GDI2 – 5 - Quicksort

Idea

Divide: Partition (rearrange) the array $A[p \,..\, r]$ into two (possibly empty) subarrays $A[p \,..\, q-1]$ and $A[q+1 \,..\, r]$ such that each element of $A[p \,..\, q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1 \,..\, r]$. Compute the index q as part of this partitioning procedure.

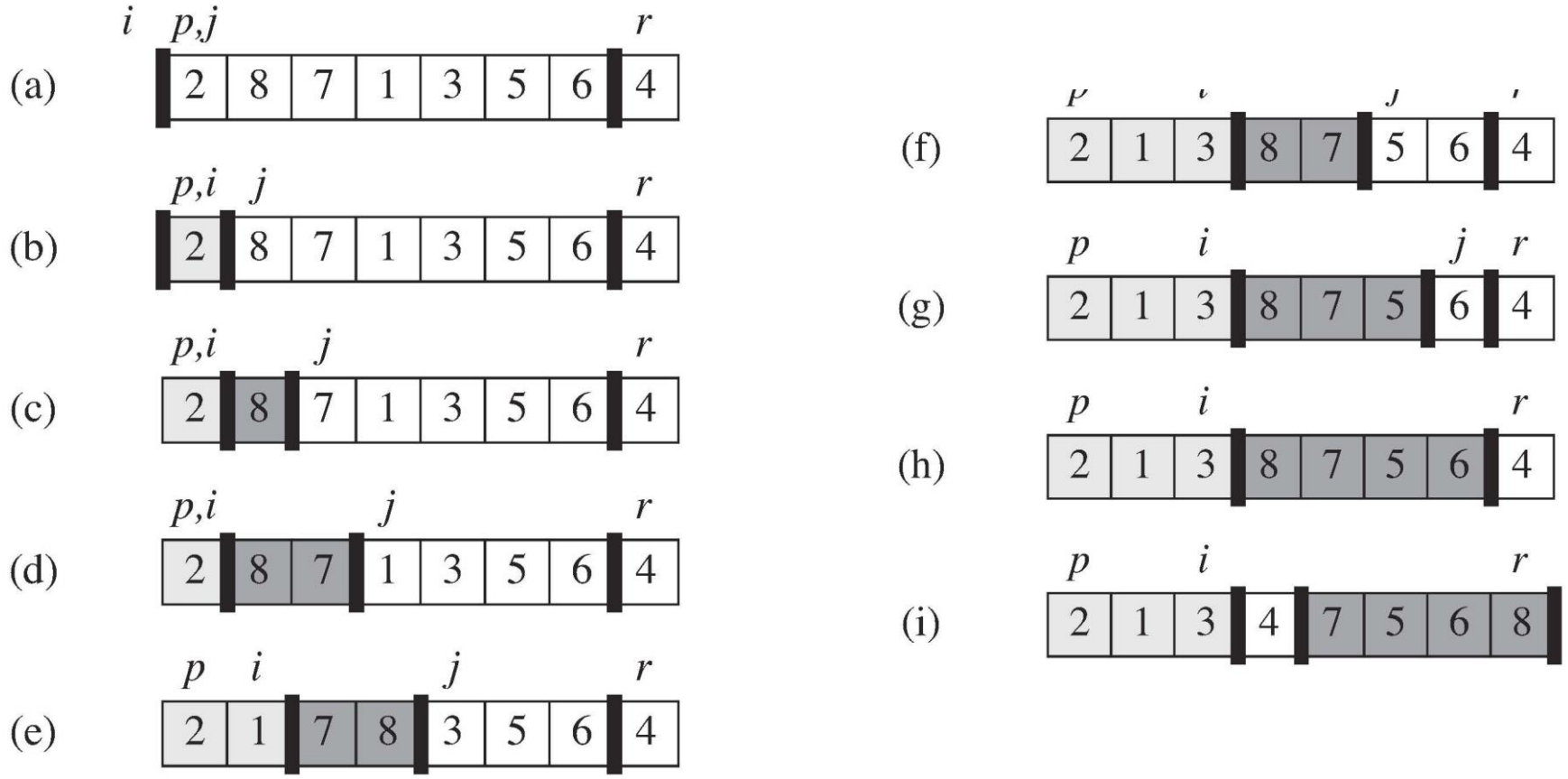
Conquer: Sort the two subarrays $A[p \,..\, q-1]$ and $A[q+1 \,..\, r]$ by recursive calls to quicksort.

