Divide & Conquer Algorithm

Quicksort

Quicksort

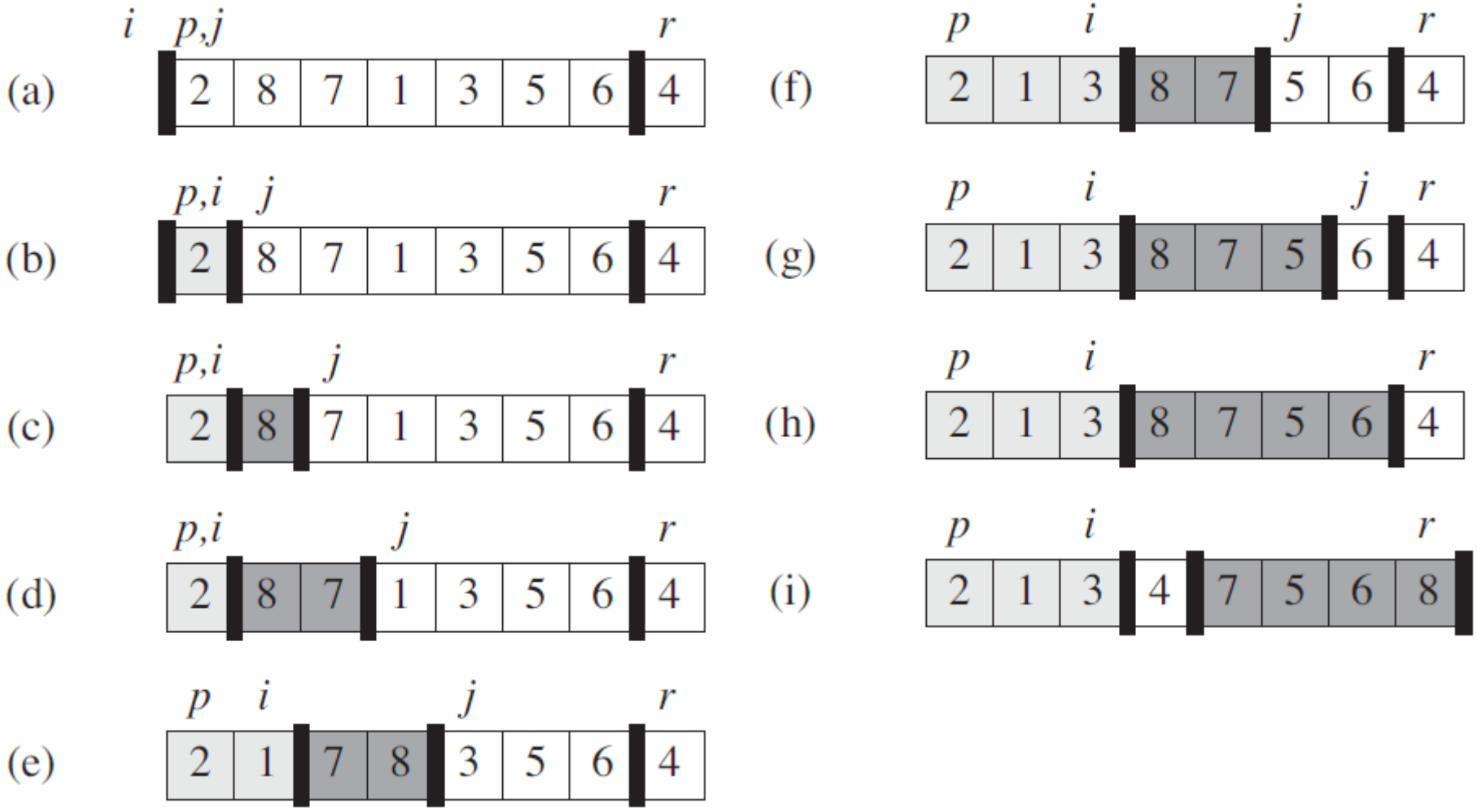
- The best practical algorithm.
- It applies the divide and conquer paradigm.
- Quicksort(A, p, r)
  1. if p < r
  2. find a pivot element at q and partition the array into three parts: left part smaller than pivot, pivot itself and right part larger than pivot.
  3. Quicksort(A, p, q – 1)
  4. Quicksort(A, q + 1, r)

Quicksort

- Partition(A, p, r)

```
x = A[r]
i = p - 1
for j = p to r - 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Quicksort


(a)
i p,j r
2 8 7 1 3 5 6 4
(b)
p,i j r
2 8 7 1 3 5 6 4
(c)
p,i j r
2 8 7 1 3 5 6 4
(d)
p,i j r
2 8 7 1 3 5 6 4
(e)
p i j r
2 1 7 8 3 5 6 4
(f)
p i j r
2 1 3 8 7 5 6 4
(g)
p i j r
2 1 3 8 7 5 6 4
(h)
p i r
2 1 3 8 7 5 6 4
(i)
p i r
2 1 3 4 7 5 6 8

Performance

- **Worst-case partitioning:**

Partitioning routine produces one subproblem with n – 1 elements and one with 0 element.

The partitioning costs $\Theta(n)$ time. Since the recursive call on an array of size 0 just returns, $T(0) = \Theta(1)$

$$T(n) = T(n-1) + T(0) + \Theta(n)$$

$$= T(n-1) + \Theta(n)$$

This recurrence has the solution $T(n) = \Theta(n^2)$.

Performance

- **Best-case partitioning:**

Partitioning routine produces two subproblems, each of size no more than n/2, since one is of size $\lfloor n/2 \rfloor$ and one of size $\lceil n/2 \rceil - 1$.

The recurrence for the running time is:

$T(n) = 2T(n/2) + \Theta(n)$

This recurrence has the solution $T(n) = \Theta(n \lg n)$.