Divide and Conquer

Divide and Conquer

- Termination: If the problem is small enough, then solve it directly.
- Divide: Break-down the problem into one or more subproblems.
- Conquer: Solve the subproblems
- Combine: Compute the solution by combining the solutions of subproblems

Long Multiplication

```
   234     234     234     234
  ×123   ×   3   ×   2   ×   1
 -----   -----   -----   -----
   702      12       8       4
  468       9       6       3
 234       6       4       2
 -----   -----   -----   -----
 28782     702     468     234
```

Long Multiplication

- Termination: $x,y \in \{0,\ldots,9\}$
- Divide: If $y=y_n \ldots y_0 \geq 10$, then divide the problem into $x \times y_n, \ldots, x \times y_0$. If $x=x_m \ldots x_0 \geq 10$ and $y<10$, then divide the problem into $x_m \times y, \ldots, x_0 \times y$.
- Conquer: Solve the subproblems
- Combine: Compute $\Sigma_{0 \leq i \leq n} x \times y_i \times 10^i$ for the first case or $\Sigma_{0 \leq j \leq m} x_j \times y \times 10^j$ for the second.

Assume multiplying 10^p and adding two numbers can be done in $O(1)$.

Long Multiplication

- Time complexity: $T(x,y)$
- $T(x,y)=O(1)$ if $x,y\in\{0,\dots,9\}$
- $T(x,y)=\Sigma_{0\leq i\leq n}T(x,y_i)+O(n)$ if $y=y_n\dots y_0\geq 10$
- $T(x,y)=\Sigma_{0\leq j\leq m}T(x_j,y)+O(m)$ if $x=x_m\dots x_0\geq 10$ and $y<10$
- $T(x,y)=\Sigma_{0\leq i\leq n}T(x,y_i)+O(n)$
  $=\Sigma_{0\leq i\leq n}(\Sigma_{0\leq j\leq m}T(x_j,y_i)+O(m))+O(n)$
  $=(n+1)(m+1)O(1)+(n+1)O(m)+O(n)$
  $=O(mn)=O(\log x\times\log y)$

Faster Multiplication

- Andrey Kolmogorov conjectured multiplication takes $\Omega(nm)$ in 1952.
- In 1960, a 23-year-old student, Anatolii Alexeevitch Karatsuba, found a simple $O(n^{1.59})$-time algorithm.
- Toom–Cook: $O(n^{\log(2k-1)/\log k})$
- Schönhage–Strassen: FFT-based algorithm in $O(n\log n\log\log n)$

Karatsuba Algorithm

- Let $x=x_HB+x_L$ and $y=y_HB+y_L$ where $x_L<B$, $y_L<B$, and $y\leq x<B^2$.
- $xy=x_Hy_HB^2+x_Ly_HB+x_Hy_LB+x_Ly_L$.
- 4 subproblems x_Hy_H, x_Ly_H, x_Hy_L, x_Ly_L.
- $T(n)=4T(n/2)+O(n)=O(n^2)$ … we'll justify this later
- Karatsuba's Goal: reduce the number of subproblems to 3!

Karatsuba Algorithm

- $z = z_H B^2 + z_M B + z_L$
  $= xy = x_H y_H B^2 + x_L y_H B + x_H y_L B + x_L y_L$
- $z_H = x_H y_H$ and $z_L = x_L y_L$
- $z_M = x_L y_H + x_H y_L$
  $= (x_H + x_L) \times (y_H + y_L) - x_H y_H - x_L y_L$
  $= (x_H + x_L) \times (y_H + y_L) - z_H - z_L$

Karatsuba Algorithm

- Termination: If x and y are small, multiply them by long multiplication.
- Divide: $x_H \times y_H$, $x_L \times y_L$, $(x_H + x_L) \times (y_H + y_L)$
- Conquer: Solve the subproblems
- Combine: Let $z_H = x_H y_H$, $z_L = x_L y_L$, $z_M = (x_H + x_L) \times (y_H + y_L) - z_H - z_L$, and $x \times y = z_H B^2 + z_M B + z_L$.
- Time: $T(n) = 3T(n/2) + O(n) = O(n^{\log 3/\log 2})$

Strassen algorithm

- Multiply two n-by-n matrices A and B takes O(n³) multiplications by definition.

