Data Structures & Algorithms

20: Merge Sort

(Divide & Conquer)

Dr Ram Prasad Krishnamoorthy

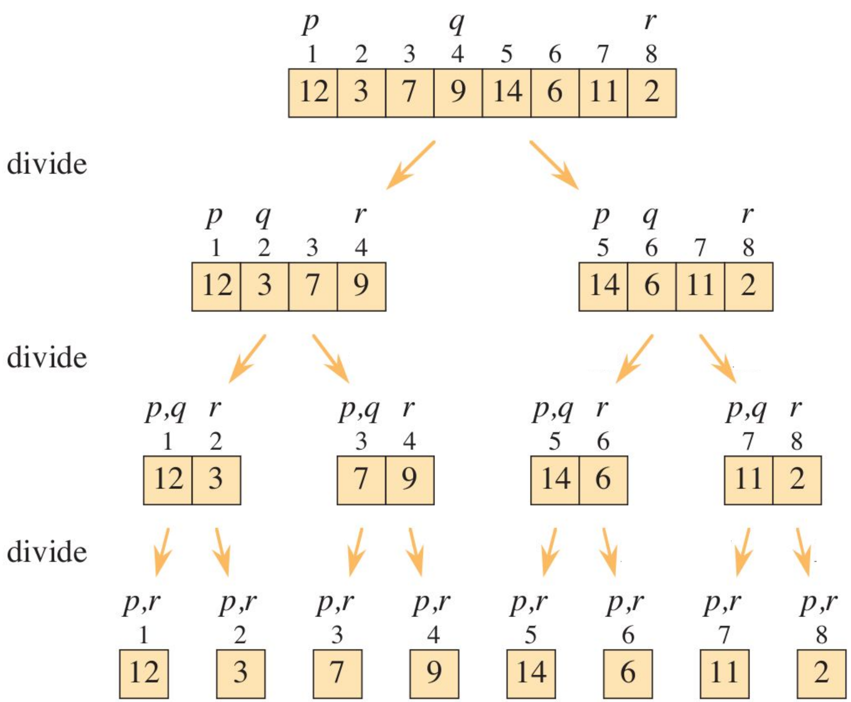
Associate Professor
School of Computing and Data Science

ram.krish@saiuniversity.edu.in

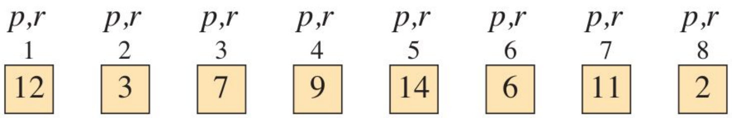
SAI UNIVERSITY

Merge Sort

Merge Sort



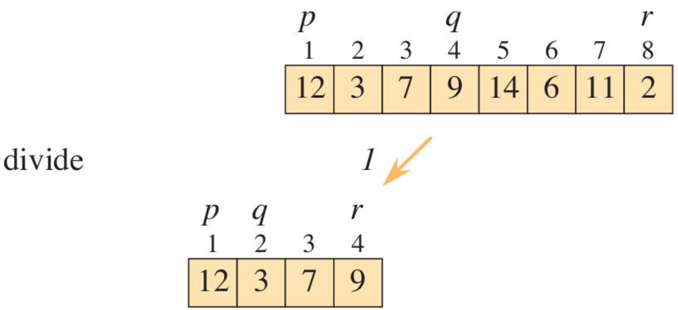
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
p,q r
7 8
11 2
divide
p,r 1 12
p,r 2 3
p,r 3 7
p,r 4 9
p,r 5 14
p,r 6 6
p,r 7 11
p,r 8 2

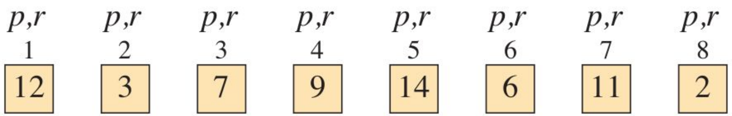


p,r 1 12
p,r 2 3
p,r 3 7
p,r 4 9
p,r 5 14
p,r 6 6
p,r 7 11
p,r 8 2

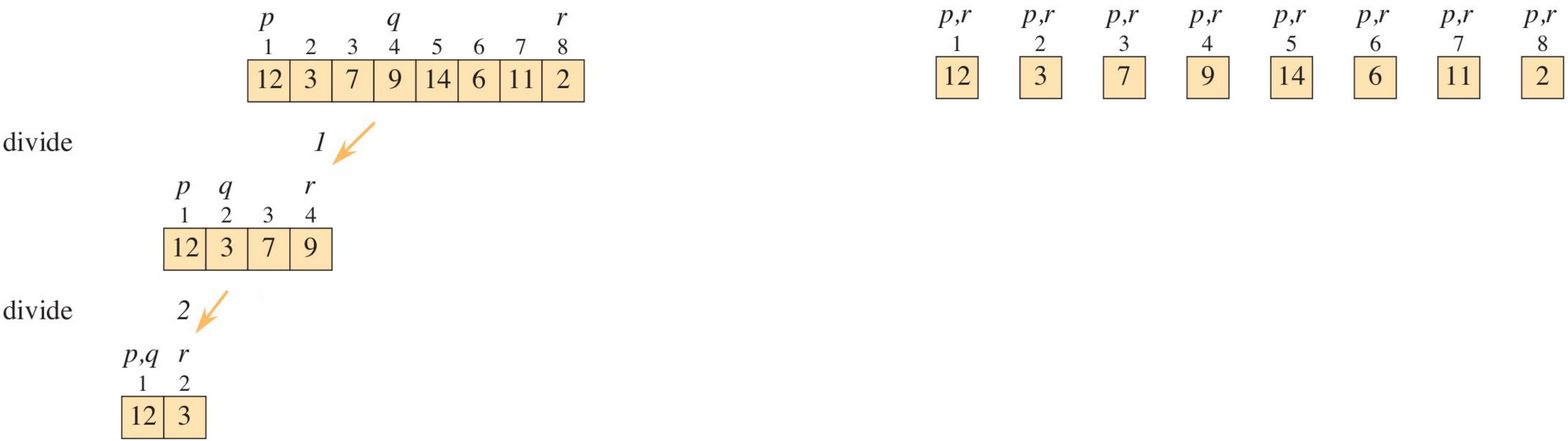
Splitting is done Recursively

Merge Sort

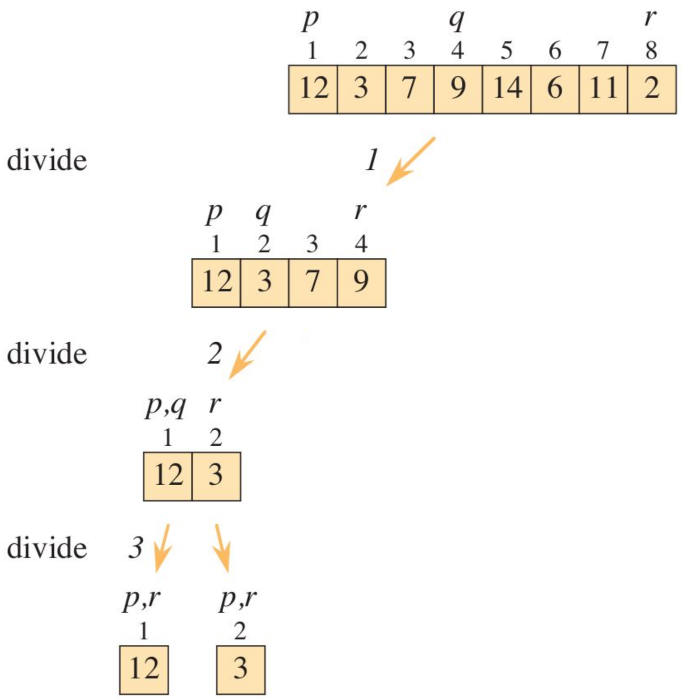

p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9

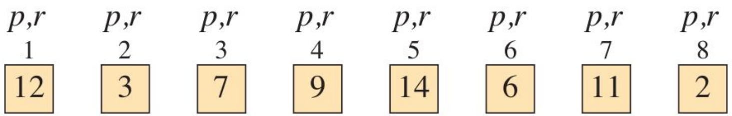

p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2

Merge Sort

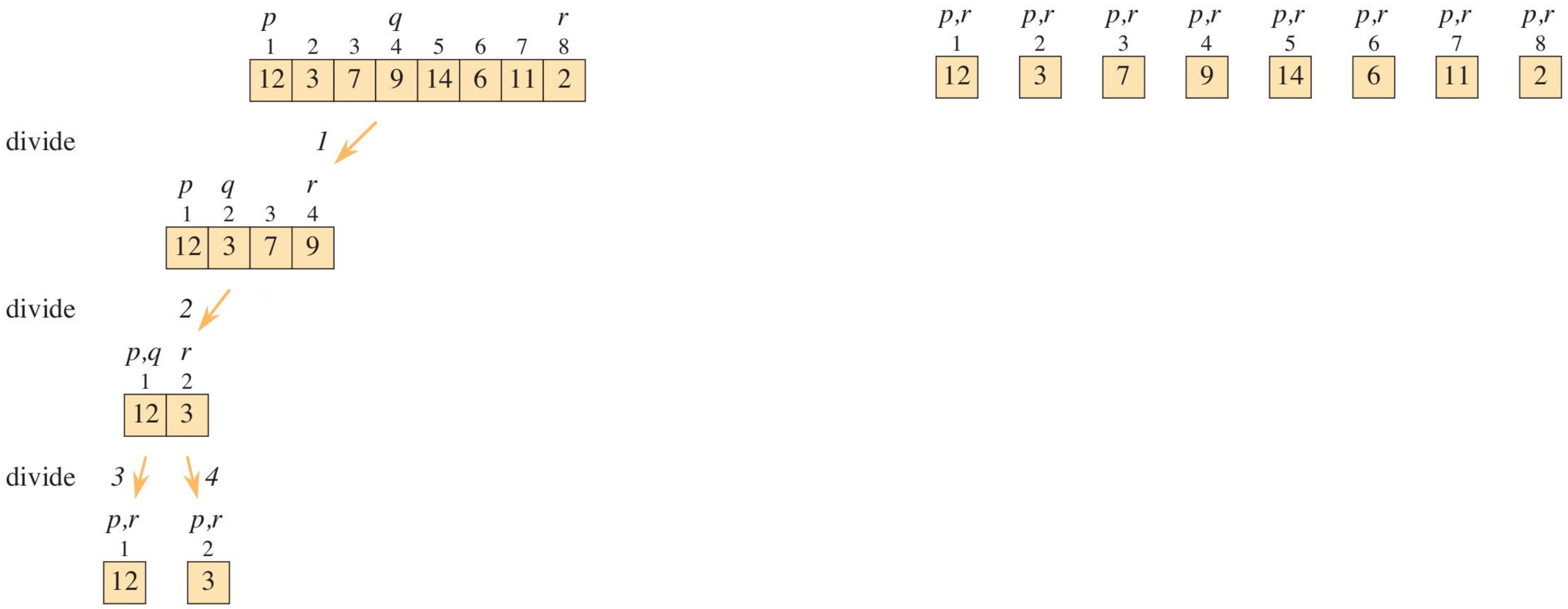

p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide 1
p q r
1 2 3 4
12 3 7 9
divide 2
p,q r
1 2
12 3
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2

Merge Sort

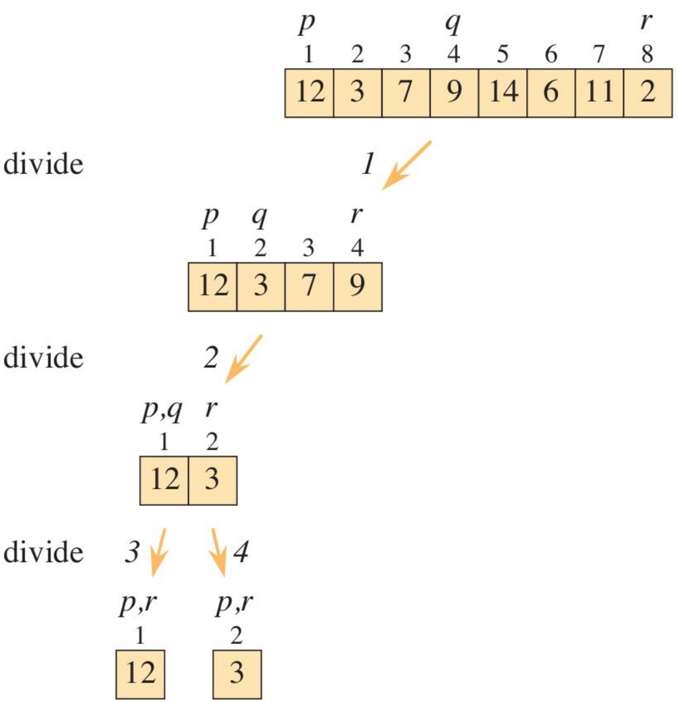

p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide 1
p q r
1 2 3 4
12 3 7 9
divide 2
p,q r
1 2
12 3
divide 3
p,r p,r
1 2
12 3

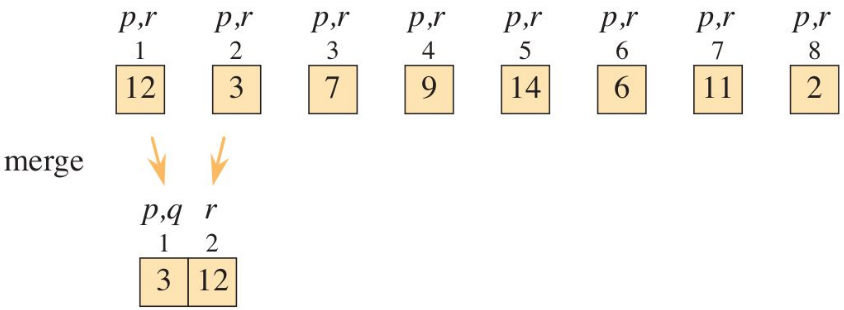

p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2

Merge Sort

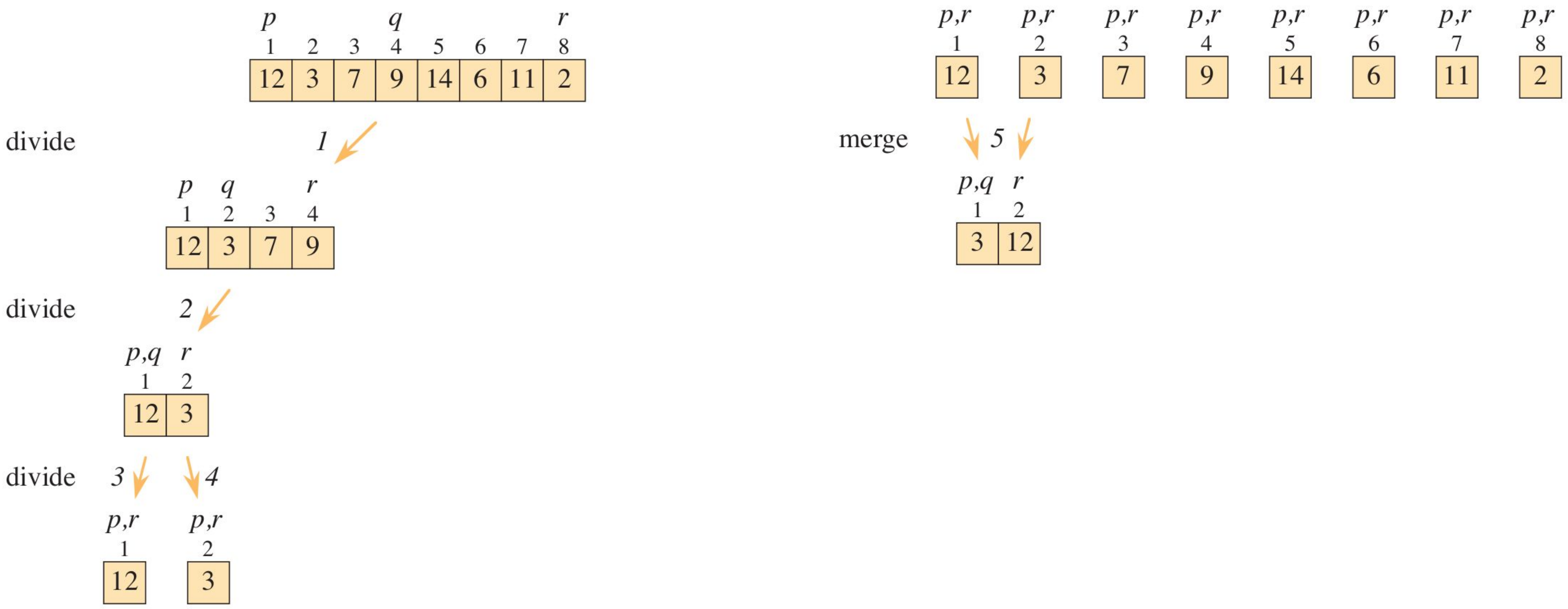

p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide 1
p q r
1 2 3 4
12 3 7 9
divide 2
p,q r
1 2
12 3
divide 3 4
p,r p,r
1 2
12 3
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2

Merge Sort


p
q
r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2
p,q r
1 2
12 3
divide
3
4
p,r
1
12
p,r
2
3

