Brute Force Strengths and Weaknesses

- Strengths
  - wide applicability
  - simplicity
  - yields reasonable algorithms for some important problems (e.g., matrix multiplication, sorting, searching, string matching)

- Weaknesses
  - rarely yields efficient algorithms
  - some brute-force algorithms are unacceptably slow
  - not as constructive as some other design techniques

Divide-and-Conquer

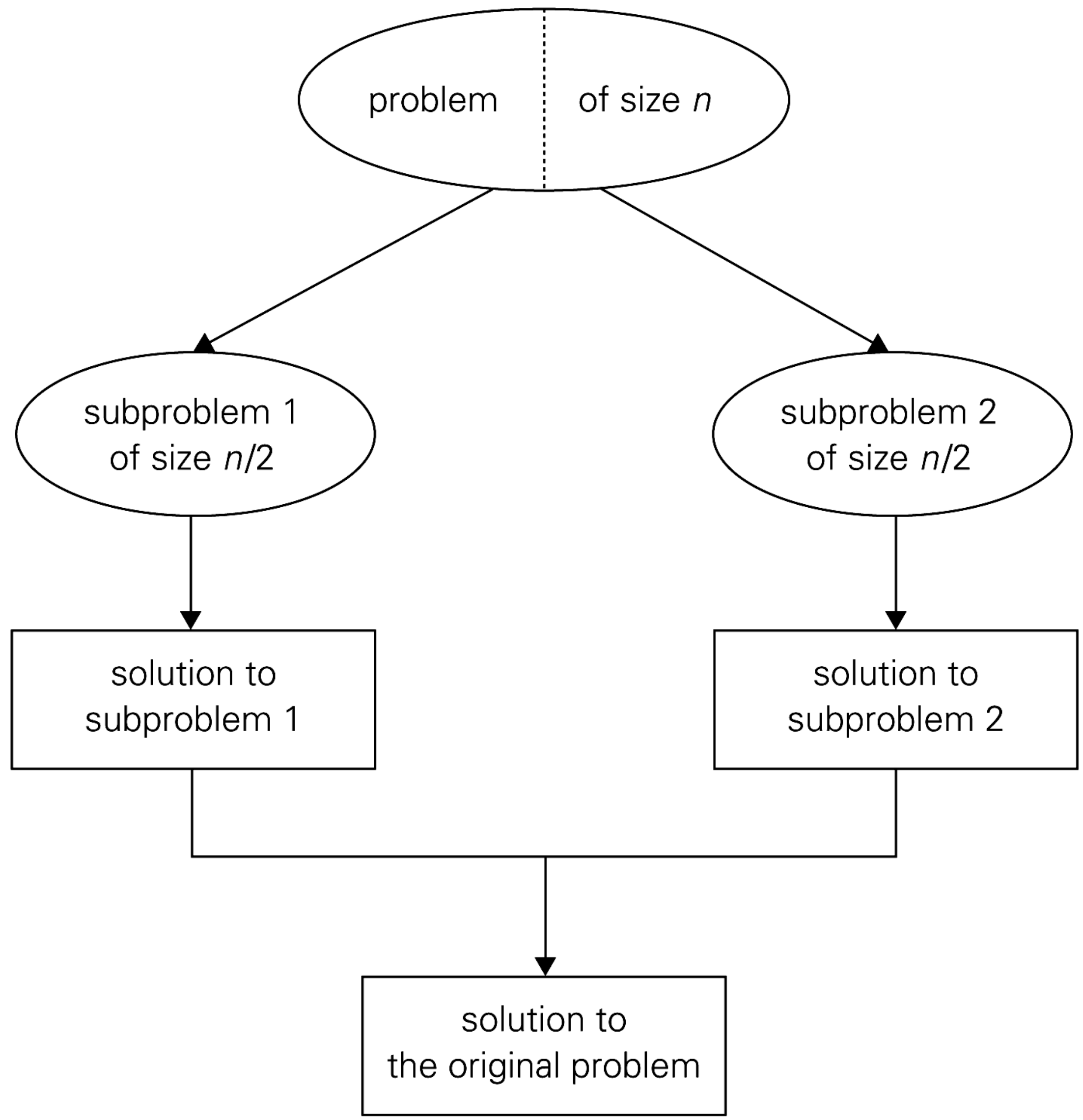


FIGURE 4.1 Divide-and-conquer technique (typical case)

Divide-and-Conquer: a case for the Master Theorem

Theorem (Master Theorem):

Let $T(n)$ be an eventually nondecreasing function that satisfies the recurrence

$$T(n) = aT(n/b) + f(n) \quad \text{for } n = b^k,\ k = 1, 2, \ldots$$
$$T(1) = c$$

where $a \geq 1$, $b \geq 2$, $c > 0$. If $f(n) \in \Theta(n^d)$ where $d \geq 0$, then

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Example: summation

Mergesort

1) Split array A[0..*n*-1] in two about equal halves and make copies of each half in arrays B and C
2) Sort arrays B and C recursively
3) Merge sorted arrays B and C into array A
   a) copy smallest element from B or C to A
   b) once B or C is processed, copy the remaining unprocessed elements from the other array into A.