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \qquad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$$

$$C = AB = \begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}$$

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Strassen algorithm

- Naive recursive version: 8 subproblems
- $T(n)=8T(n/2)+O(n^2)=O(n^3)$
- Strassen defines
  - $M_1=(A_{1,1}+A_{2,2})(B_{1,1}+B_{2,2})$
    $M_2=(A_{2,1}+A_{2,2})B_{1,1}$
    $M_3=A_{1,1}(B_{1,2}-B_{2,2})$
    $M_4=A_{2,2}(B_{2,1}-B_{1,1})$
    $M_5=(A_{1,1}+A_{1,2})B_{2,2}$
    $M_6=(A_{2,1}-A_{1,1})(B_{1,1}+B_{1,2})$
    $M_7=(A_{1,2}-A_{2,2})(B_{2,1}+B_{2,2})$

Strassen algorithm

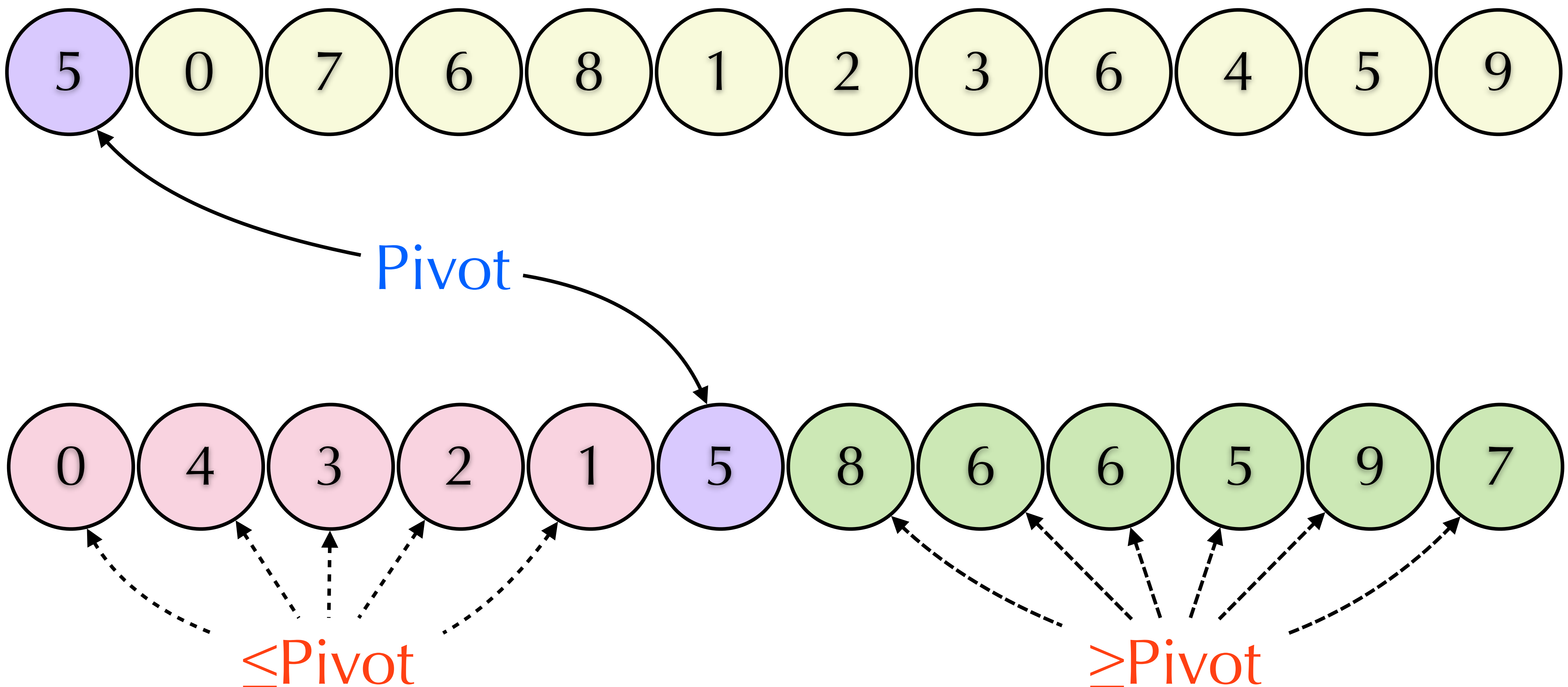
- $C_{1,1}=M_1+M_4-M_5+M_7$
- $C_{1,2}=M_3+M_5$
- $C_{1,3}=M_2+M_4$
- $C_{1,4}=M_1-M_2+M_3+M_6$
- 7 subproblems
- $T(n)=7T(n/2)+O(n^2)$
  $=O(n^{\log 7/\log 2})=O(n^{2.81})$ … err, why?