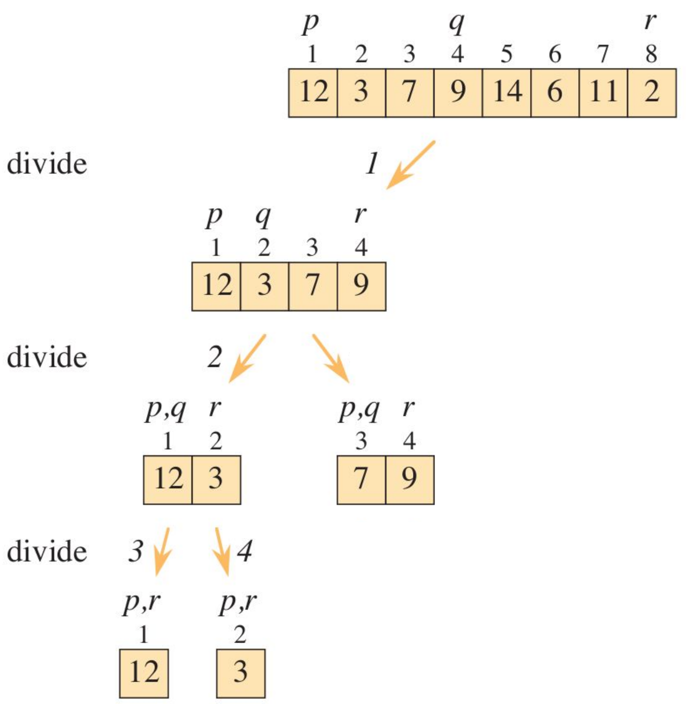

p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
p,q r
1 2
3 12

Merge Sort

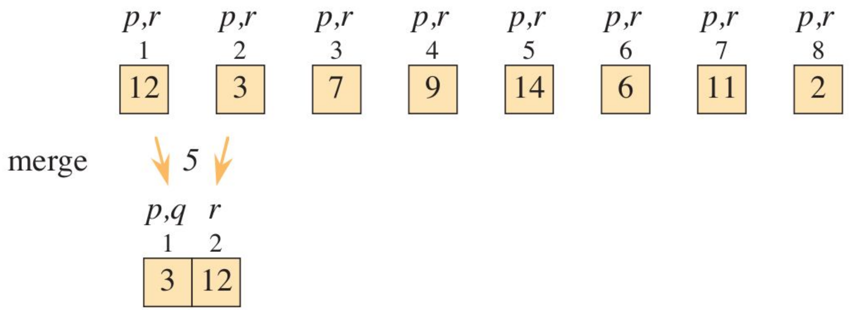


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide 1
p q r
1 2 3 4
12 3 7 9
divide 2
p,q r
1 2
12 3
divide 3 4
p,r p,r
1 2
12 3
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge 5
p,q r
1 2
3 12

Merge Sort

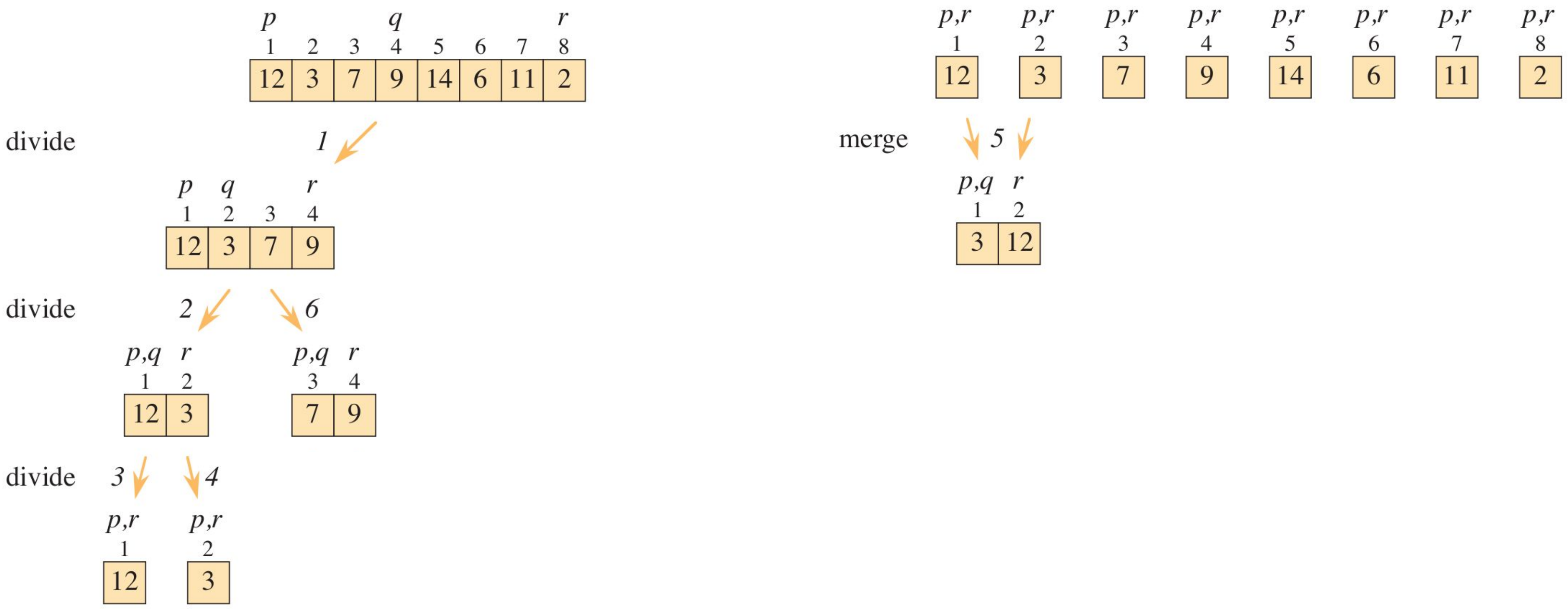


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4
p,r
1
12
p,r
2
3

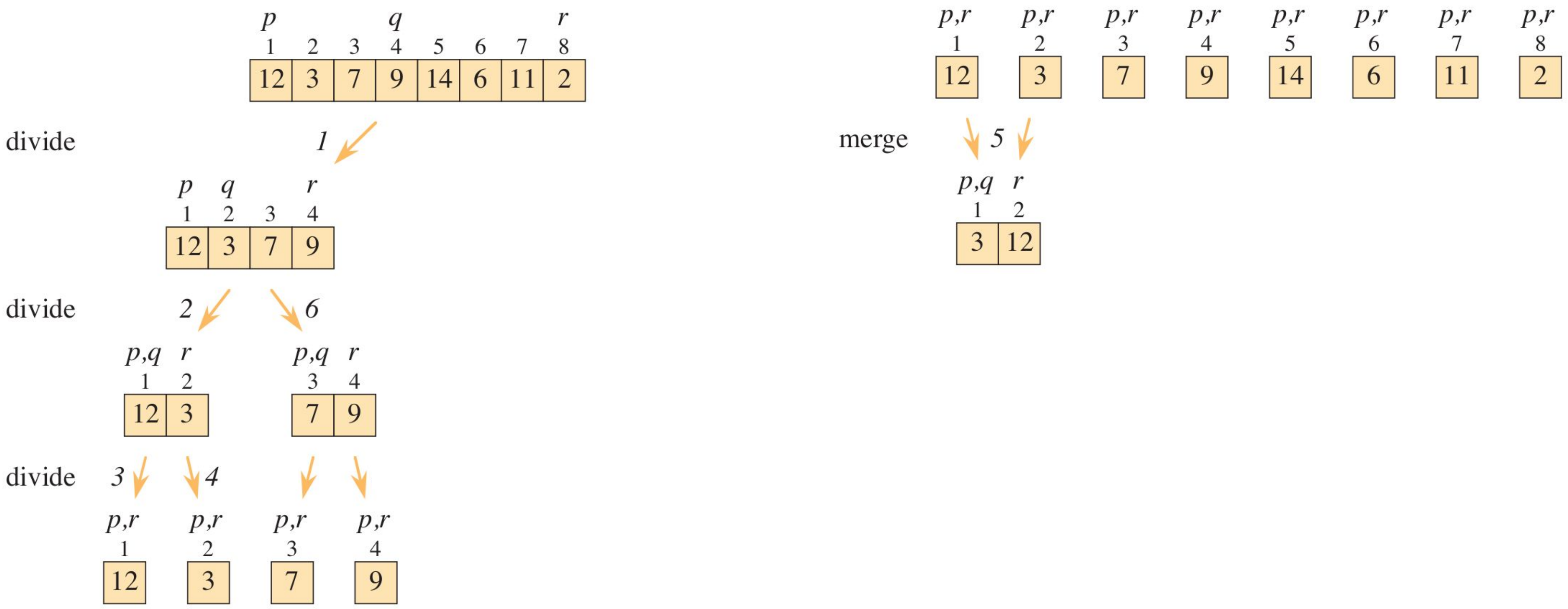


p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
p,q r
1 2
3 12

MERGE SORT

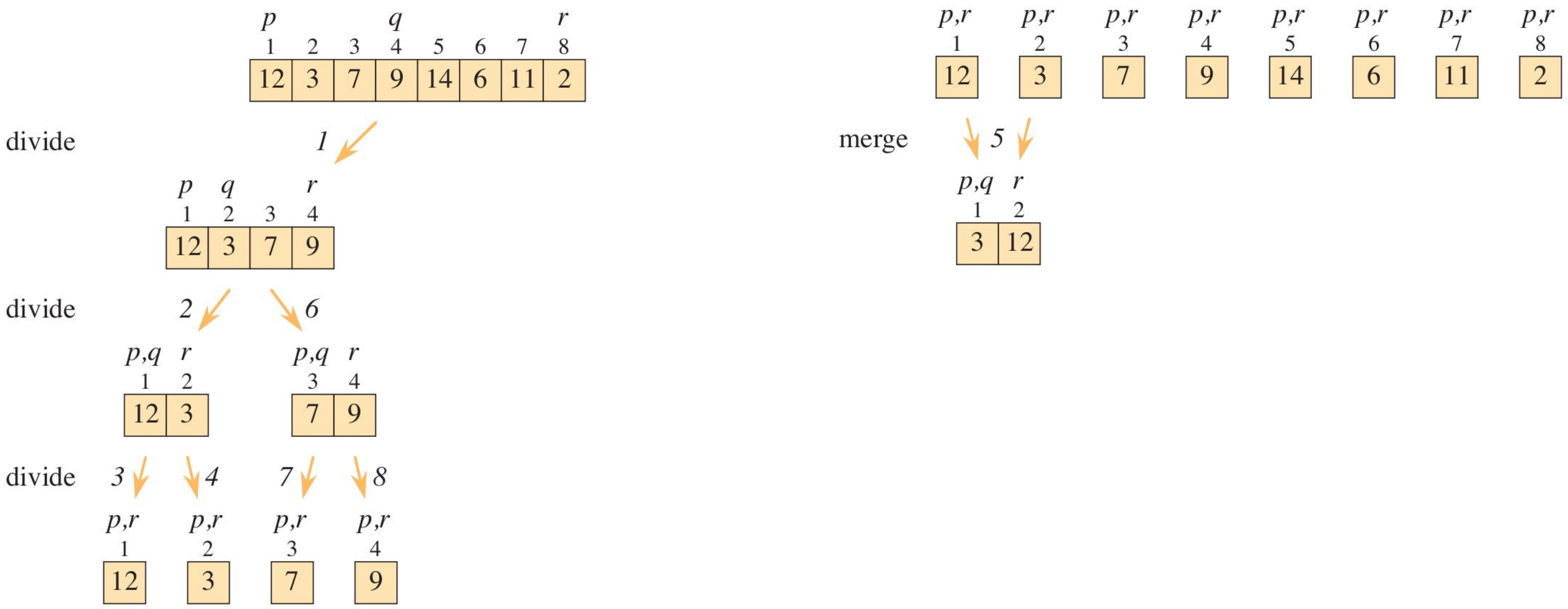

divide
divide
divide
merge
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
1
p q r
1 2 3 4
12 3 7 9
2
6
p,q r
1 2
12 3
p,q r
3 4
7 9
3
4
p,r
1
12
p,r
2
3
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
5
p,q r
1 2
3 12

MERGE SORT

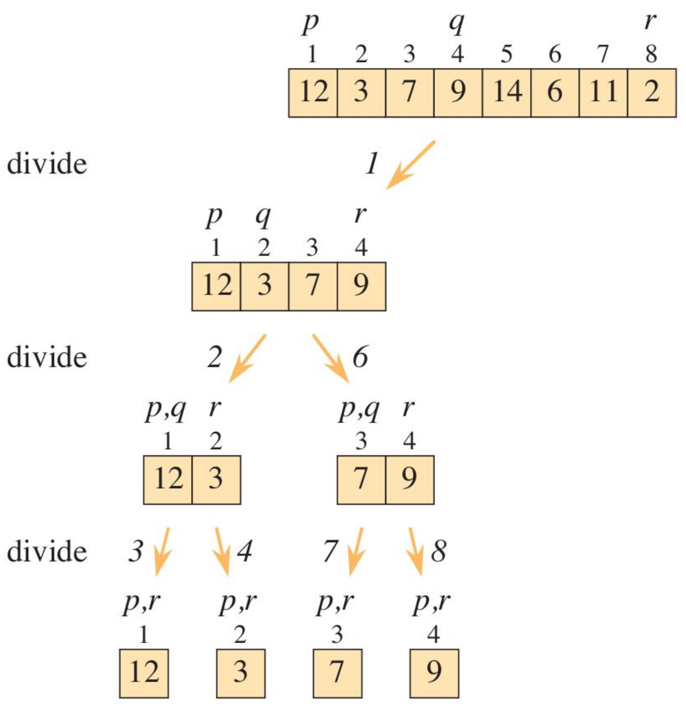


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2 6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4
p,r p,r p,r p,r
1 2 3 4
12 3 7 9
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
p,q r
1 2
3 12

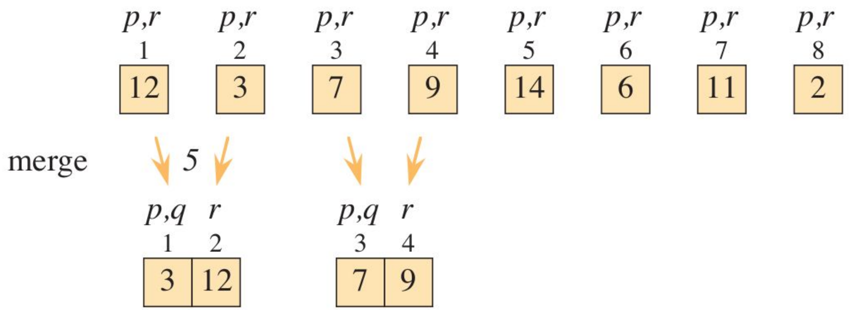
Merge Sort



Merge Sort

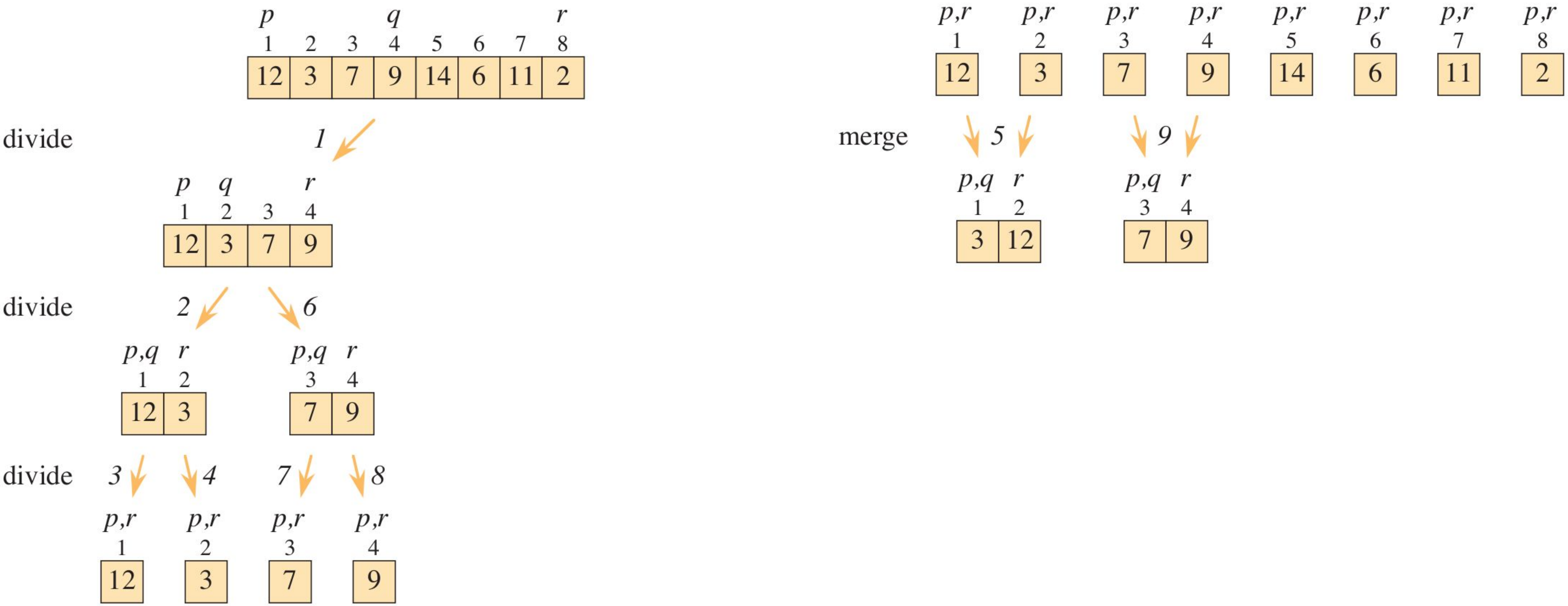


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2
6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3
4
7
8
p,r
1
12
p,r
2
3
p,r
3
7
p,r
4
9



