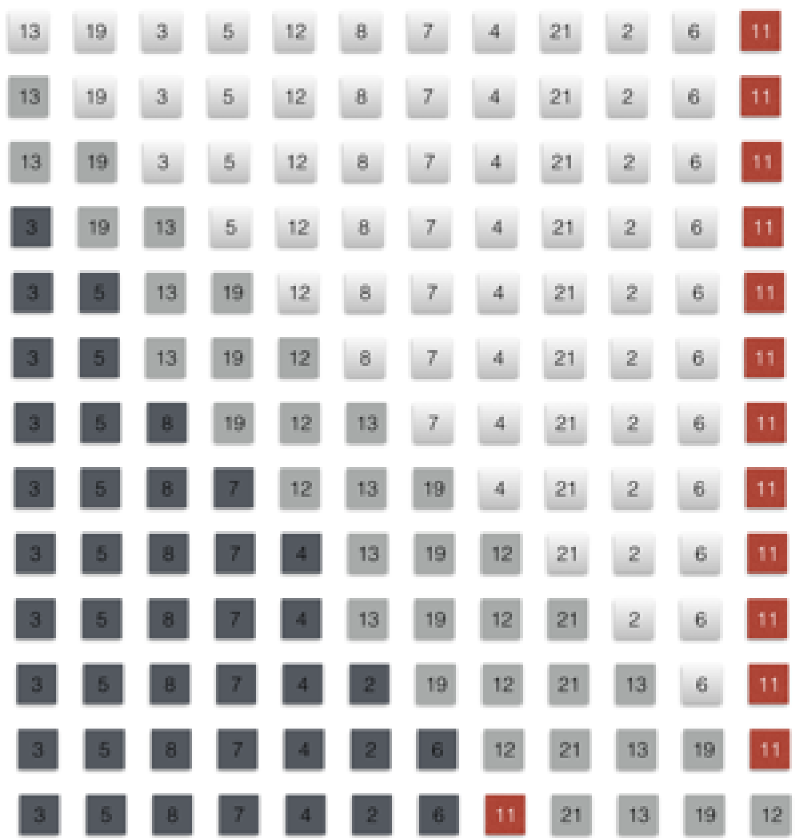
Solution 5: Advanced Sorting Algorithms

Answer 1

```
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
```

Using Figure 7.1 as a model, illustrate the operation of PARTITION on the array $A = [13, 19, 3, 5, 12, 8, 7, 4, 21, 2, 6, 11]$.



Answer 2

The algorithm will begin by preprocessing exactly as COUNTING-SORT does in lines 1 through 9, so that $C[i]$ contains the number of elements less than or equal to i in the array. When queried about how many integers fall into the range $[a..b]$, simply compute $C[b] - C[a-1]$. This takes $O(1)$ times and yields the desired output.

```
Preprocess(A, a, b):
    let C[0..k] be a new array
    for i = 0 to k
        C[i]=0
    for j = 1 to A.length
        C[A[j]] = C[A[j]] + 1
    // C[i] now contains the number of elements equal to i.
    for i = 1 to k
        C[i] = C[i] + C[i-1]
    // C[i] now contains the number of elements less than or equal to i.
    return C
```

The time complexity of the preprocess procedure is Θ(n+k).

```
Query(C, a, b):
    return C[b]-C[a-1]
```

The time complexity of the query procedure is O(1).

Answer 3

Starting with the unsorted words on the left, and stable sorting by progressively more important positions.

0	1	2	3
COW	SEA	SEA	BOX
DOG	MOB	MOB	COW
SEA	DOG	DOG	DOG
RUG	RUG	COW	MOB
ROW	COW	ROW	ROW
MOB	ROW	BOX	RUG
BOX	BOX	RUG	SEA