Quick Sort

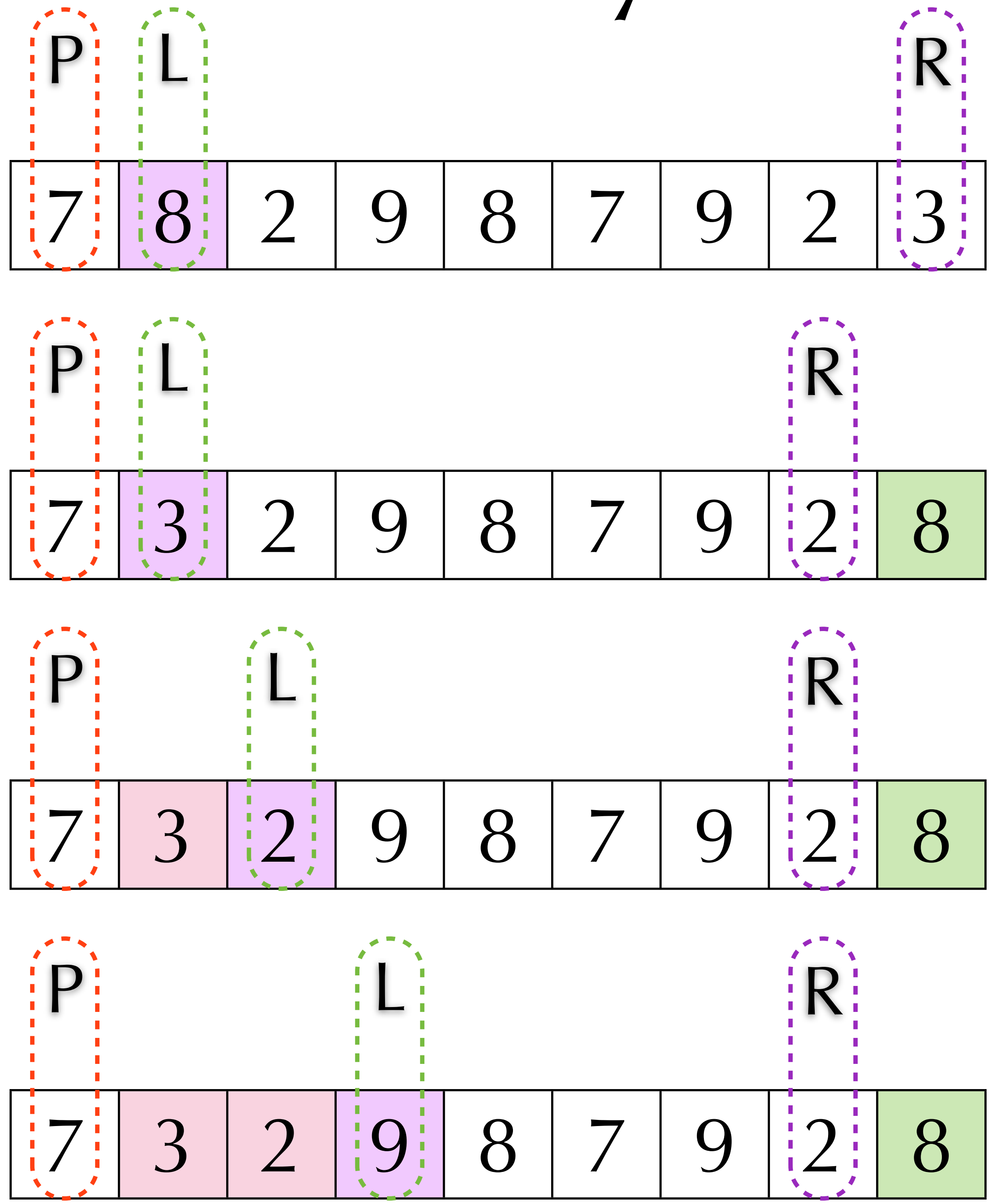
$\boxed{\text{Sort } A[1],\ldots,A[n]}$

- Termination: It is sorted when n=1.
- Divide: Reorder A and find m such that
  - For $i<m$, $A[i]\leq A[m]$.
  - For $i>m$, $A[i]\geq A[m]$.
- Conquer: Sort $A[1..m-1]$ and $A[m+1..n]$.
- Combine: No need.
- Time: $T(n)=T(m-1)+T(n-m)+O(n)$