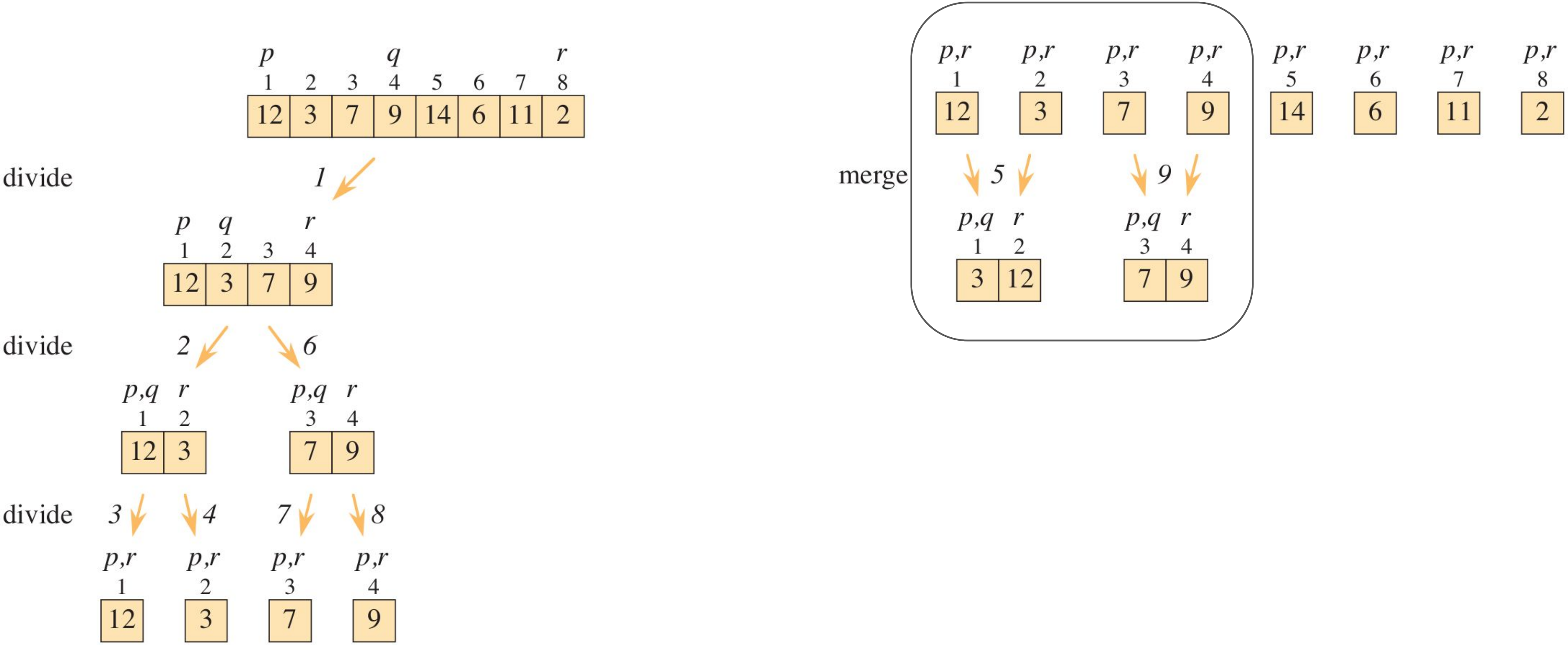
p,r
1
12
p,r
2
3
p,r
3
7
p,r
4
9
p,r
5
14
p,r
6
6
p,r
7
11
p,r
8
2
merge
5
p,q r
1 2
3 12
p,q r
3 4
7 9

Merge Sort



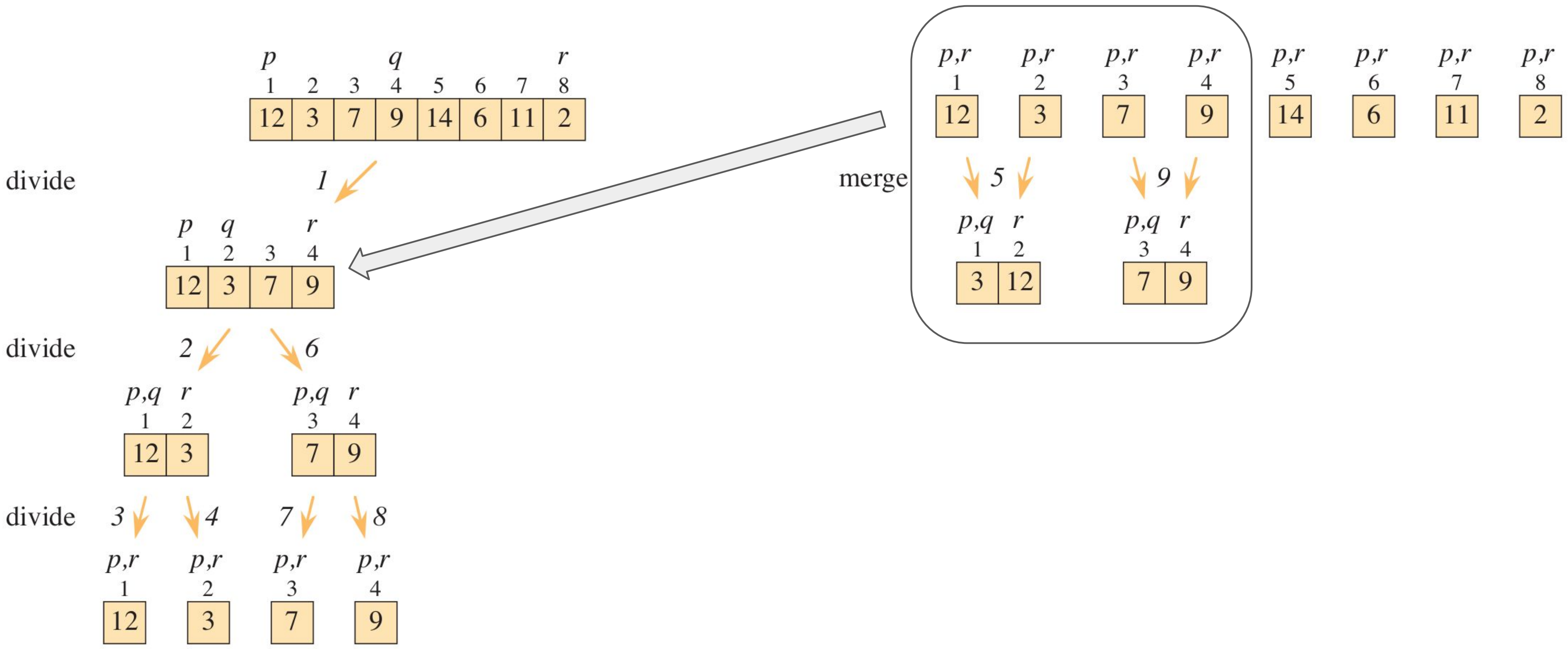
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2 6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4 7 8
p,r p,r p,r p,r
1 2 3 4
12 3 7 9
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5 9
p,q r
1 2
3 12
p,q r
3 4
7 9

Merge Sort



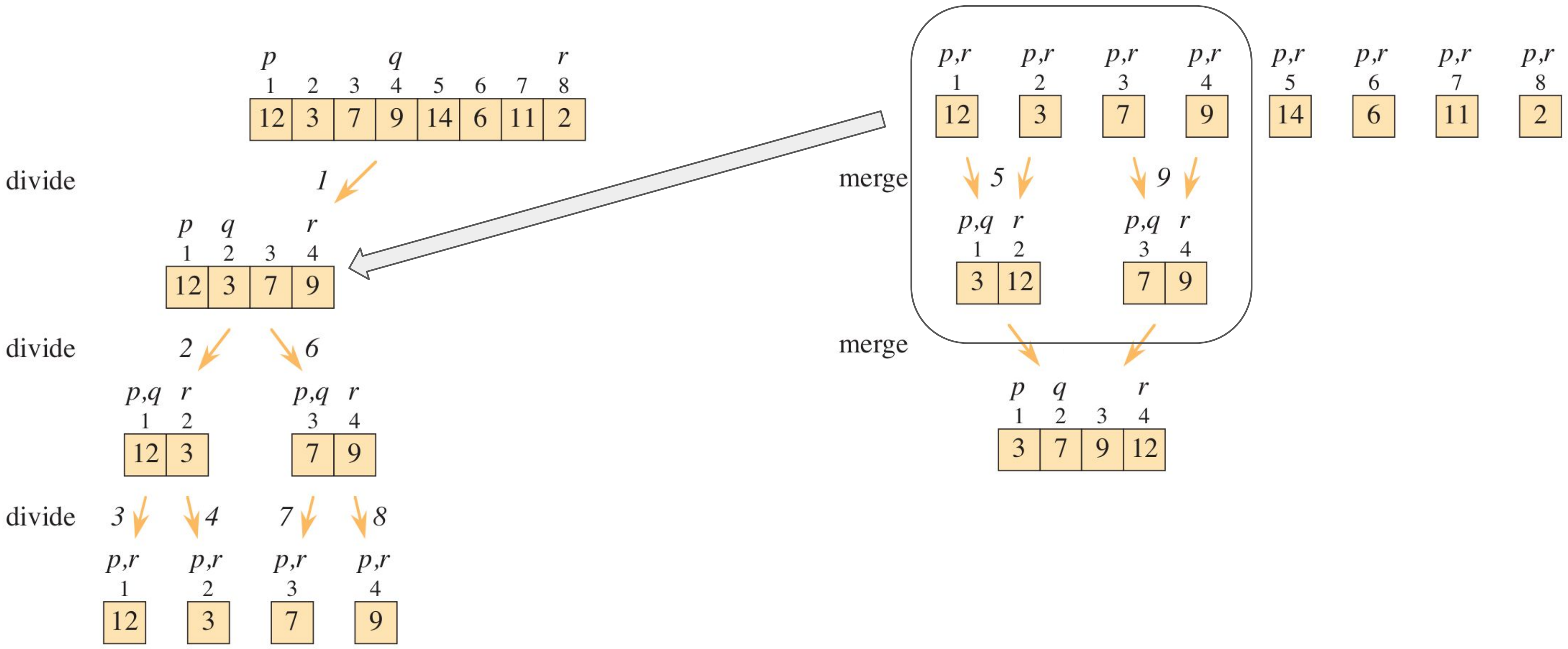
divide
divide
divide
merge
p
q
r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
1
2
6
3
4
7
8
5
9
p,q
r
p,r

Merge Sort

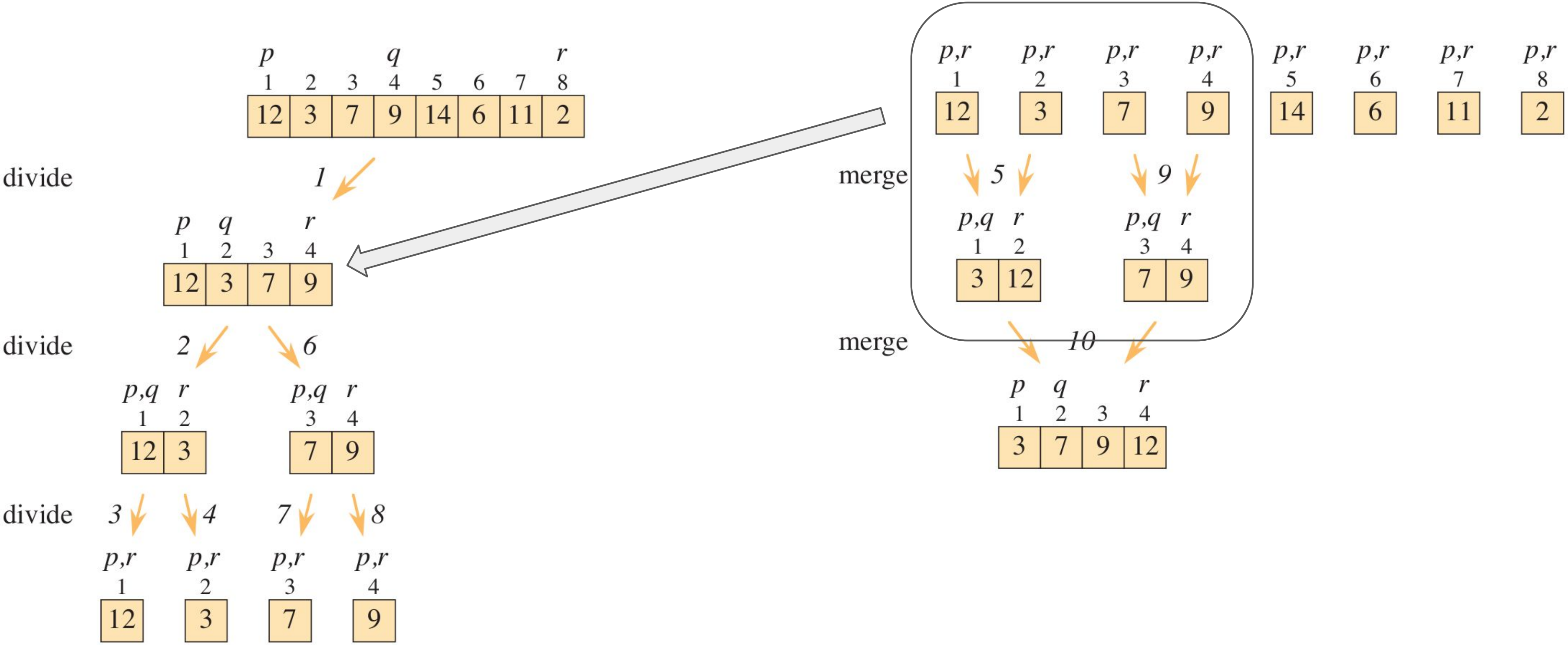


divide
divide
divide
merge
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
1 2 3 4 5 6 7 8 9
p,q r
p,r

Merge Sort

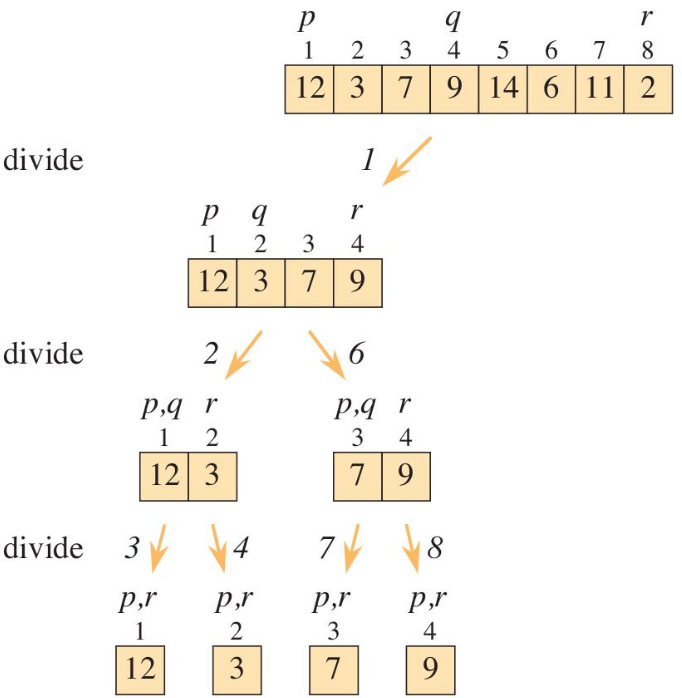

p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2 6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4 7 8
p,r p,r p,r p,r
1 2 3 4
12 3 7 9
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5 9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
p q r
1 2 3 4
3 7 9 12