Combine: Because the subarrays are already sorted, no work is needed to combine them: the entire array $A[p \,..\, r]$ is now sorted.

Quicksort

```
QUICKSORT(A, p, r)
1   if p < r
2       q = PARTITION(A, p, r)
3       QUICKSORT(A, p, q − 1)
4       QUICKSORT(A, q + 1, r)
```

Partition I


(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)
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

Partition II

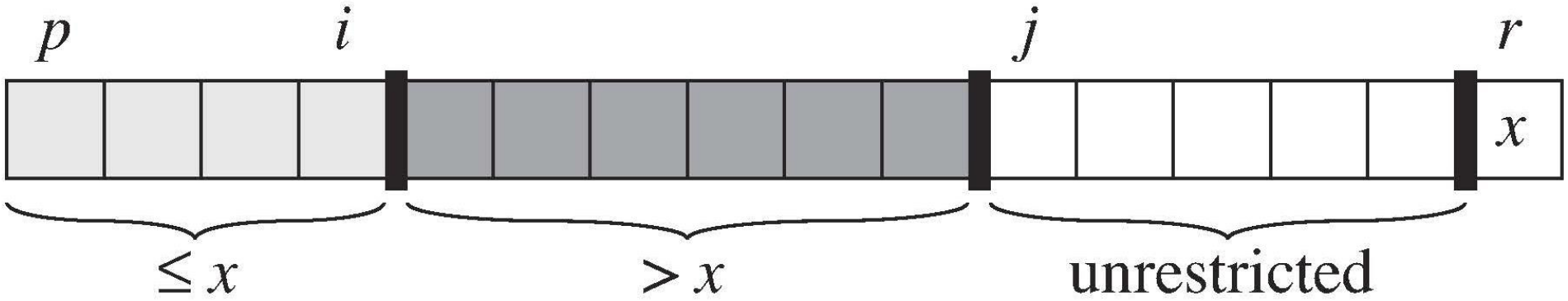
```
PARTITION(A, p, r)
1   x = A[r]
2   i = p − 1
3   for j = p to r − 1
4       if A[j] ≤ x
5           i = i + 1
6           exchange A[i] with A[j]
7   exchange A[i + 1] with A[r]
8   return i + 1
```

Invariante

At the beginning of each iteration of the loop of lines 3–6, for any array index k,

1. If $p \leq k \leq i$, then $A[k] \leq x$.
2. If $i + 1 \leq k \leq j - 1$, then $A[k] > x$.
3. If $k = r$, then $A[k] = x$.