Partition by Pivot



Partition by Pivot

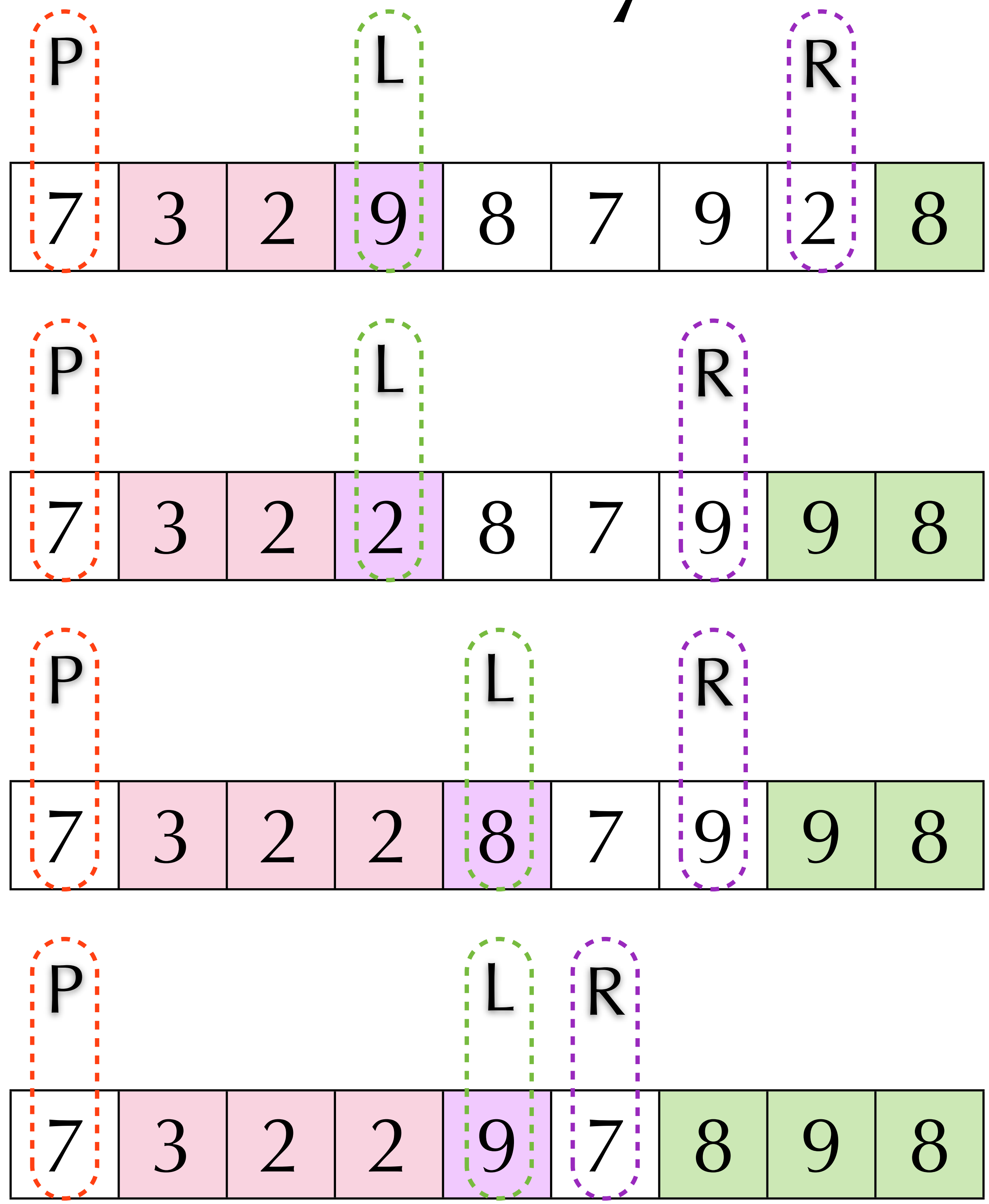


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If A[P]≥A[L]:
L=L+1

If A[P]≥A[L]:
L=L+1