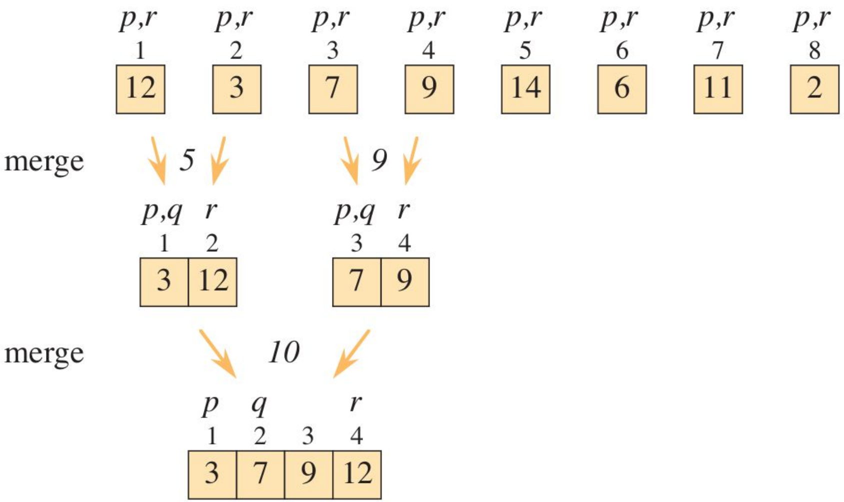
Merge Sort



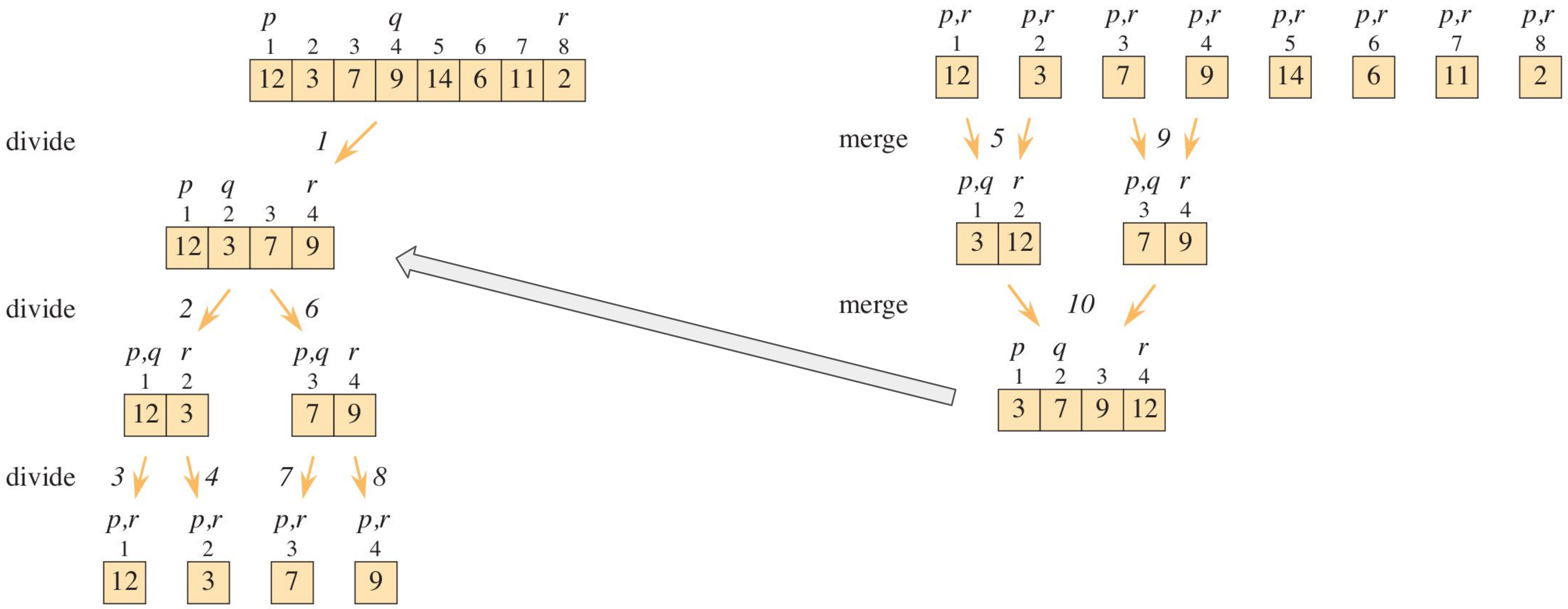
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2
6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4 7 8
p,r p,r p,r p,r
1 2 3 4
12 3 7 9
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2
6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3
4
7
8
p,r p,r p,r p,r
1 2 3 4
12 3 7 9

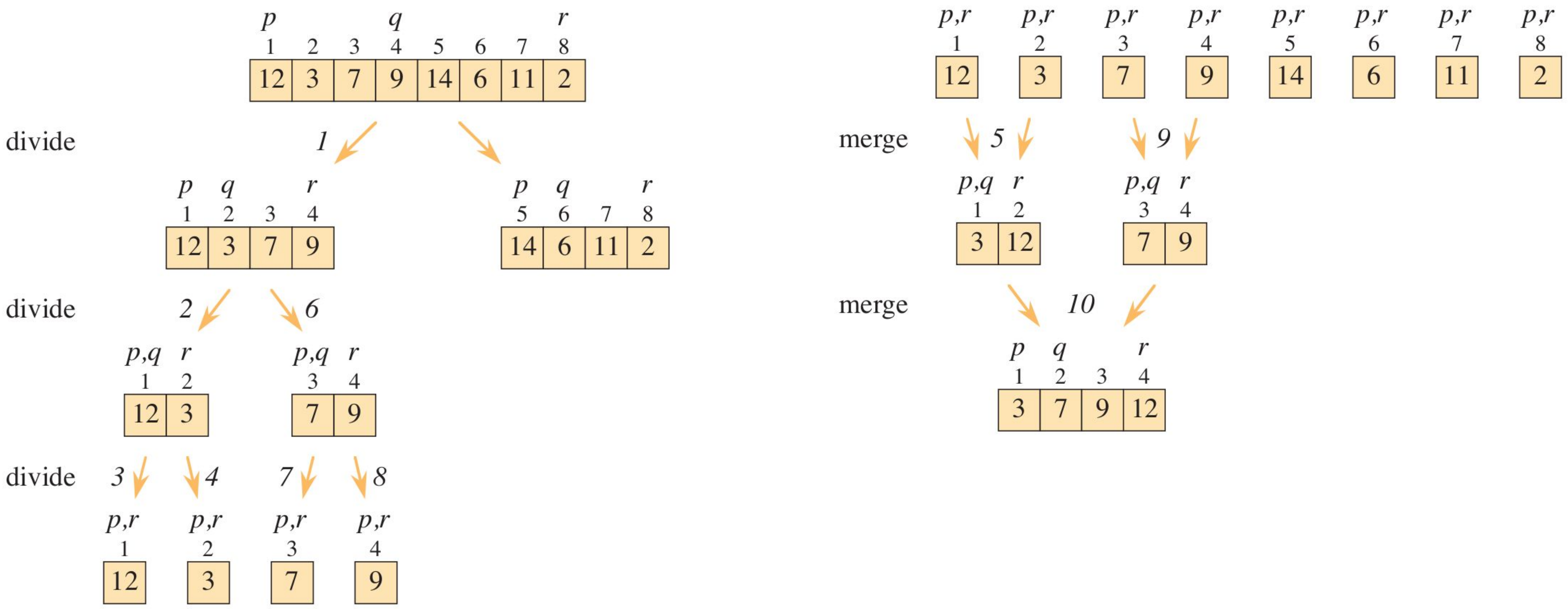

p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort



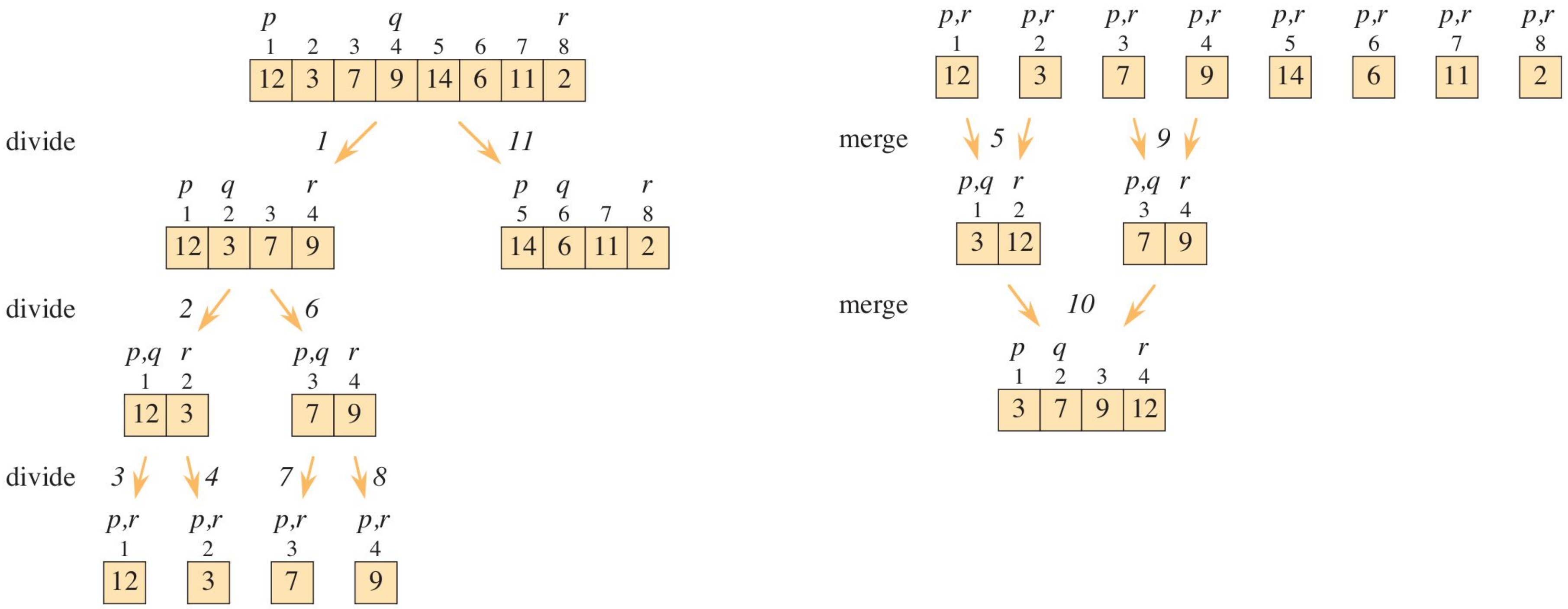
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
divide
2 6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4 7 8
p,r p,r p,r p,r
1 2 3 4
12 3 7 9
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5 9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort

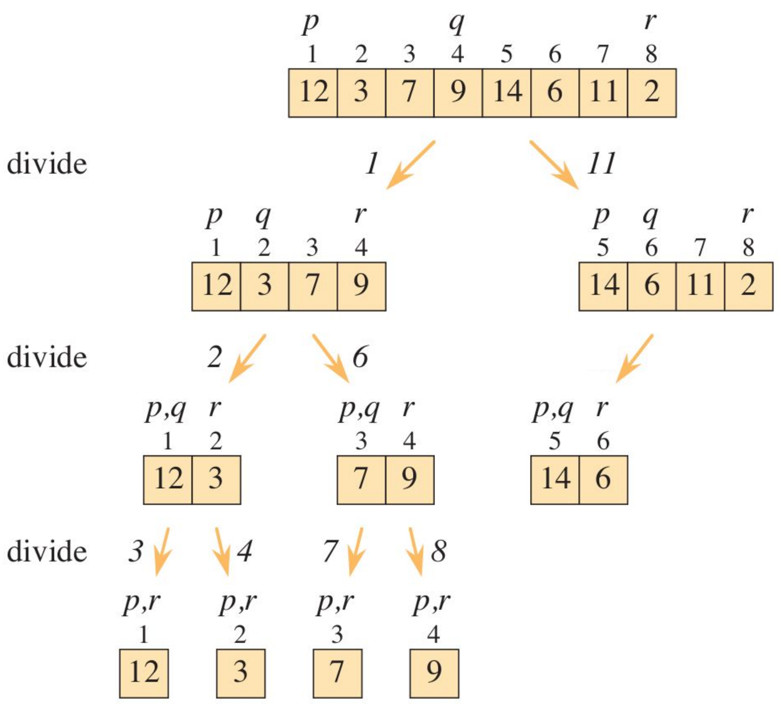


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2 6
p,q r
1 2
12 3
p,q r
3 4
7 9
divide
3 4 7 8
p,r p,r p,r p,r
1 2 3 4
12 3 7 9
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5 9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
10
p q r
1 2 3 4
3 7 9 12

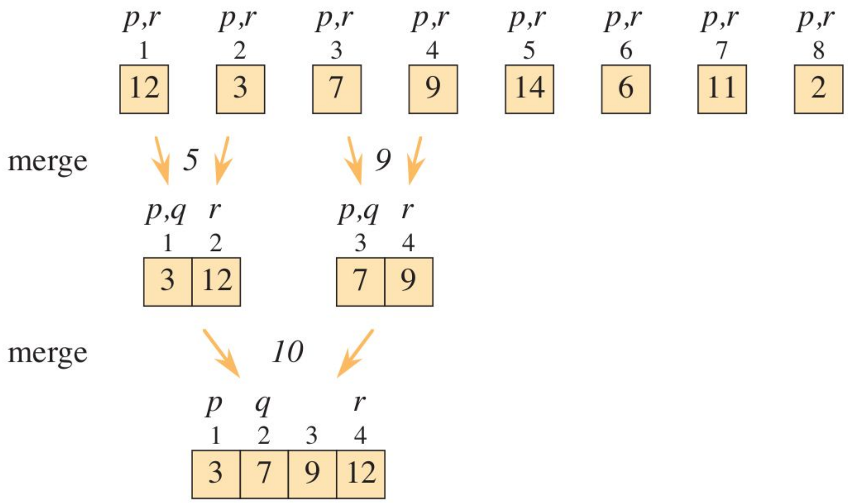
Merge Sort



Merge Sort

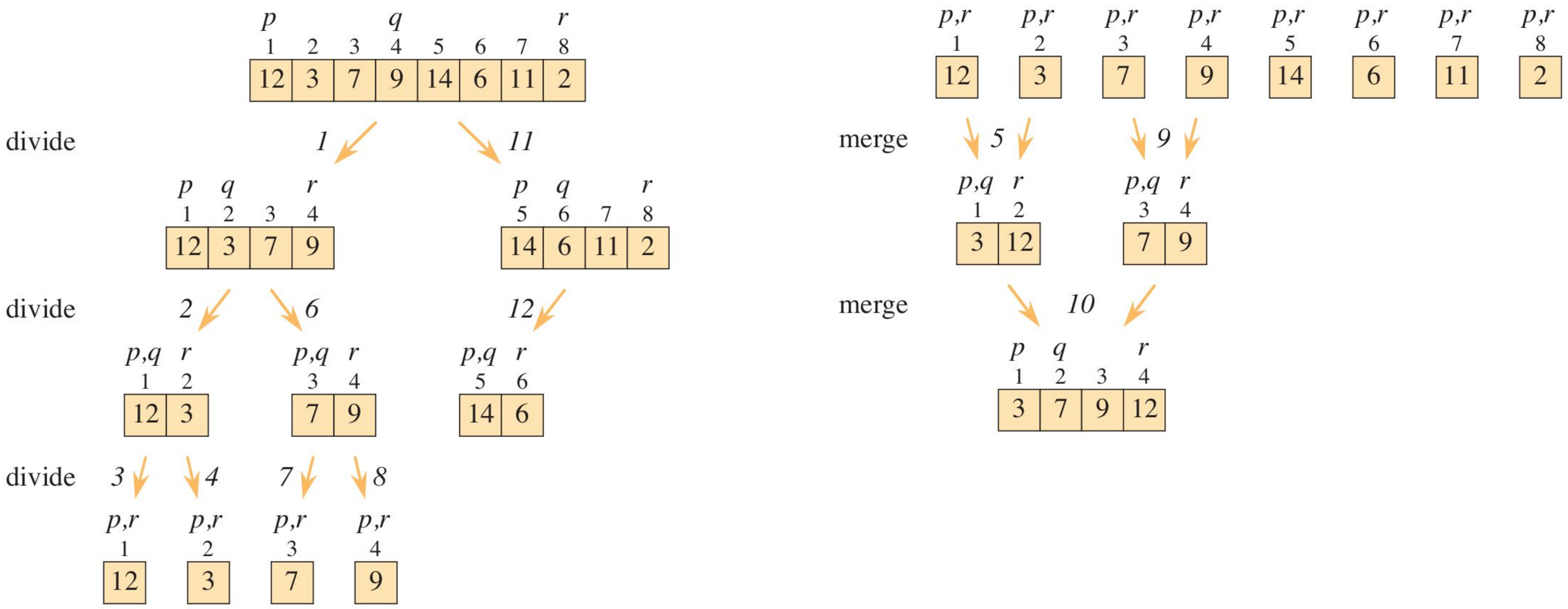


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
divide
3
4
7
8
p,r
1
12
p,r
2
3
p,r
3
7
p,r
4
9