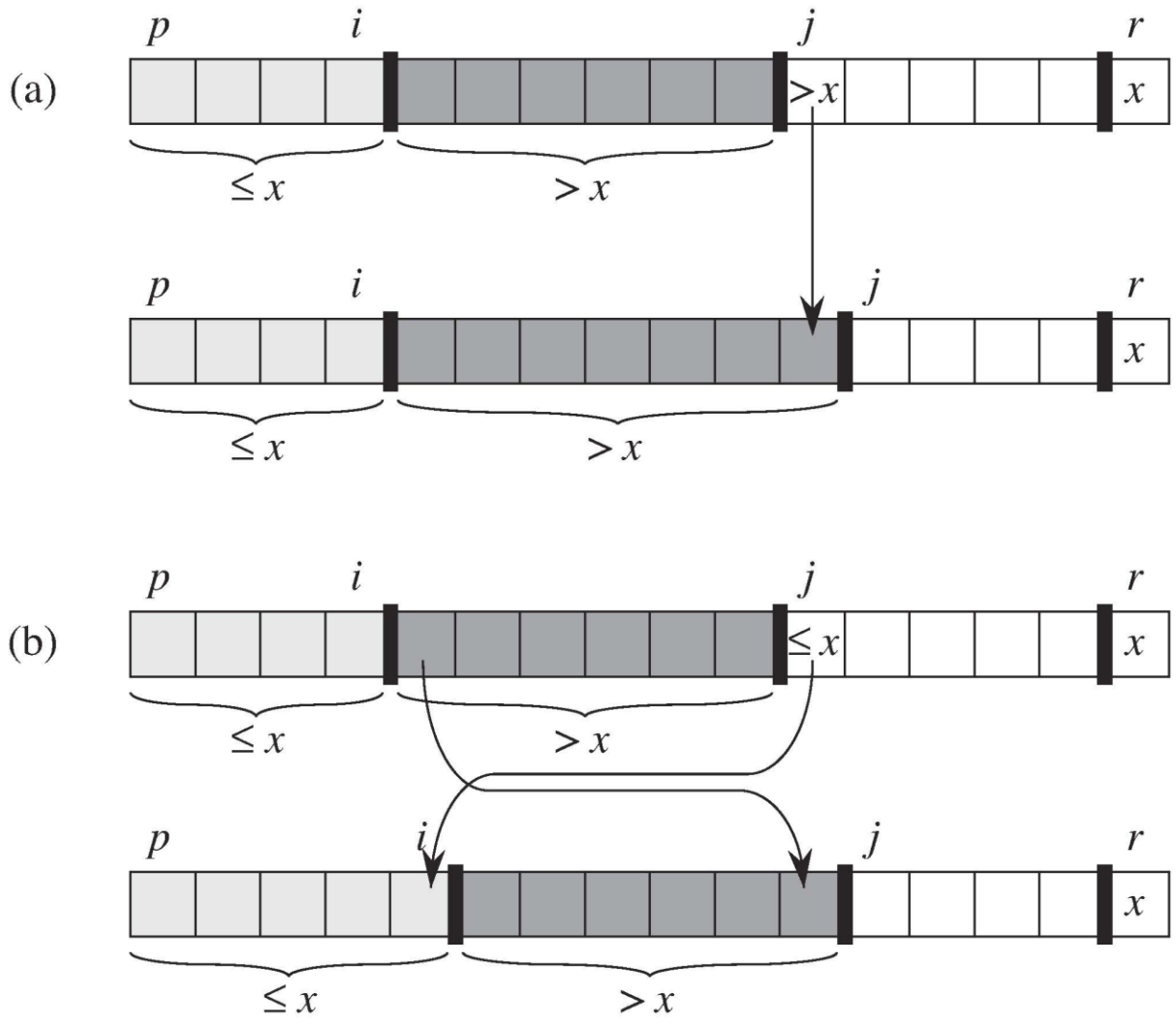
Erhaltung



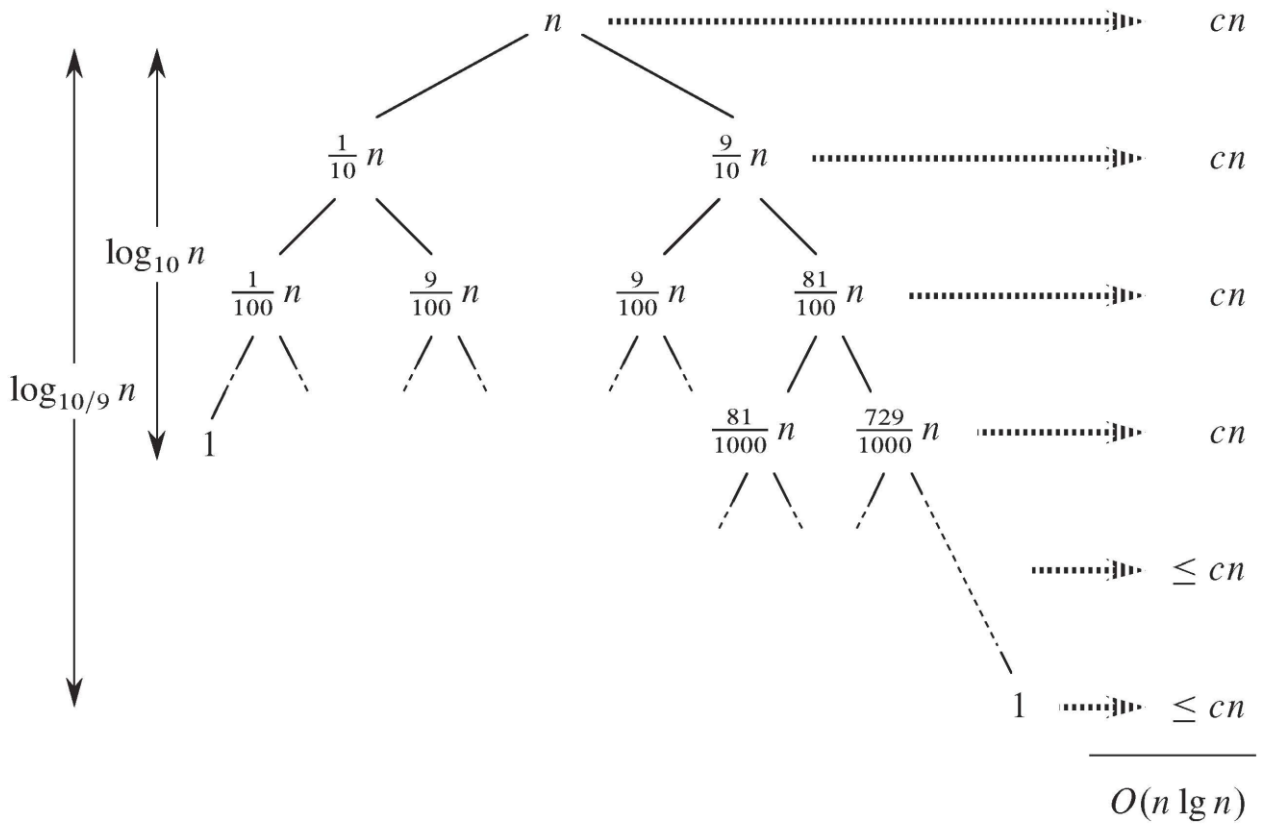
Initialisierung

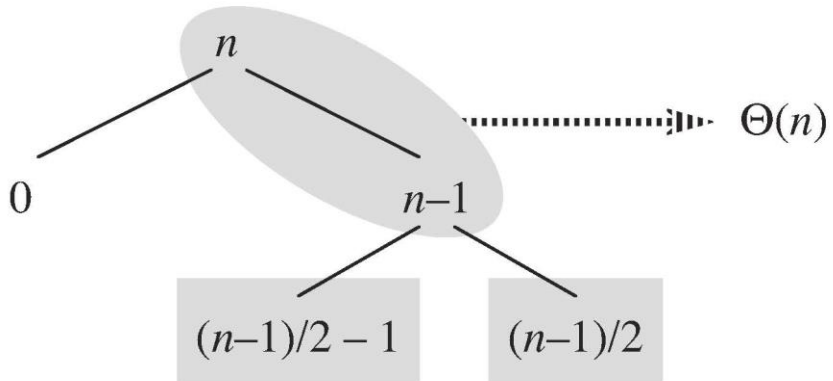
Initialization: Prior to the first iteration of the loop, $i = p - 1$ and $j = p$. Because no values lie between p and i and no values lie between $i + 1$ and $j - 1$, the first two conditions of the loop invariant are trivially satisfied. The assignment in line 1 satisfies the third condition.

Erhaltung

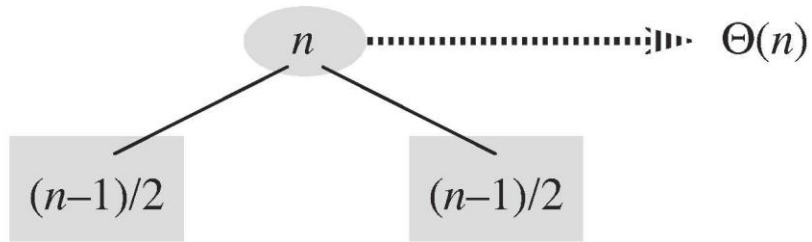


Aufteilung 1/10 – 9/10





(a)