Partition by Pivot

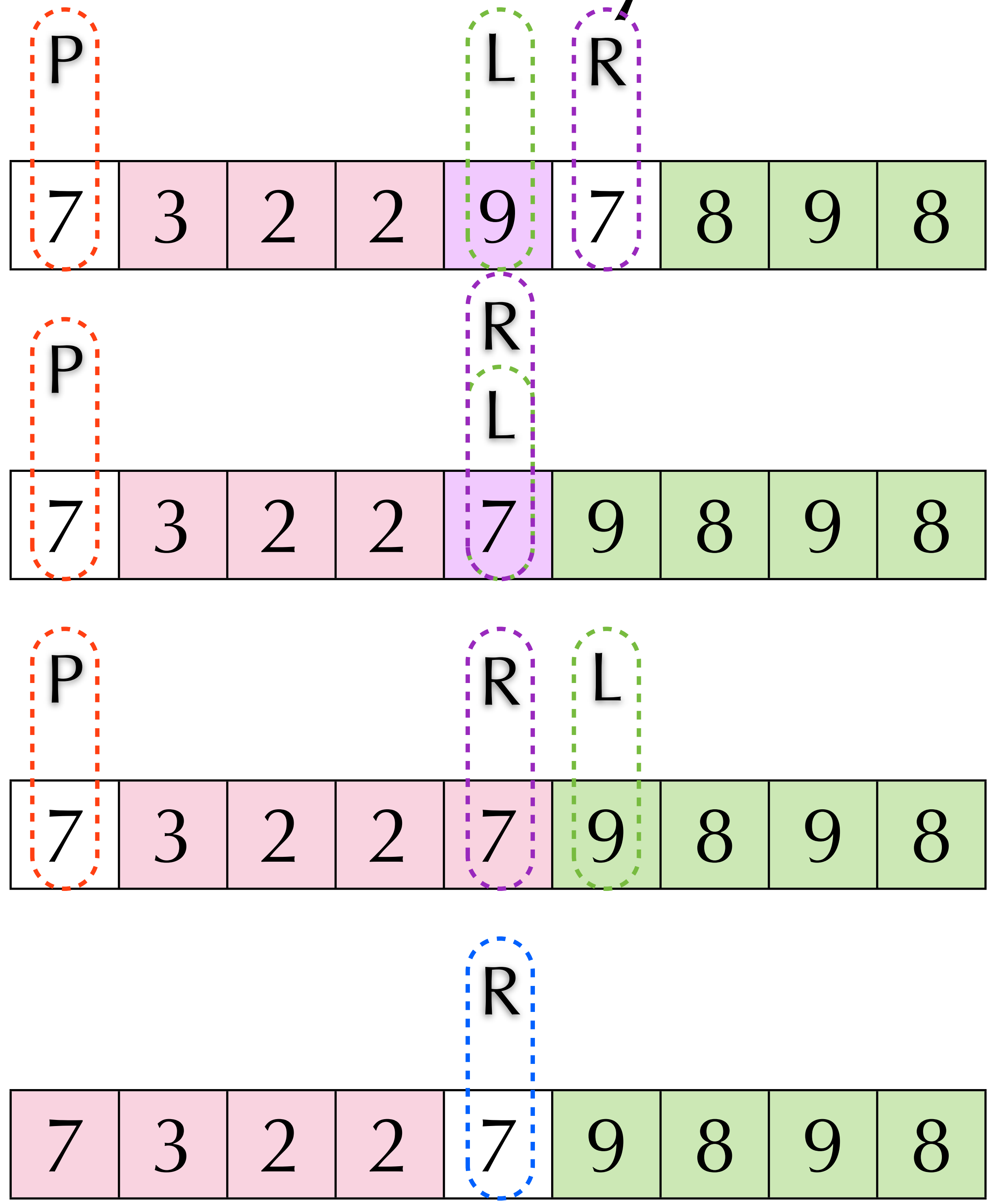


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If A[P]≥A[L]:
L=L+1

If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