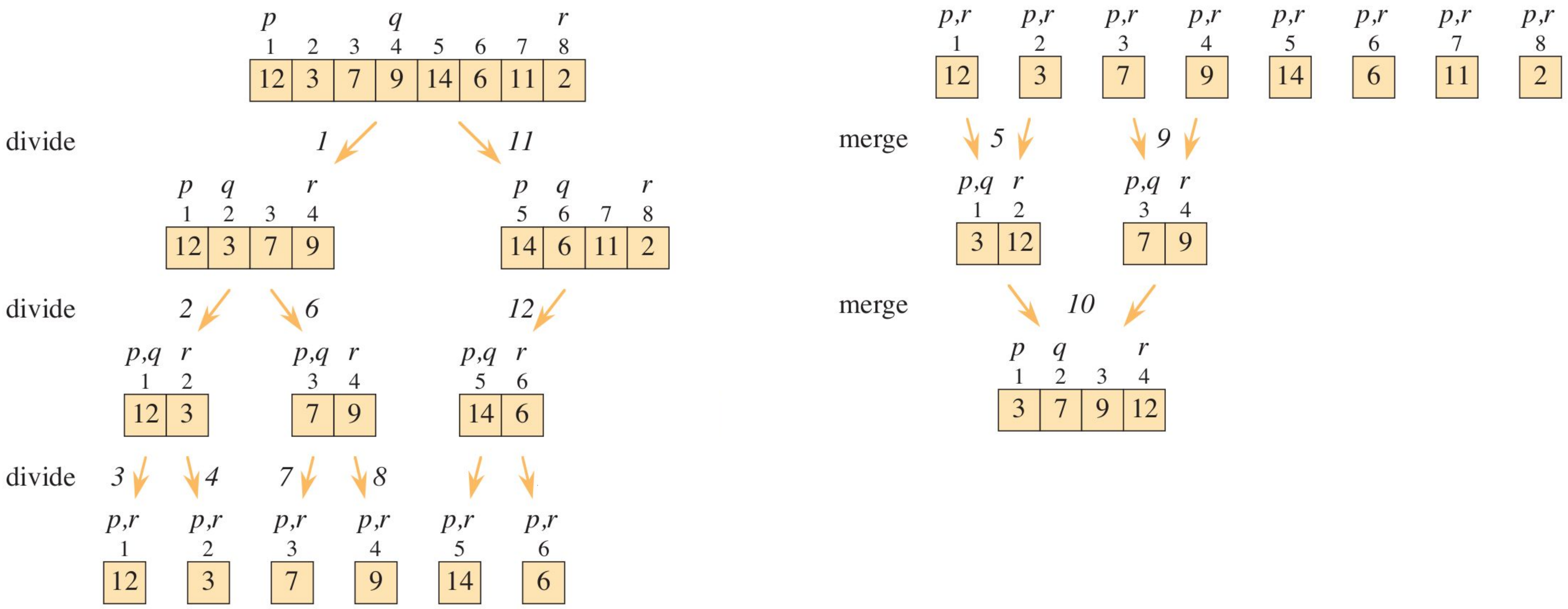


p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort

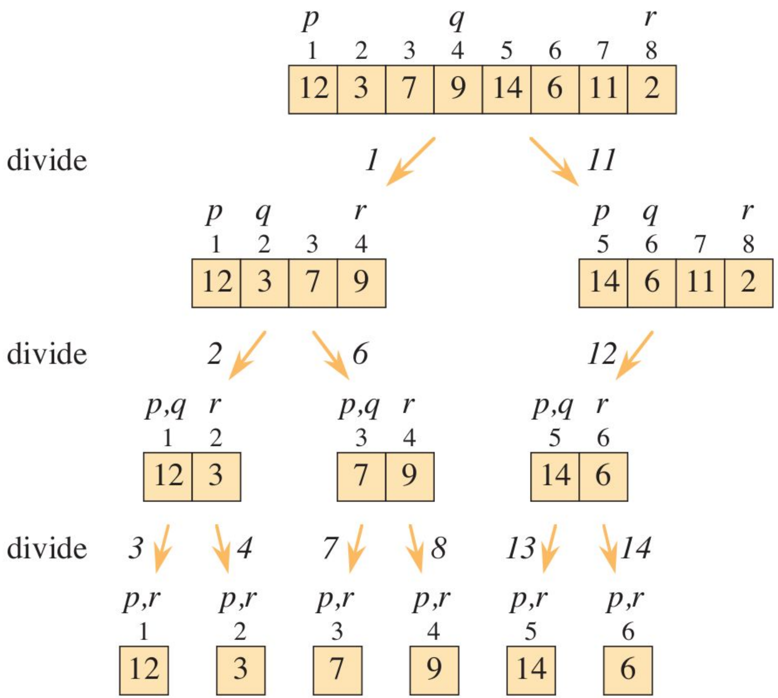


Merge Sort

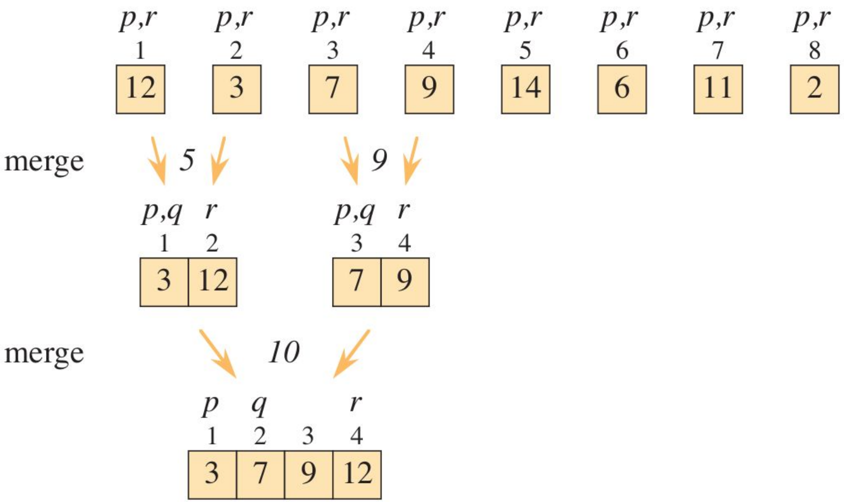


divide
divide
divide
merge
merge
12 3 7 9 14 6 11 2
12 3 7 9
14 6 11 2
3 12
7 9
3 7 9 12

Merge Sort

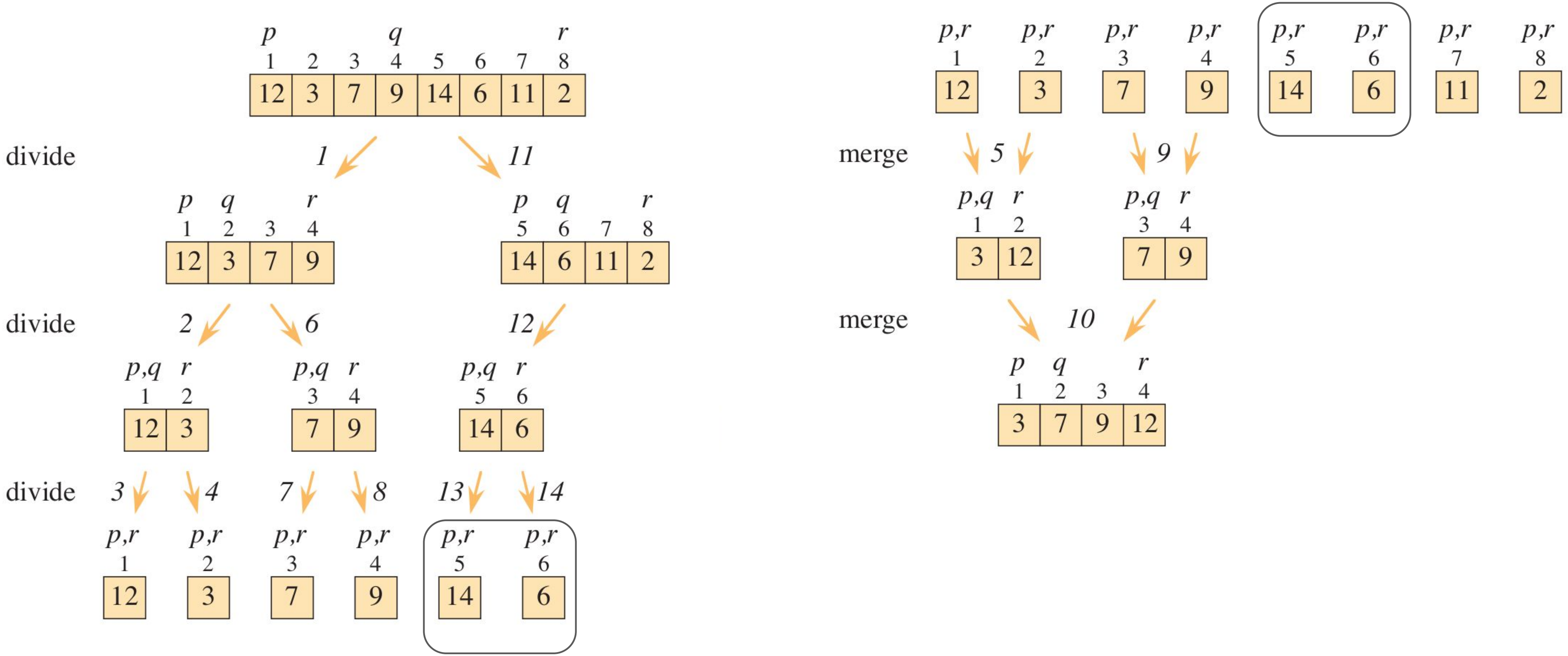


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
12
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
divide
3 4 7 8 13 14
p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6
12 3 7 9 14 6

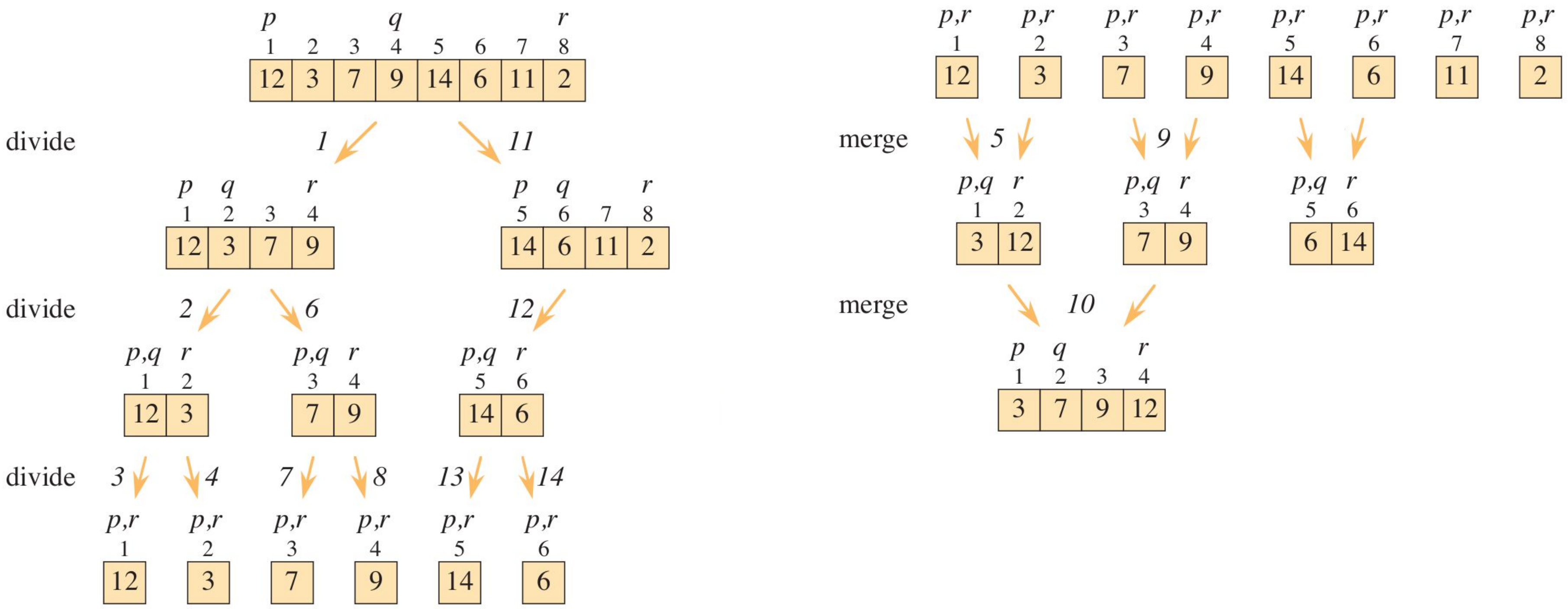


p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
p,q r
1 2
3 12
p,q r
3 4
7 9
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort

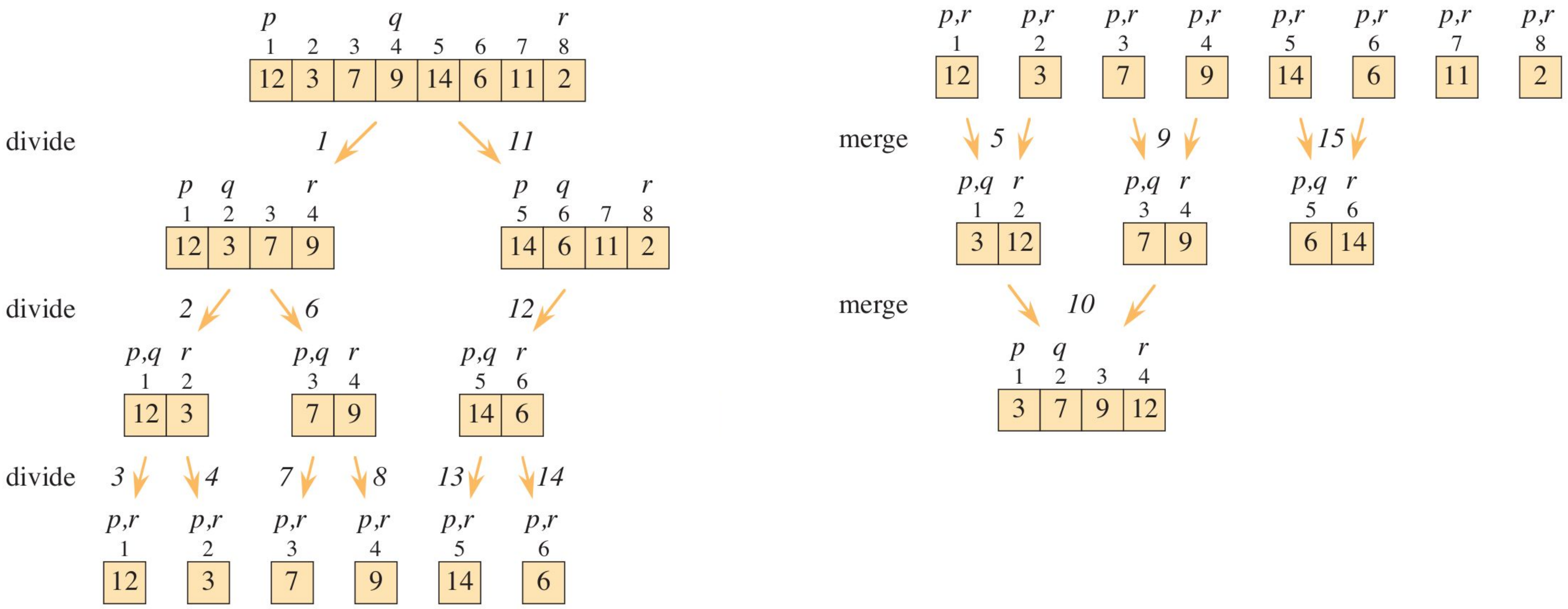

divide
divide
divide
merge
merge
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
2
6
12
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
3 4 7 8 13 14
p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6
12 3 7 9 14 6
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
5
9
p,q r
1 2
3 12
p,q r
3 4
7 9
10
p q r
1 2 3 4
3 7 9 12

Merge Sort



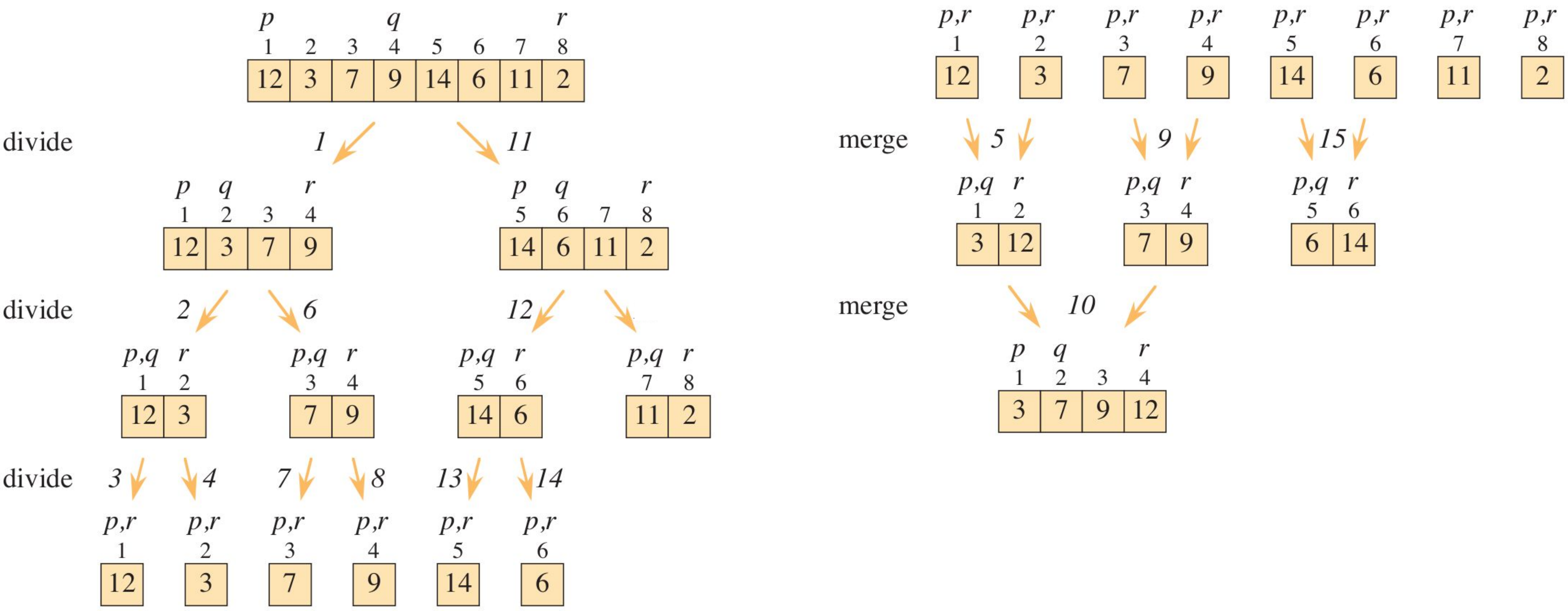
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
12
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
divide
3 4 7 8 13 14
p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6
12 3 7 9 14 6
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
p,q r
1 2
3 12
p,q r
3 4
7 9
p,q r
5 6
6 14
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort



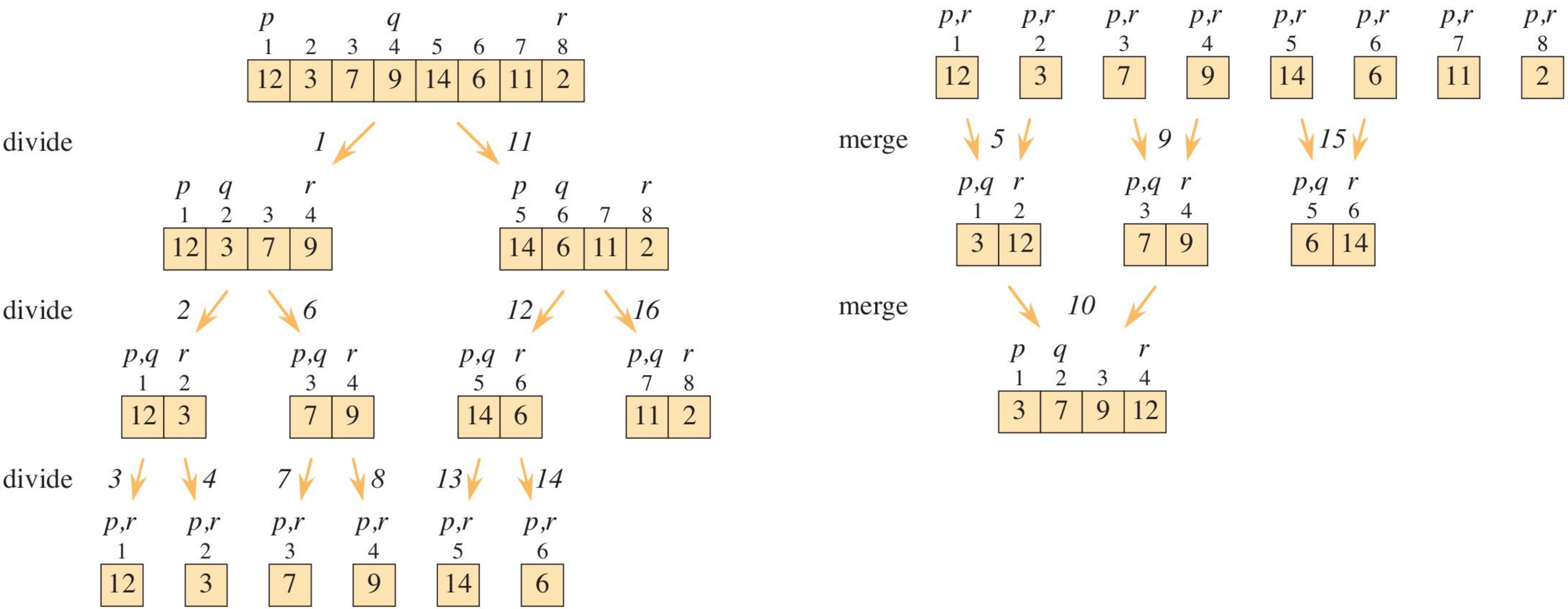
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
12
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
divide
3 4 7 8 13 14
p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6
12 3 7 9 14 6
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
15
p,q r
1 2
3 12
p,q r
3 4
7 9
p,q r
5 6
6 14
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort



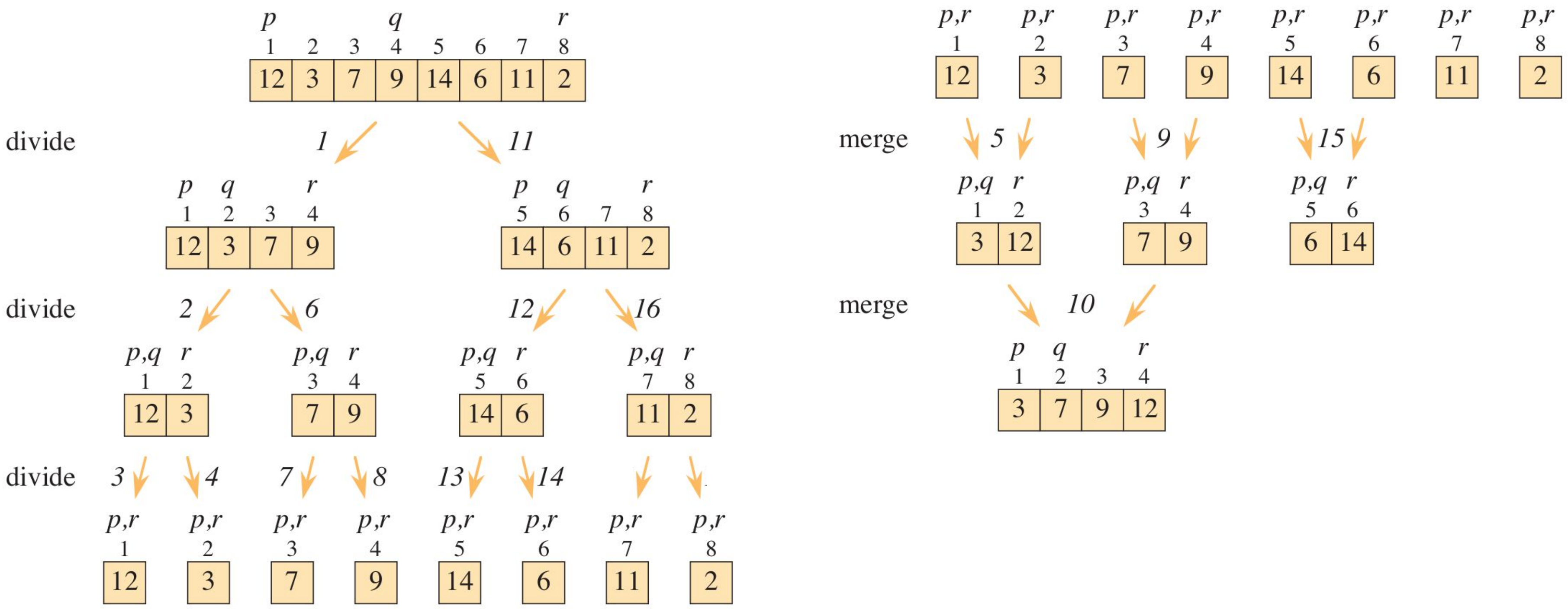
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
12
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
p,q r
7 8
11 2
divide
3 4 7 8 13 14
p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6
12 3 7 9 14 6
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
15
p,q r
1 2
3 12
p,q r
3 4
7 9
p,q r
5 6
6 14
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort



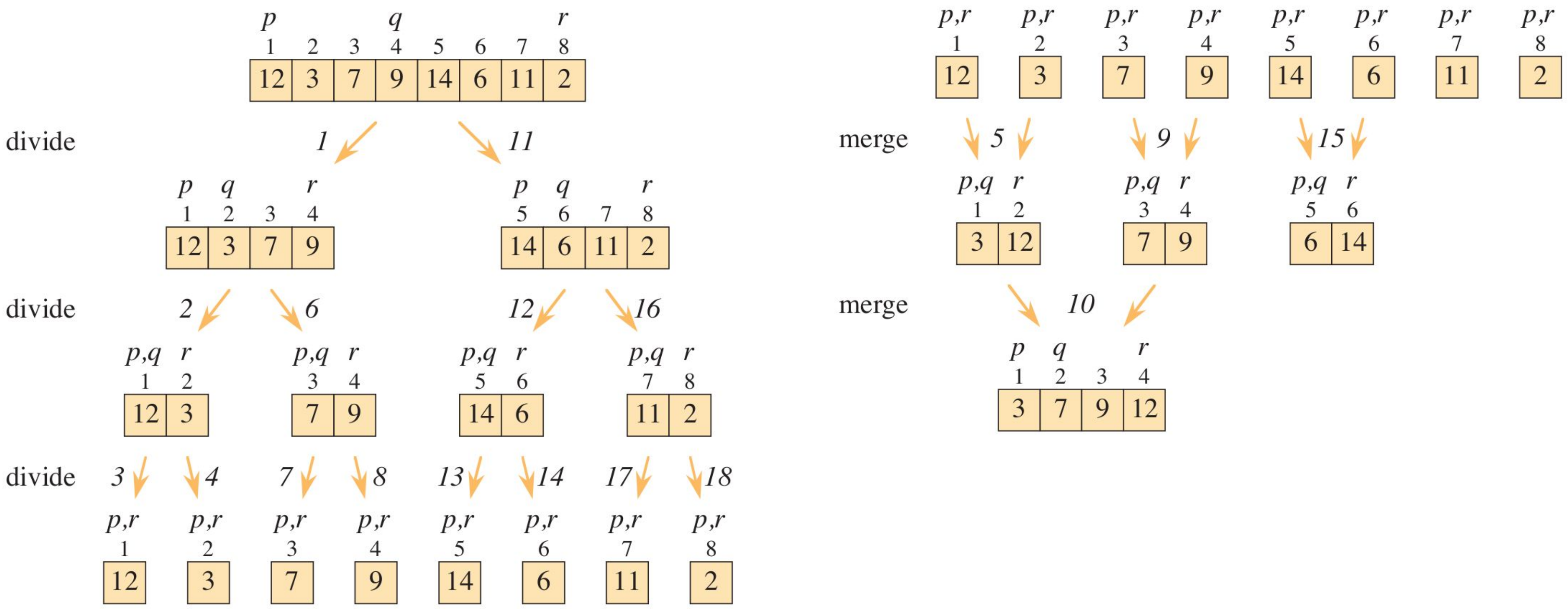
divide
divide
divide
merge
merge
12 3 7 9 14 6 11 2
12 3 7 9
14 6 11 2
3 12
7 9
6 14
3 7 9 12

Merge Sort



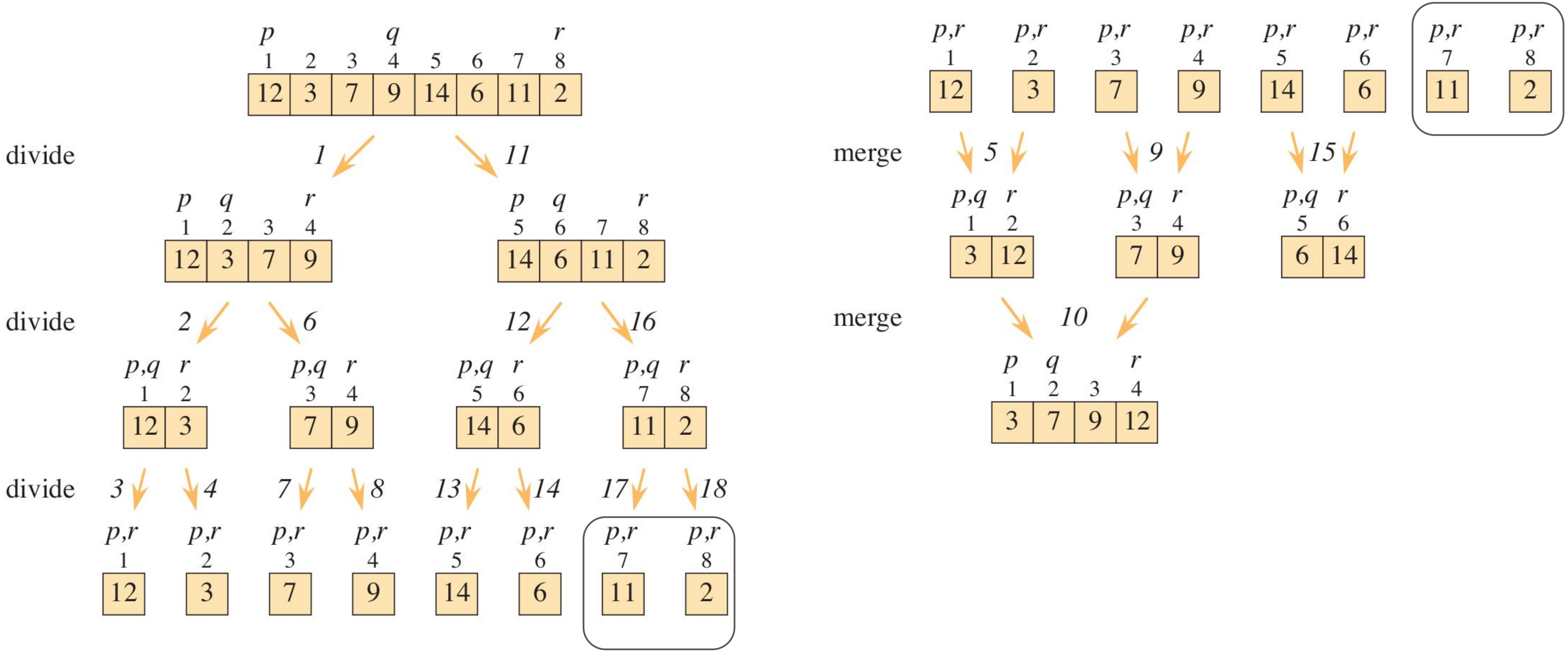
divide
divide
divide
merge
merge
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
1
11
2
6
12
16
3
4
7
8
13
14
5
9
15
10
p,q r
p,r
3 7 9 12

Merge Sort

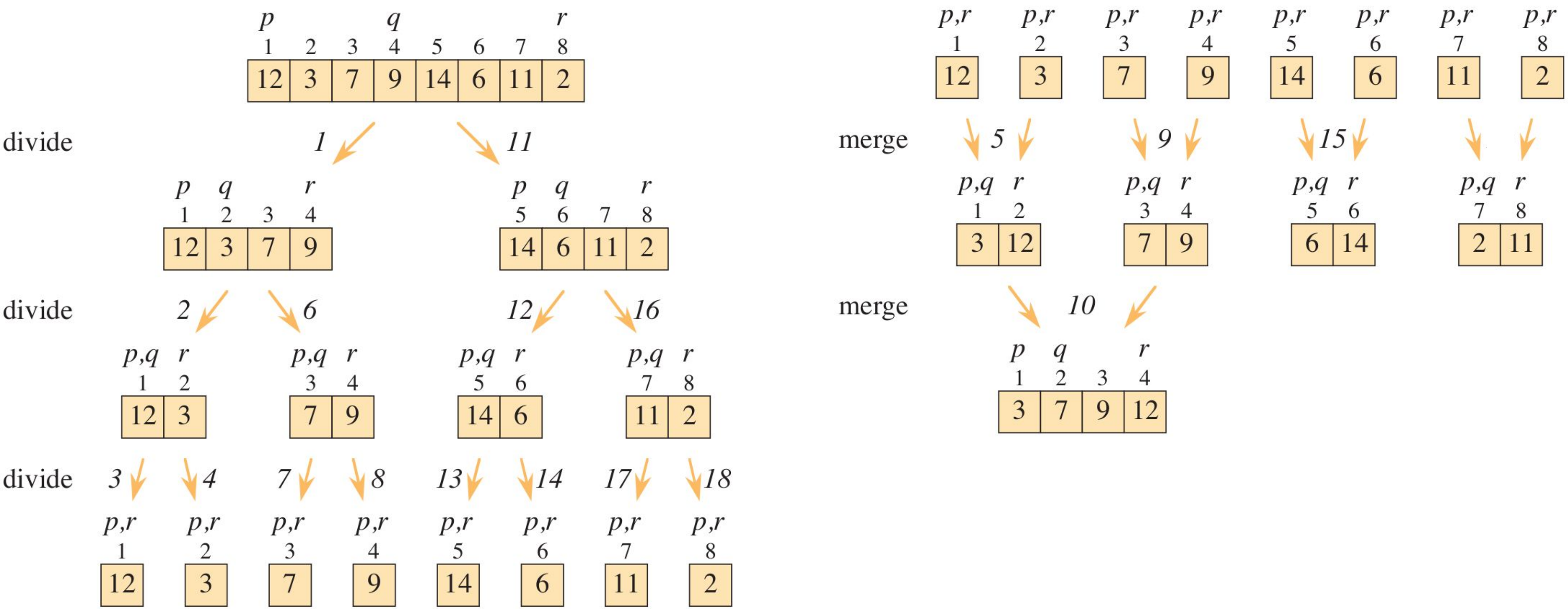


p
q
r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
12
16
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
p,q r
7 8
11 2
divide
3
4
7
8
13
14
17
18
p,r
1
12
p,r
2
3
p,r
3
7
p,r
4
9
p,r
5
14
p,r
6
6
p,r
7
11
p,r
8
2
merge
5
9
15
p,q r
1 2
3 12
p,q r
3 4
7 9
p,q r
5 6
6 14
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort

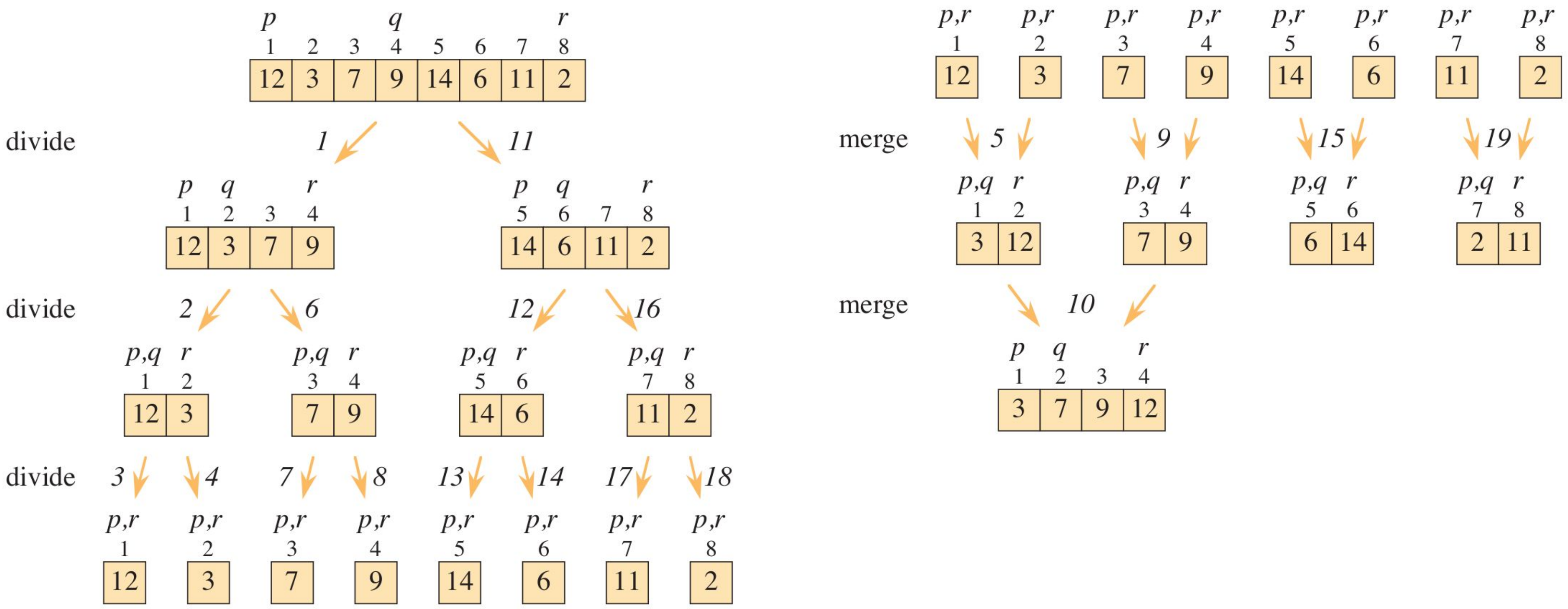


Merge Sort

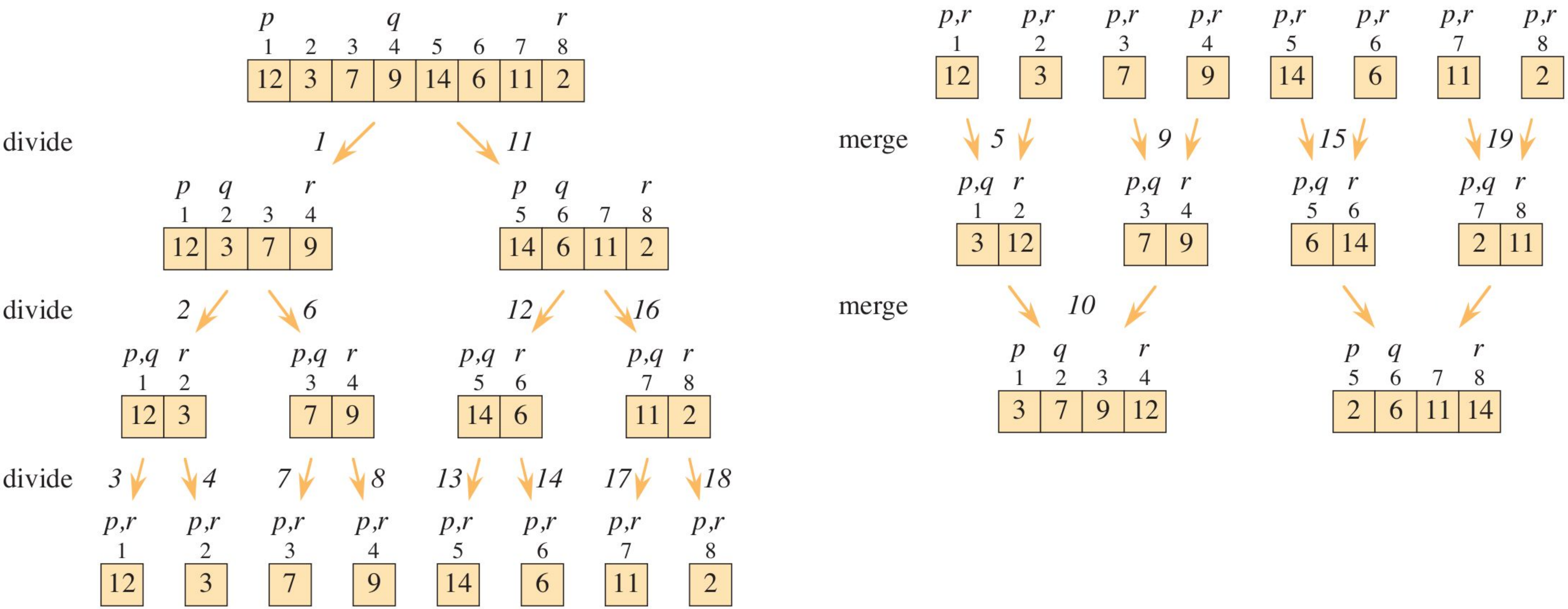


p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2
6
12
16
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
p,q r
7 8
11 2
divide
3 4 7 8 13 14 17 18
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
p,r p,r p,r p,r p,r p,r p,r p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
merge
5
9
15
p,q r
1 2
3 12
p,q r
3 4
7 9
p,q r
5 6
6 14
p,q r
7 8
2 11
merge
10
p q r
1 2 3 4
3 7 9 12

Merge Sort

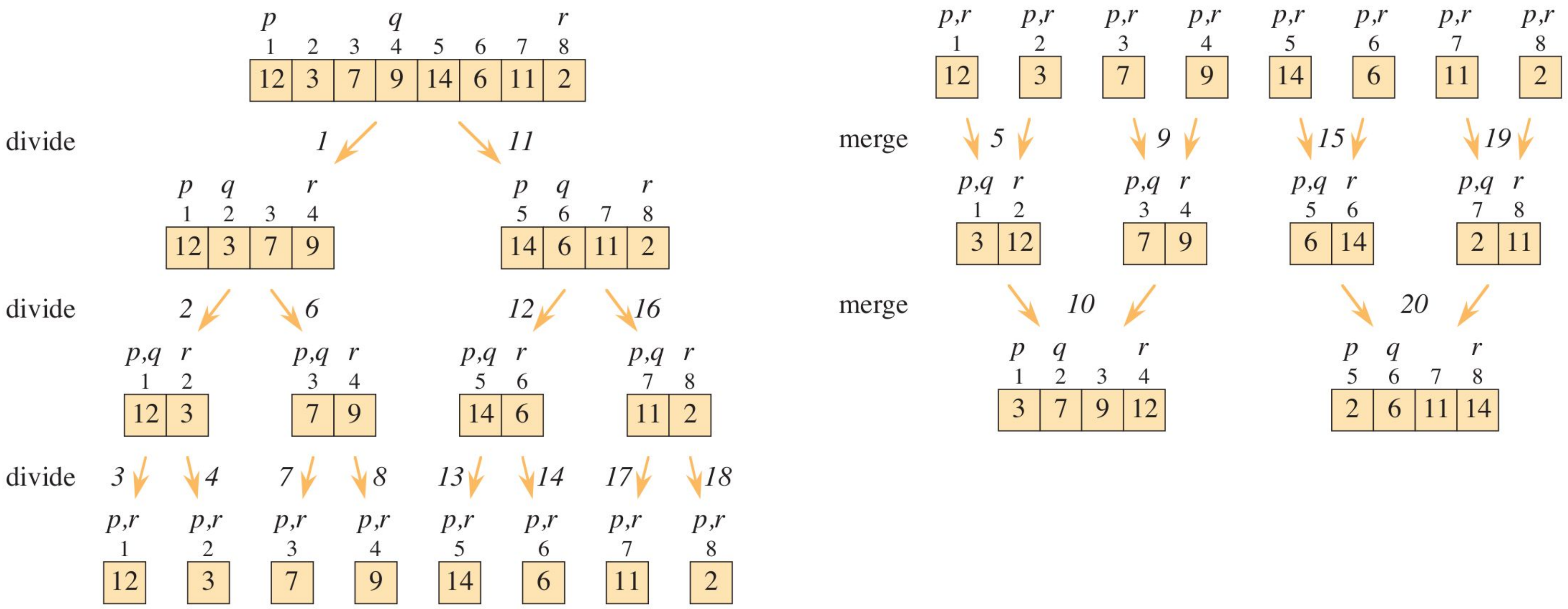


Merge Sort



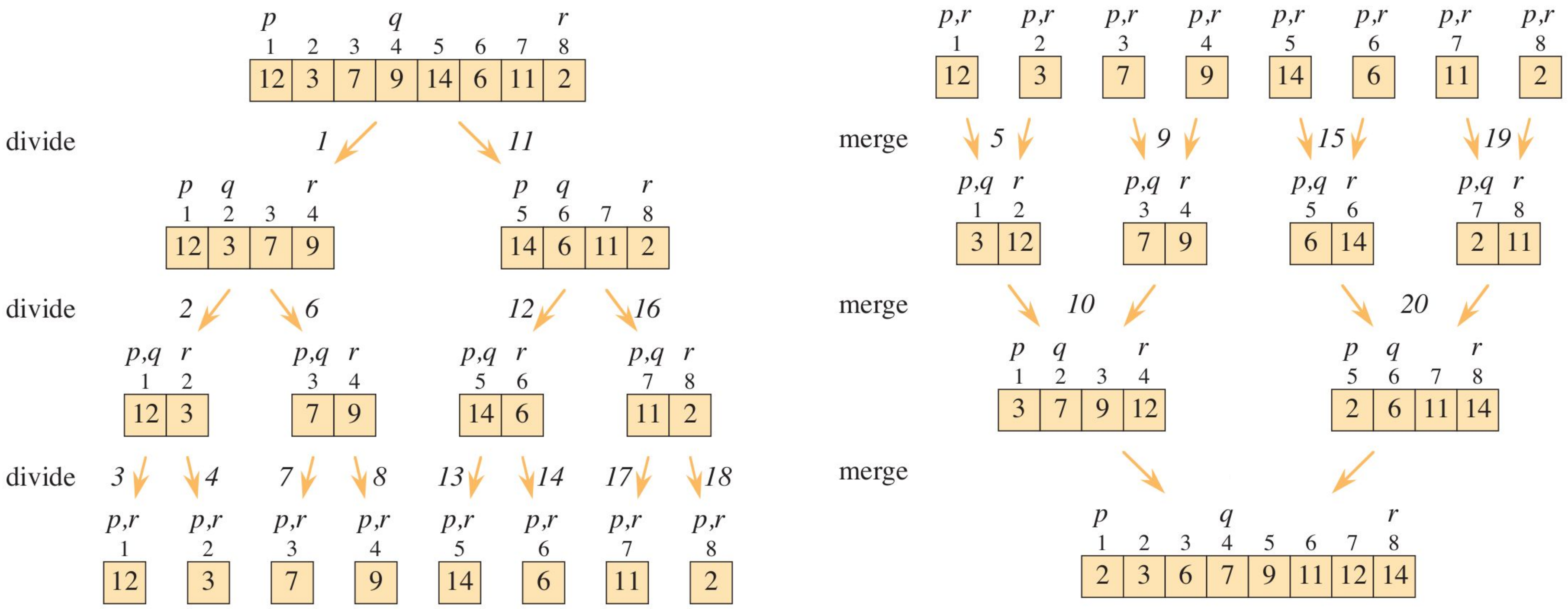
divide
divide
divide
merge
merge
p
q
r
p,q
p,r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
12 3 7 9
14 6 11 2
3 12
7 9
6 14
2 11
3 7 9 12
2 6 11 14

Merge Sort



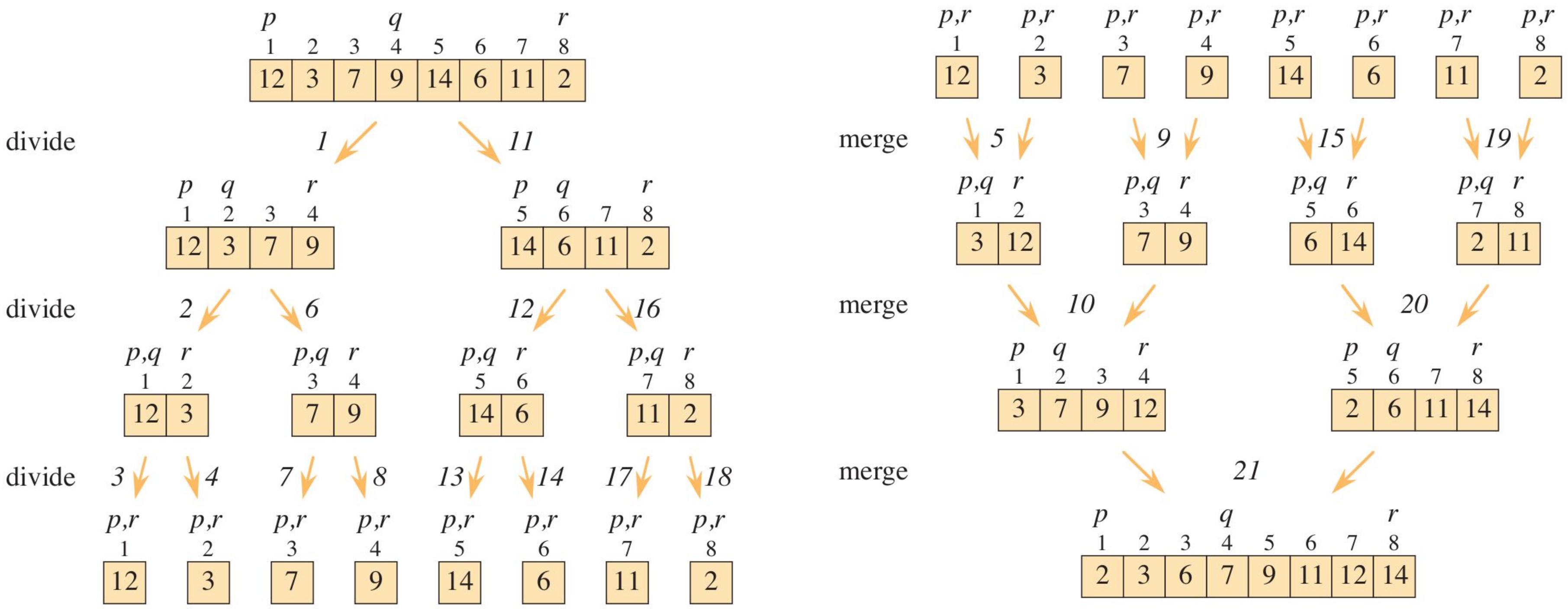
divide
divide
divide
merge
merge
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
1 11
p q r
1 2 3 4
12 3 7 9
5 6 7 8
14 6 11 2
2 6 12 16
p,q r
1 2
12 3
3 4
7 9
5 6
14 6
7 8
11 2
3 4 7 8 13 14 17 18
p,r
1 12
2 3
3 7
4 9
5 14
6 6
7 11
8 2
5 9 15 19
3 12
7 9
6 14
2 11
10 20
3 7 9 12
2 6 11 14

Merge Sort



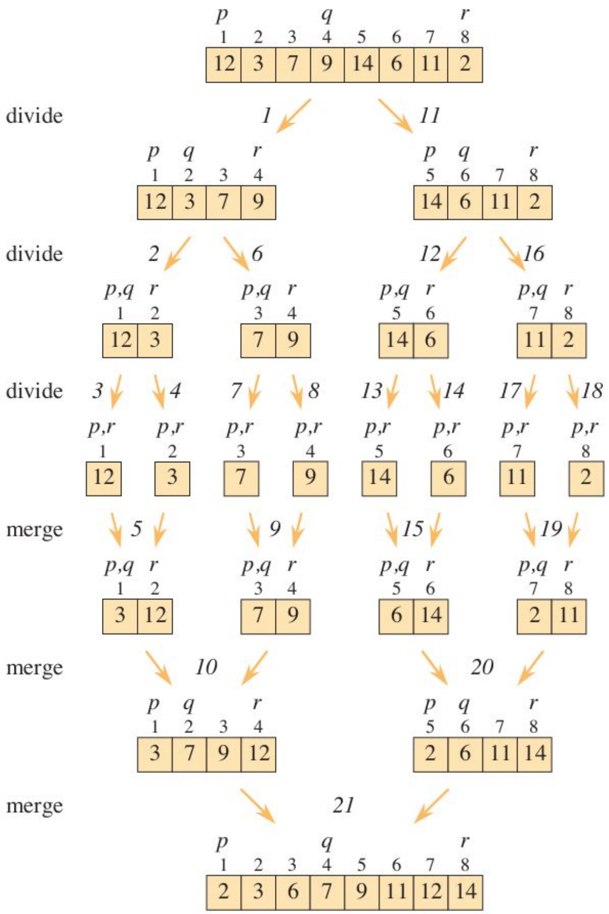
divide
divide
divide
merge
merge
merge
p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
2 3 6 7 9 11 12 14