ALGORITHM *Mergesort*(A[0..*n*-1])

//Sorts array A[0..*n*-1] by recursive mergesort
//Input: An array A[0..*n*-1] of orderable elements
//Output: Array A[0..*n*-1] sorted in nondecreasing order

if $n > 1$
copy A[0..$\lfloor n/2 \rfloor$-1] to B[0..$\lfloor n/2 \rfloor$-1]
copy A[$\lfloor n/2 \rfloor$..*n*-1] to C[0.. $\lceil n/2 \rceil$ -1]

Mergesort(B[0.. $\lfloor n/2 \rfloor$ -1])
Mergesort(C[0.. $\lceil n/2 \rceil$ -1])
Merge(B, C, A)

ALGORITHM *Merge*(B[0..*p*-1], C[0..*q*-1], A[0..*p*+*p*-1])

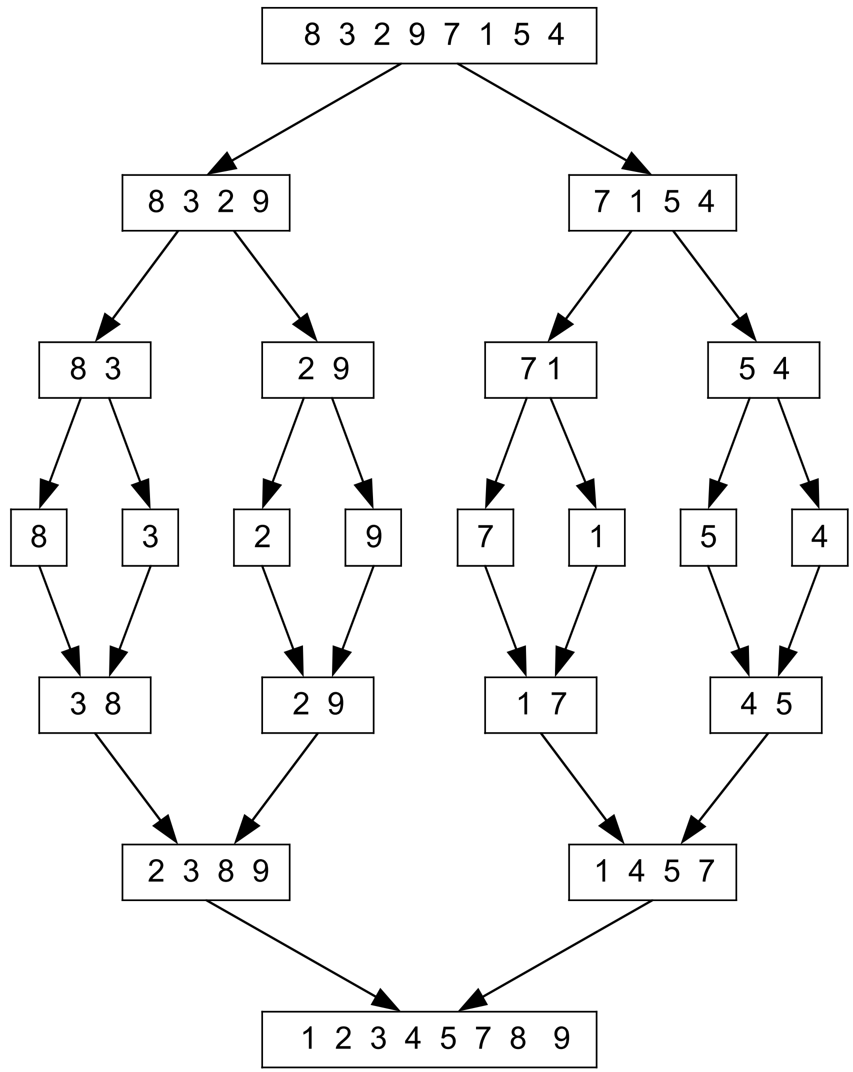
//Merges two sorted arrays into one sorted array
//Input: Arrays B[0..*p*-1] and C[0..*q*-1] both sorted
//Output: Sorted array A[0..*p*+*q*-1] of the elements of B and C

```
i ← 0; j ← 0; k ← 0
while i < p and j < q do
    if B[i] ≤ C[j]
        A[k] ← B[i]; i ← i + 1
    else
        A[k] ← C[j]; j ← j + 1
    k ← k + 1

if i = p
    copy C[j..q-1] to A[k..p+q-1]
else
    copy B[i..p-1] to A[k..p+q-1]
```

CS395: Analysis of Algorithms

Mergesort



Analysis of Mergesort

- All cases have same efficiency: $\Theta(n \log n)$
  - Side Note: Number of comparisons in the worst case is close to theoretical minimum for comparison-based sorting:

$$\lceil \log_2 n! \rceil \approx n \log_2 n - 1.44n$$

- Space requirement: $\Theta(n)$
  - version without this requirements exist, but are more costly
- Can be implemented without recursion (bottom-up)