Partition by Pivot

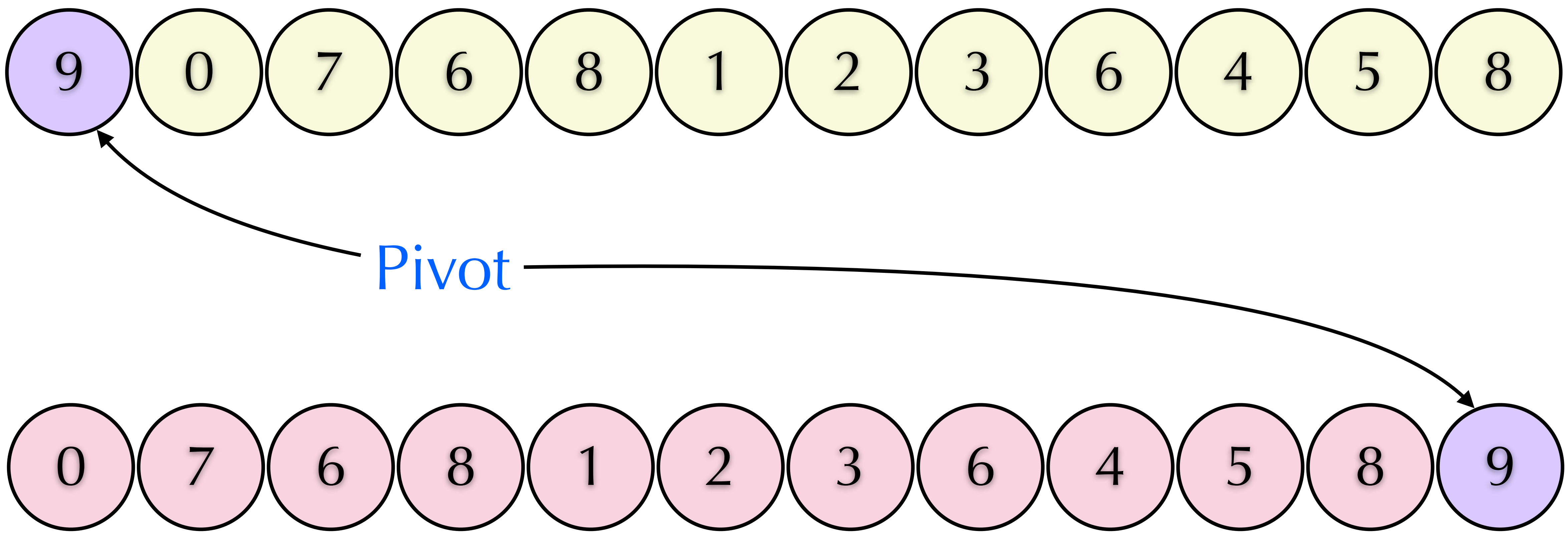


If $A[P]<A[L]$:
swap(A[L],A[R])
$R=R-1$

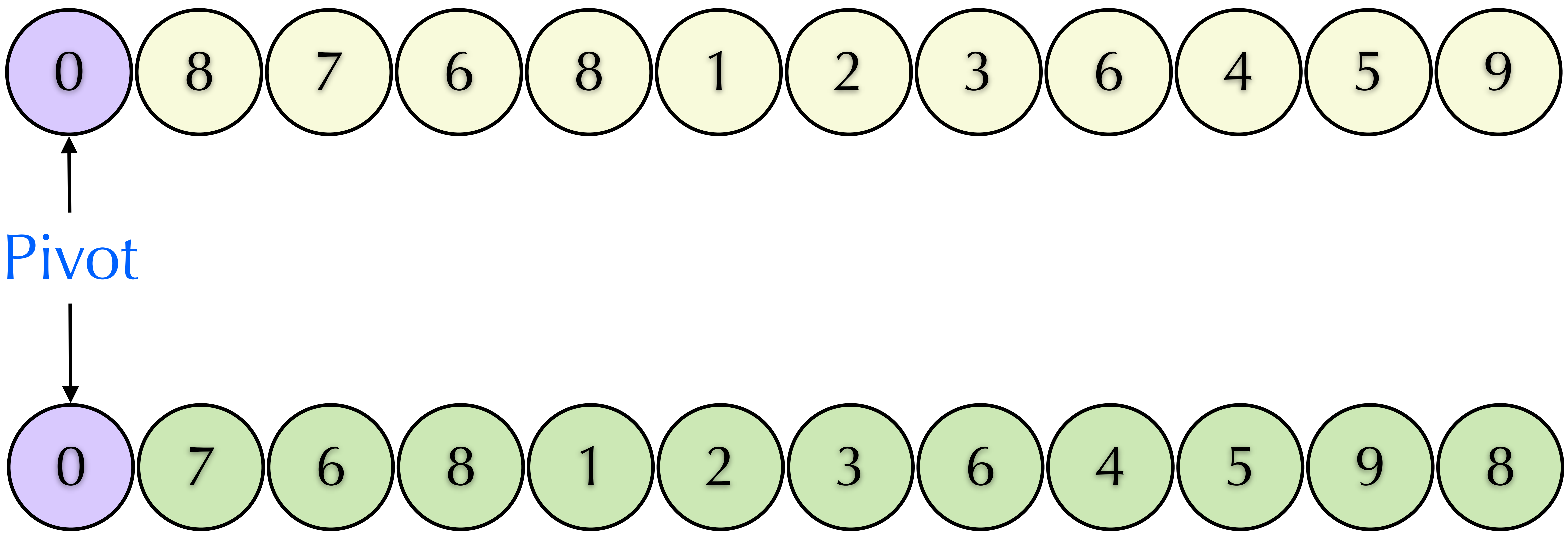
If $A[P]\geq A[L]$:
$L=L+1$

If $R<L$:
