Merge Sort

Merge Sort—divide-and-conquer

- **Divide:** divide the n-element sequence into two subproblems of $n/2$ elements each.
- **Conquer**: sort the two subsequences recursively using merge sort. If the length of a sequence is 1, do nothing since it is already in order.
- **Combine**: merge the two sorted subsequences to produce the sorted answer.

Merge Sort - Algorithm

```
MergeSort(A, left, right) {
  if (left < right) {
      mid = floor((left + right) / 2);
      MergeSort(A, left, mid);
      MergeSort(A, mid+1, right);
      Merge(A, left, mid, right);
  }
}

// Merge() takes two sorted subarrays of A and
// merges them into a single sorted subarray of A
//    (how long should this take?)
```

Analysis of Merge Sort

<u>Statement   Effort</u>

```
MergeSort(A, left, right) {                    T(n)
   if (left < right) {                         Θ(1)
      mid = floor((left + right) / 2);            Θ(1)
      MergeSort(A, left, mid);                    T(n/2)
      MergeSort(A, mid+1, right);                 T(n/2)
      Merge(A, left, mid, right);                 Θ(n)
   }
}
```

- So $T(n) =$ $\Theta(1)$ when $n = 1$, and
  $2T(n/2) + \Theta(n)$ when $n > 1$
- So what (more succinctly) is $T(n)$?

Recurrences

- The expression:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

is a *recurrence*.

  - Recurrence: an equation that describes a function in terms of its value on smaller functions

Compute *T*(*n*) by Recursive Tree

- The recursive equation can be solved by recursive tree.
- $T(n) = 2T(n/2) + cn$, (See its Recursive Tree).
- lg n+1 levels, cn at each level, thus
- Total cost for merge sort is
  - $T(n) = cn\lg n + cn = \Theta(n\lg n)$.
  - Question: best, worst, average?
- In contrast, insertion sort is
  - $T(n) = \Theta(n^2)$.

Recursion tree of $T(n)=2T(n/2)+cn$

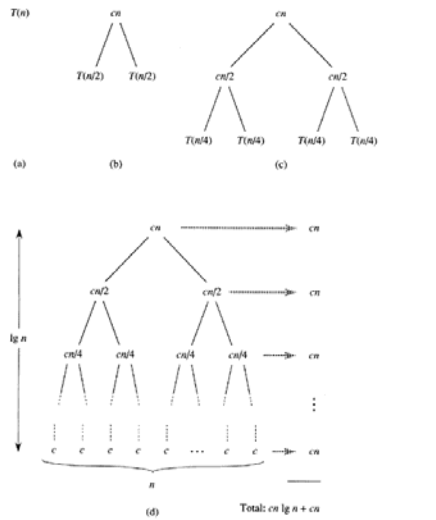

Figure 2.5 The construction of a recursion tree for the recurrence $T(n) = 2T(n/2) + cn$. Part **(a)** shows $T(n)$, which is progressively expanded in **(b)–(d)** to form the recursion tree. The fully expanded tree in part (d) has $\lg n + 1$ levels (i.e., it has height $\lg n$, as indicated), and each level contributes a total cost of cn. The total cost, therefore, is $cn \lg n + cn$, which is $\Theta(n \lg n)$.