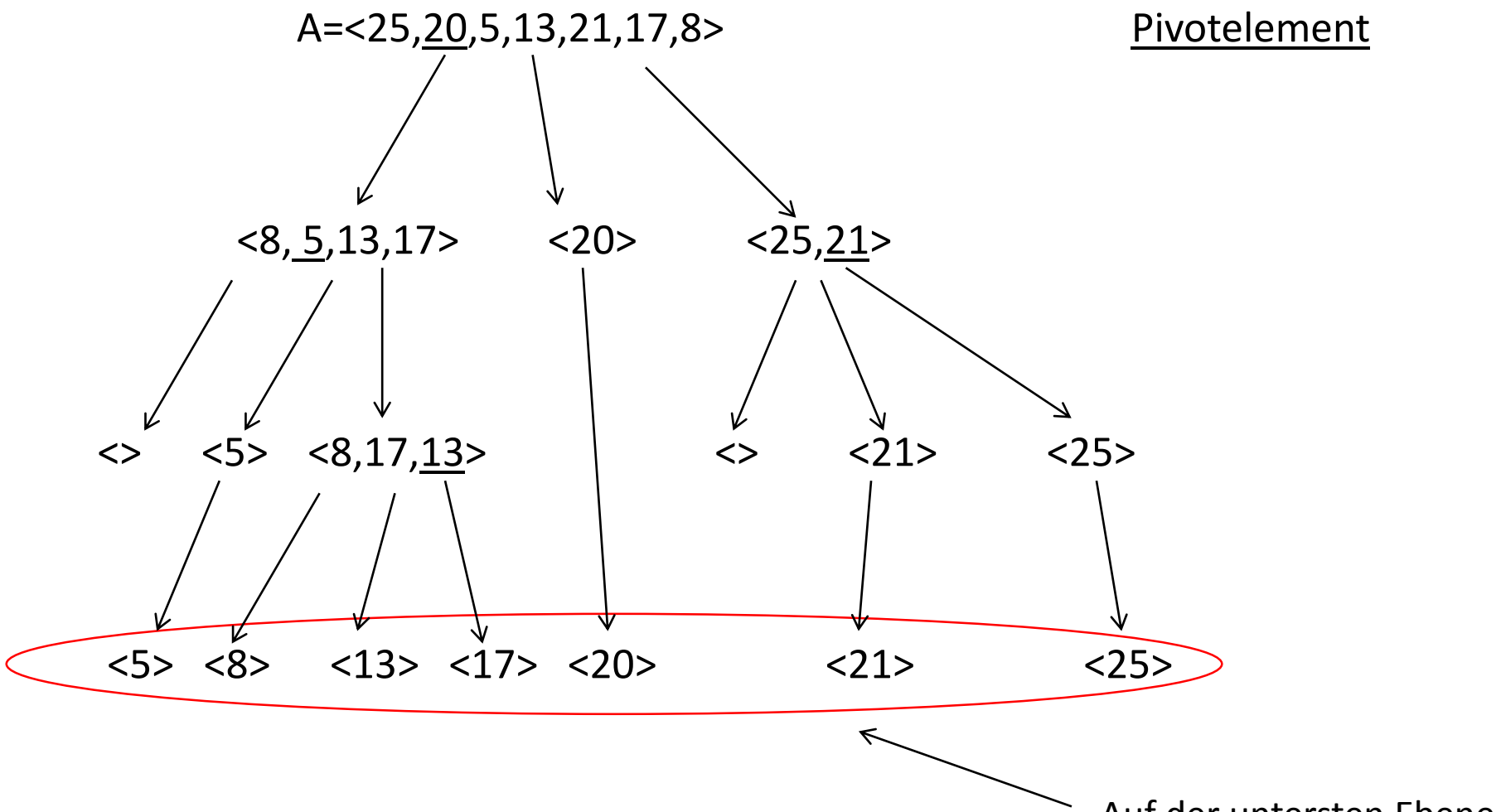


(b)