swap(A[P],A[R])
return R

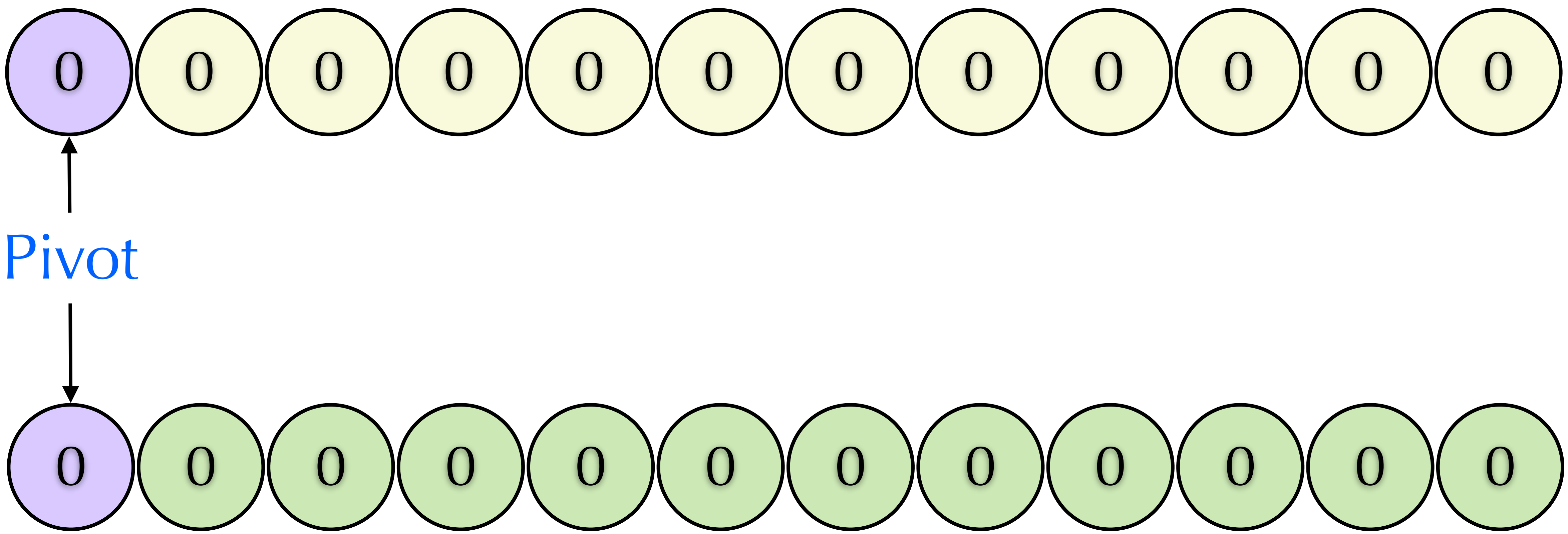
Partition: Worst Case 1



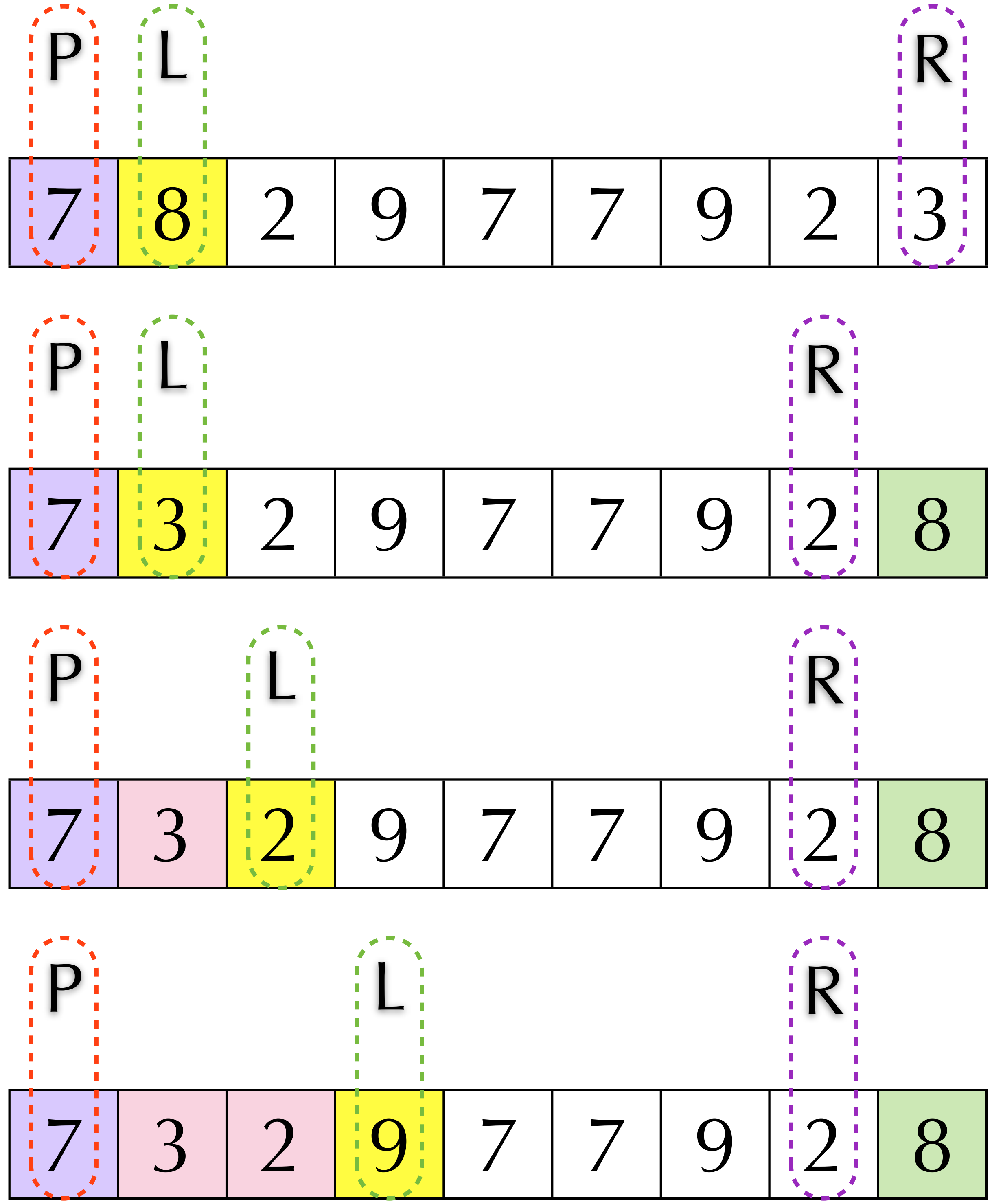
Partition: Worst Case 2



Partition: Worst Case 3