Merge Sort



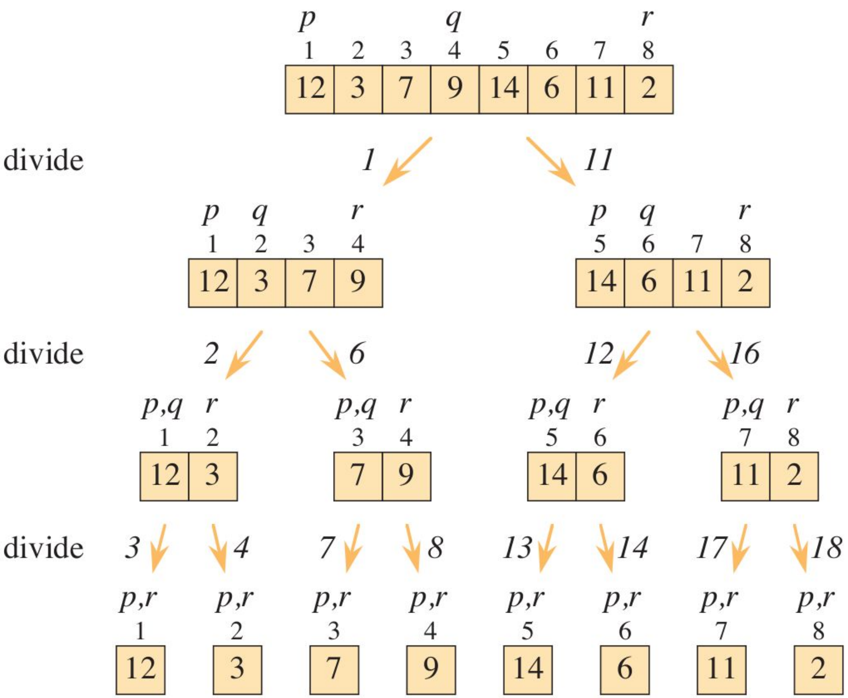
divide
divide
divide
merge
merge
merge
12 3 7 9 14 6 11 2
3 12
7 9
6 14
2 11
3 7 9 12
2 6 11 14
2 3 6 7 9 11 12 14

MERGE SORT



p q r
1 2 3 4 5 6 7 8
12 3 7 9 14 6 11 2
divide
1
11
p q r
1 2 3 4
12 3 7 9
p q r
5 6 7 8
14 6 11 2
divide
2 6 12 16
p,q r
1 2
12 3
p,q r
3 4
7 9
p,q r
5 6
14 6
p,q r
7 8
11 2
divide
3 4 7 8 13 14 17 18
p,r 1 12
p,r 2 3
p,r 3 7
p,r 4 9
p,r 5 14
p,r 6 6
p,r 7 11
p,r 8 2
merge
5 9 15 19
p,q r
1 2
3 12
p,q r
3 4
7 9
p,q r
5 6
6 14
p,q r
7 8
2 11
merge
10 20
p q r
1 2 3 4
3 7 9 12
p q r
5 6 7 8
2 6 11 14
merge
21
p q r
1 2 3 4 5 6 7 8
2 3 6 7 9 11 12 14

Merge Sort

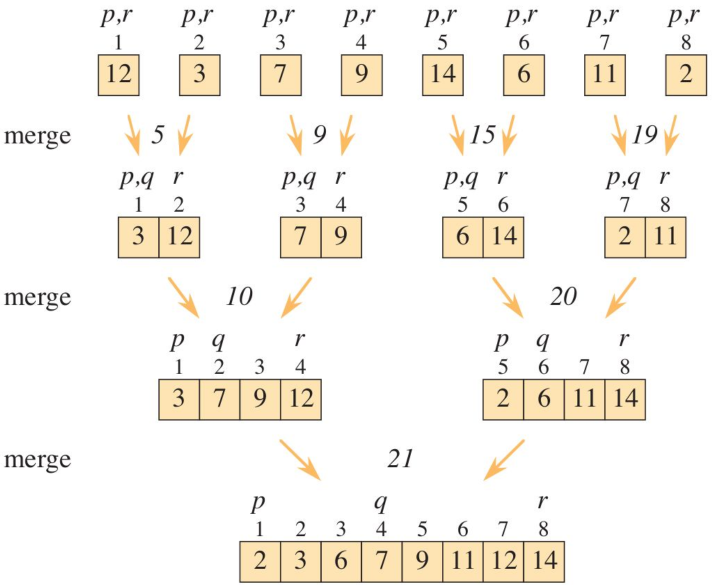



```
MERGE-SORT(A, p, r)
  if p ≥ r
      return
  q = ⌊(p + r)/2⌋
  MERGE-SORT(A, p, q)
  MERGE-SORT(A, q + 1, r)
  // Merge A[p : q] and A[q + 1 : r] into A[p : r].
  MERGE(A, p, q, r)
```

MERGE SORT

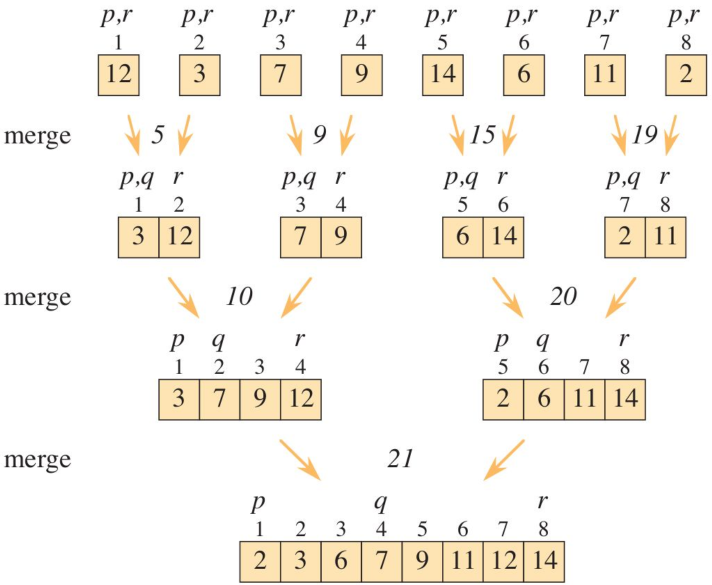
MERGE(A, p, q, r)

$n_L = q - p + 1$ // length of $A[p:q]$
$n_R = r - q$ // length of $A[q+1:r]$
let $L[0:n_L - 1]$ and $R[0:n_R - 1]$ be new arrays
for $i = 0$ **to** $n_L - 1$ // copy $A[p:q]$ into $L[0:n_L - 1]$
  $L[i] = A[p+i]$
for $j = 0$ **to** $n_R - 1$ // copy $A[q+1:r]$ into $R[0:n_R - 1]$
  $R[j] = A[q+j+1]$
$i = 0$ // i indexes the smallest remaining element in L
$j = 0$ // j indexes the smallest remaining element in R
$k = p$ // k indexes the location in A to fill



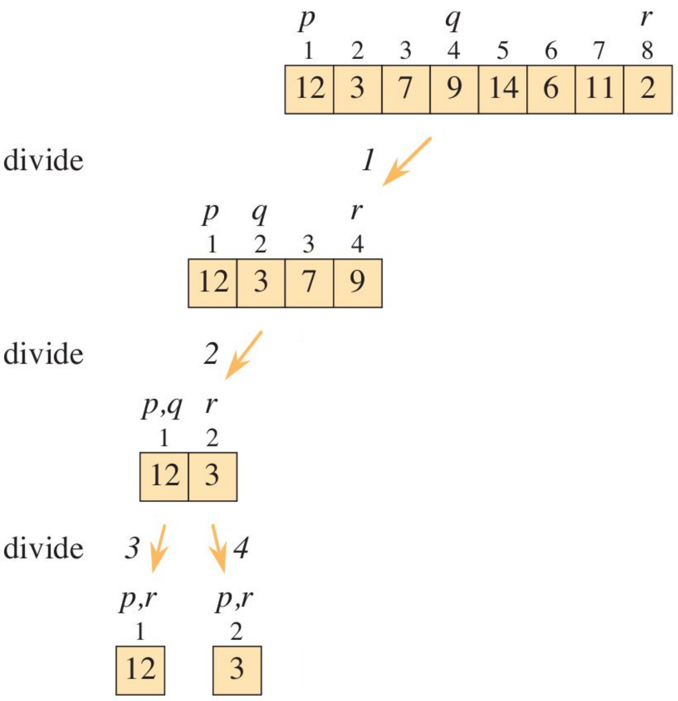
Merge Sort

```
// As long as each of the arrays L and R contains an unmerged element,
//        copy the smallest unmerged element back into A[p : r].
while i < n_L and j < n_R
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
    k = k + 1
// Having gone through one of L and R entirely, copy the
//        remainder of the other to the end of A[p : r].
while i < n_L
    A[k] = L[i]
    i = i + 1
    k = k + 1
while j < n_R
    A[k] = R[j]
    j = j + 1
    k = k + 1
```



Merge Sort - Analysis

Merge Sort



MERGE-SORT(A, p, r)

if $p \geq r$

return

$q = \lfloor (p + r)/2 \rfloor$

MERGE-SORT(A, p, q)

MERGE-SORT$(A, q + 1, r)$

// Merge $A[p:q]$ and $A[q+1:r]$ into $A[p:r]$.

MERGE(A, p, q, r)

$\log_2 8 = 3 \rightarrow 2^3 = 8$

$\log_2 4 = 2 \rightarrow 2^2 = 4$

$\log_b a = c \rightarrow b^c = a$

Merge Sort - Analysis

$$T(n) = \underbrace{T(\frac{n}{2}) + T(\frac{n}{2})}_{\text{Splitting}} + \underbrace{n}_{\text{Merging}}$$

$$T(n) = 2T(\frac{n}{2}) + n$$

Recursively apply the definition of T(n) to T(n/2)

$$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \frac{n}{2}$$

Merge Sort - Analysis

$$T(n) = \overbrace{T(\frac{n}{2}) + T(\frac{n}{2})}^{\texttt{Splitting}} + \overset{\texttt{Merging}}{n}$$

$$T(n) = 2T(\frac{n}{2}) + n$$

Recursively apply the definition of T(n) to T(n/2)

$$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \frac{n}{2}$$

$$T(n) = 2T(\frac{n}{2}) + n$$

$$\boxed{T(n) = 2\left(2T(\frac{n}{4}) + \frac{n}{2}\right) + n}$$

$$T(n) = \left(2 \cdot 2T(\frac{n}{4}) + 2\frac{n}{2}\right) + n$$

$$T(n) = 2^2T(\frac{n}{4}) + n + n$$

$$T(n) = 2^2T(\frac{n}{4}) + 2n$$

$$T(n) = 2^2T(\frac{n}{2^2}) + 2n$$

Merge Sort - Analysis

$$T(n) = \overbrace{T(\frac{n}{2}) + T(\frac{n}{2})}^{\texttt{Splitting}} + \overset{\texttt{Merging}}{n}$$

$$T(n) = 2T(\frac{n}{2}) + n$$

Recursively apply the definition of T(n) to T(n/2)

$$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \frac{n}{2}$$

$$T(n) = 2T(\frac{n}{2}) + n$$

$$\boxed{T(n) = 2\left(2T(\frac{n}{4}) + \frac{n}{2}\right) + n}$$

$$T(n) = \left(2 \cdot 2T(\frac{n}{4}) + 2\frac{n}{2}\right) + n$$

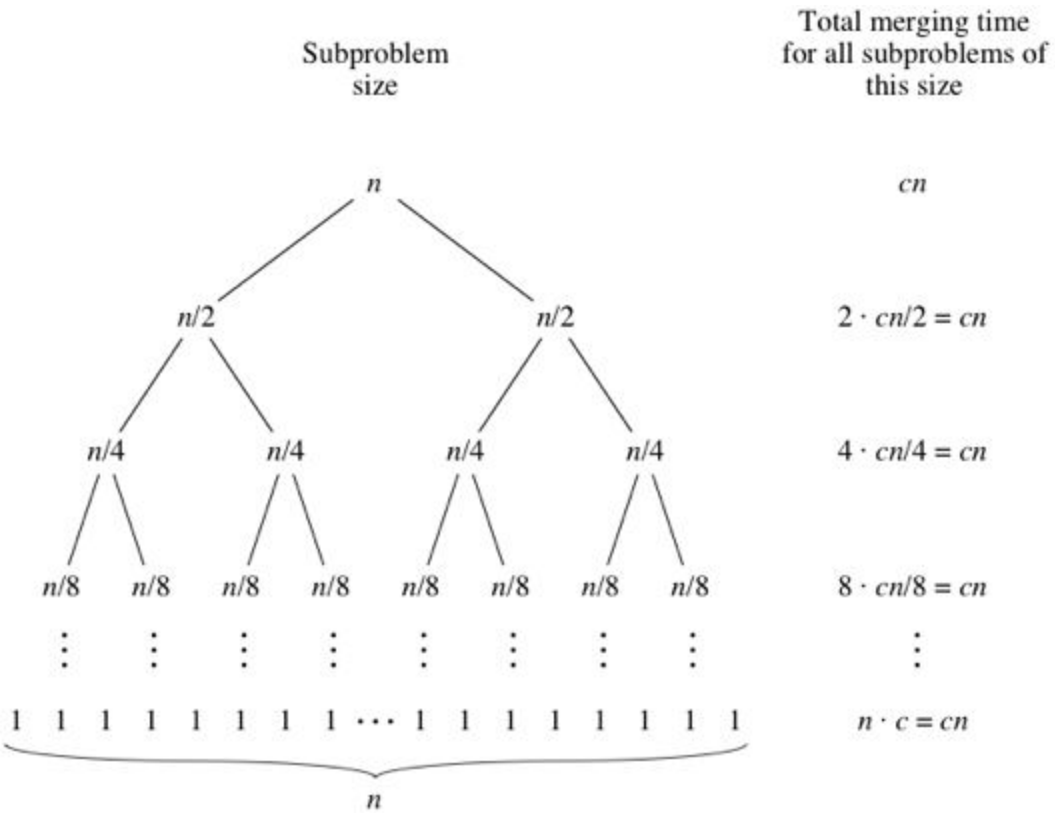
$$T(n) = 2^2T(\frac{n}{4}) + n + n$$

$$T(n) = 2^2T(\frac{n}{4}) + 2n$$

$$T(n) = 2^2T(\frac{n}{2^2}) + 2n$$

$$\boxed{T(n) = 2^iT(\frac{n}{2^i}) + i \cdot n}$$

Merge Sort - Analysis



Merge Sort - Analysis

$$\boxed{T(n) = 2^i T(\frac{n}{2^i}) + i \cdot n}$$

$$T(1) = 1$$

$$T(\frac{n}{2^i}) = 1$$

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

$$\boxed{log_2 n = i}$$

Merge Sort - Analysis

$$\boxed{T(n) = 2^i T(\frac{n}{2^i}) + i \cdot n}$$

$$T(1) = 1$$

$$T(\frac{n}{2^i}) = 1$$

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

$$\boxed{log_2 n = i}$$

$$T(n) = 2^i T(\frac{n}{2^i}) + i \cdot n$$

$$T(n) = nT(1) + log_2 n \cdot n$$

$$T(n) = n + n \cdot log_2 n$$

Merge Sort - Analysis

$$\boxed{T(n) = 2^i T(\frac{n}{2^i}) + i \cdot n}$$

$$T(1) = 1$$

$$T(\frac{n}{2^i}) = 1$$

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

$$\boxed{log_2 n = i}$$

$$T(n) = 2^i T(\frac{n}{2^i}) + i \cdot n$$

$$T(n) = nT(1) + log_2 n \cdot n$$

$$T(n) = n + n \cdot log_2 n$$

$$\boxed{T(n) = n + n \log n}$$

Insertion Sort

Analyzing Algorithms

Best case

The array is already sorted.

- Always find that $A[i] \leq key$ upon the first time the **while** loop test is run (when $i = i - 1$).
- All t_i are 1.
- Running time is

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

- Can express $\boxed{T(n) \text{ as } an + b}$ for constants a and b (that depend on the statement costs c_k) $\Rightarrow T(n)$ is a *linear function* of n.

Analyzing Algorithms

Worst case

- Since $\sum_{i=2}^{n} i = \left(\sum_{i=1}^{n} i\right) - 1$, it equals $\frac{n(n+1)}{2} - 1$.
- Letting $l = i - 1$, we see that $\sum_{i=2}^{n} (i-1) = \sum_{l=1}^{n-1} l = \frac{n(n-1)}{2}$.
- Running time is

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

- Can express $\boxed{T(n) \text{ as } an^2 + bn + c}$ for constants a, b, c (that again depend on statement costs) $\Rightarrow T(n)$ is a *quadratic function* of n.

Comparison

Merge Sort

Merge Sort

$T(n) = n + n \, log \, n$

Insertion Sort

Best: $an + b$

Worst: $an^2 + bn + c$