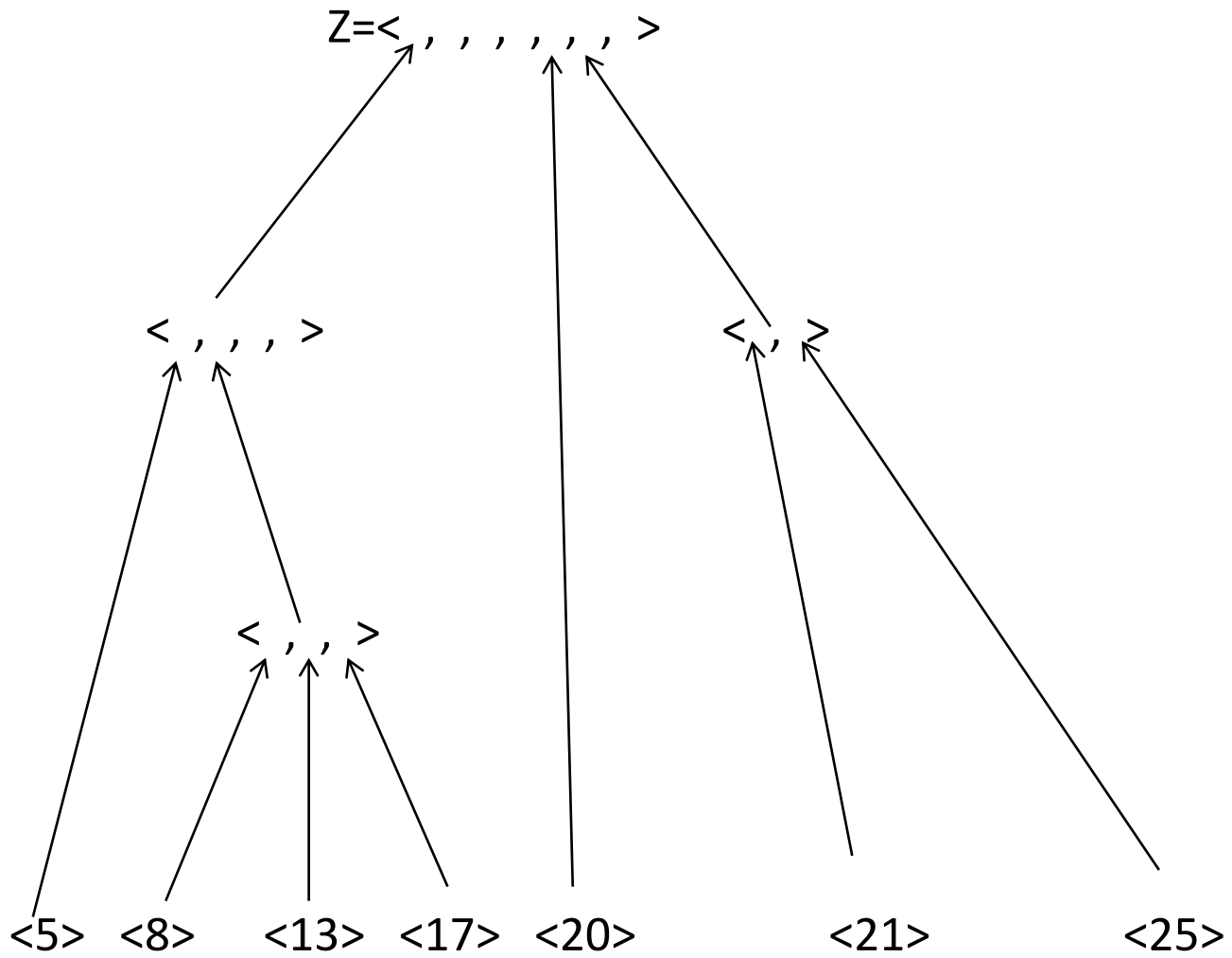
```
RANDOMIZED-PARTITION(A, p, r)
1  i = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2      q = RANDOMIZED-PARTITION(A, p, r)
3      RANDOMIZED-QUICKSORT(A, p, q − 1)
4      RANDOMIZED-QUICKSORT(A, q + 1, r)
```

