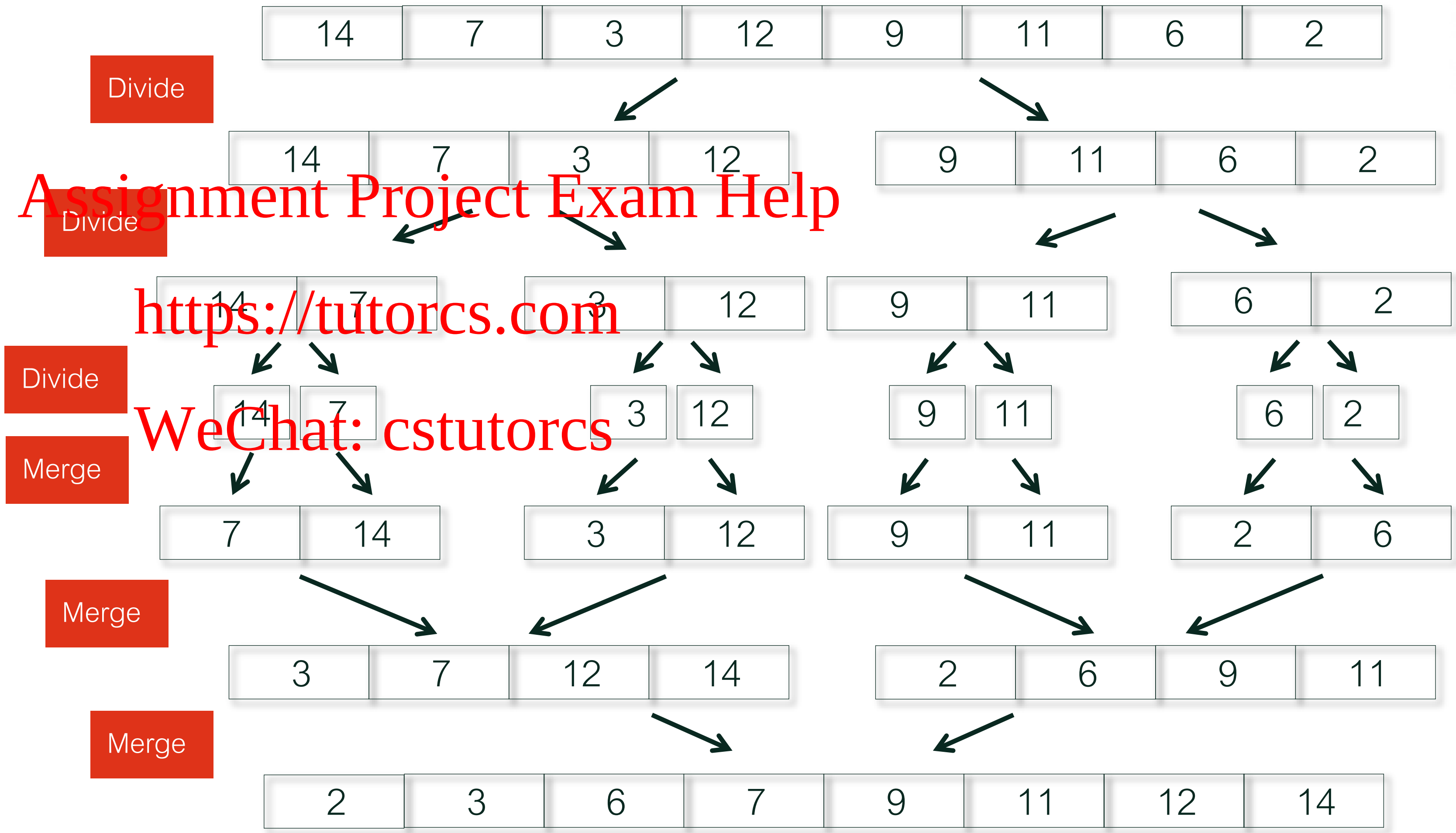
- Recursion bottoms out when we have arrays of length 1
- Such arrays are already sorted. So $T(1) = 0$
- Array of length 2 leads to two recursive calls on arrays of length 1
- These calls return immediately and then we "merge" the two sorted arrays of length 1
  - Merge takes one step.

DIVIDE AND CONQUER: MERGE SORT

Merge is “Combine” step

`mergesort`

- Divide array
- Sort each half
- Merge sorted halves



ANALYZING MERGE SORT: RECURRENCE RELATIONS

- Let $T(n)$ be the time for `mergesort` to sort a list of `n` elements
- What are the steps going into $T(n)$?
  - Divide: 0 steps
  - Conquer: need to sort 2 arrays of length $n/2$ each: $\leq 2T(n/2)$ steps
  - Combine: merge 2 sorted lists of length $n/2$ each: $\leq n$ steps

$$T(n) \leq 2\,T\left(\frac{n}{2}\right) + n$$
$$T(1) = 0$$

- *In the next segment, we will see the Master Theorem for solving such recurrences*