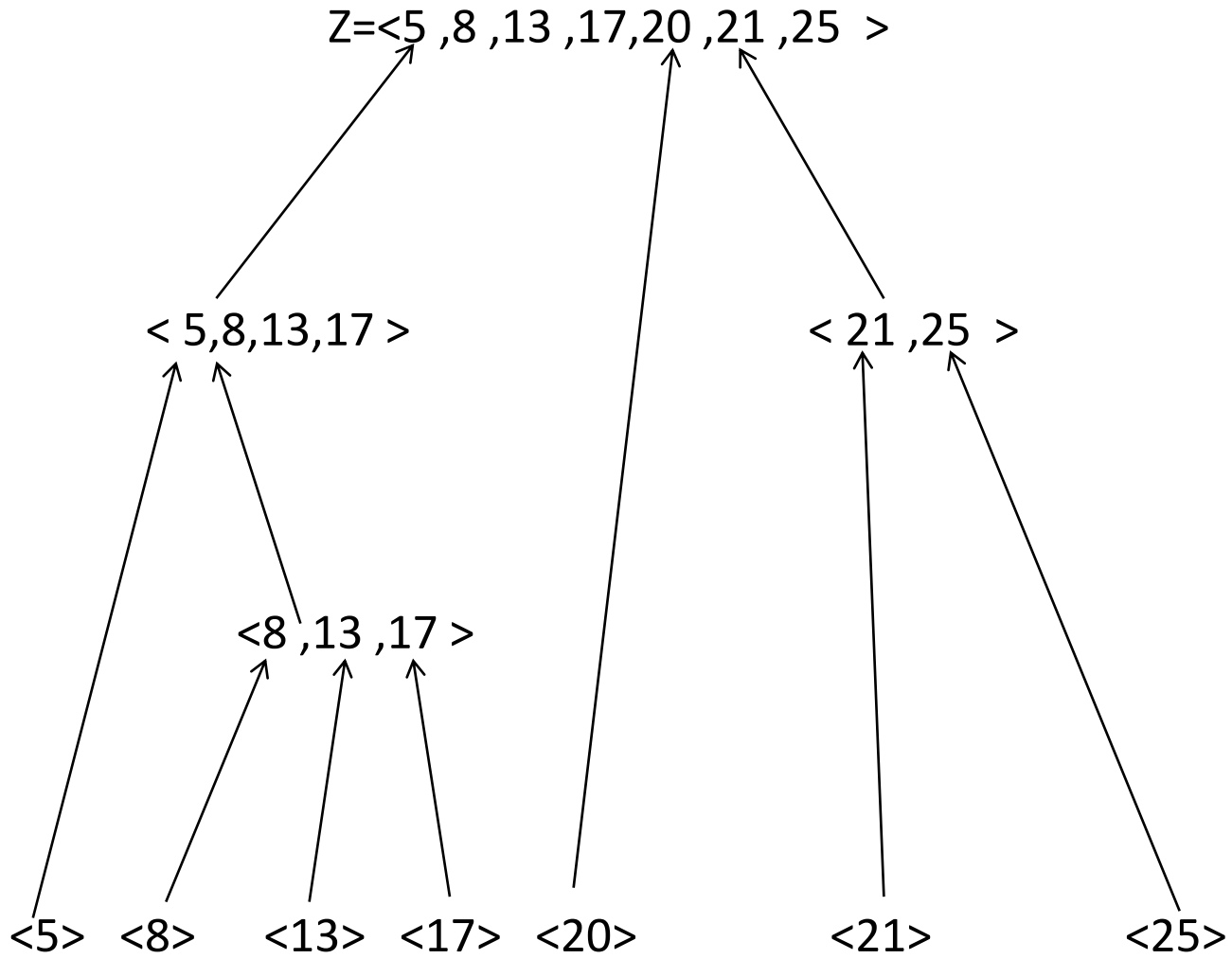
Schritt 1: Zerlegung von A



Schritt 2: Subarrays zusammensetzen I

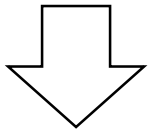


Schritt 2: Subarrays zusammensetzen II



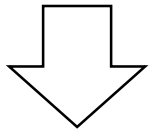
Werden 8 und 13 verglichen? Z_{ij}=<8,13>

Z=<5,8,13,17,20,21,25>

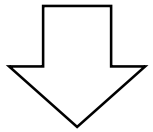


<5,8,13,17> <20> <21,25>

Z_{ij} wird zerlegt



<> <5> <8,13,17> <20> <> <21> <25>



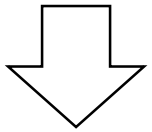
<5> <8> <13> <17> <20> <21><25>

Bei der Zerlegung von Z_{ij} ist z_j ist das Pivotelement

→Es findet ein Vergleich zwischen z_i und z_j statt

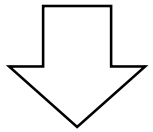
Werden 8 und 17 verglichen? Z_{ij}=<8,13,17>

Z=<5,8,13,17,20,21,25>

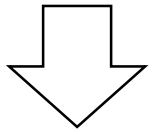


<5,8,13,17> <20> <21,25>

Z_{ij} wird zerlegt



<> <5> <8,13,17> <20> <> <21> <25>



<5> <8> <13> <17> <20> <21><25>

Bei der Zerlegung von Z_{ij} ist weder z_i noch z_j das Pivotelement

→Es findet kein Vergleich zwischen z_i und z_j statt