Modified Partition

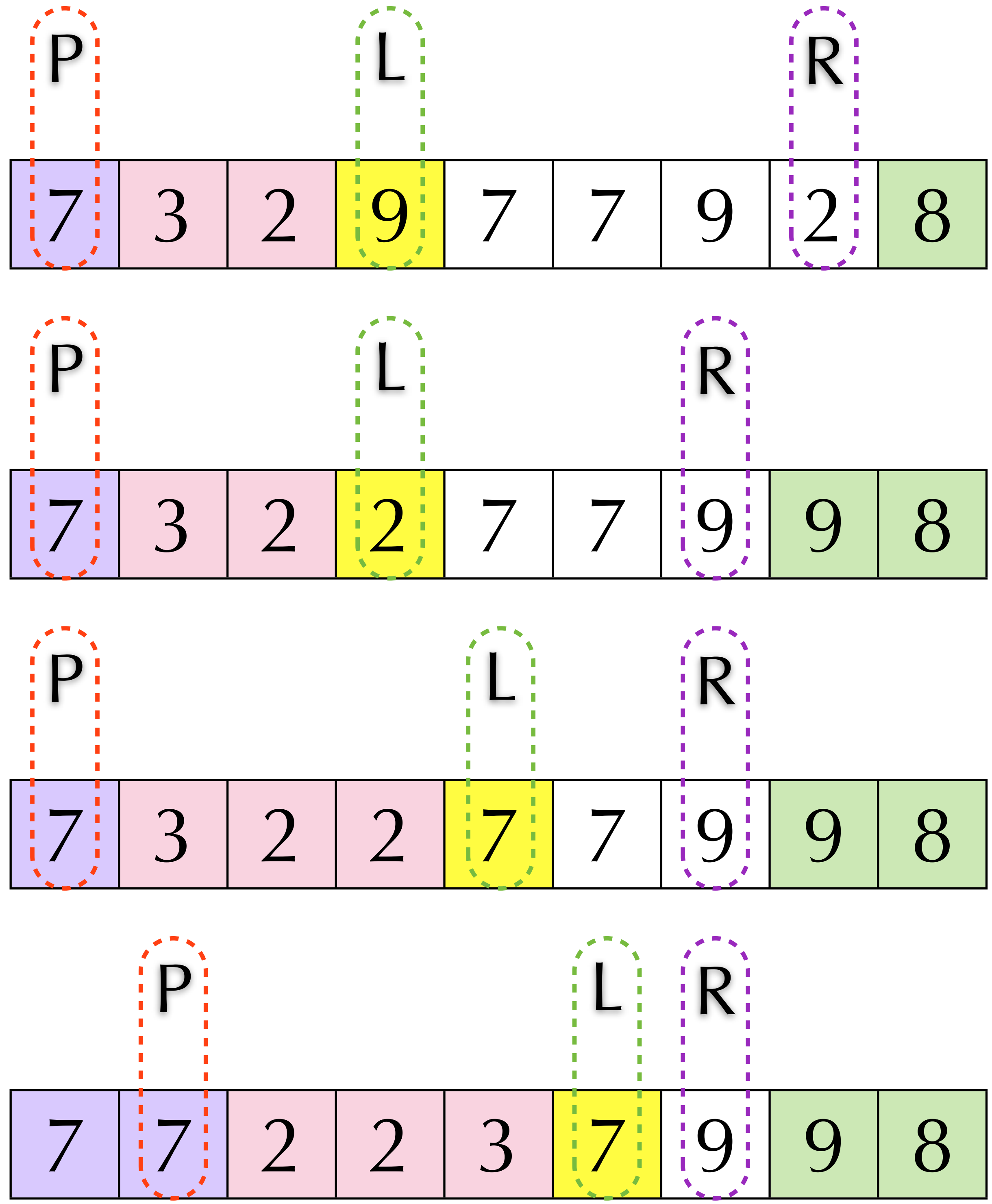


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If A[P]>A[L]:
L=L+1

If A[P]>A[L]:
L=L+1

Modified Partition

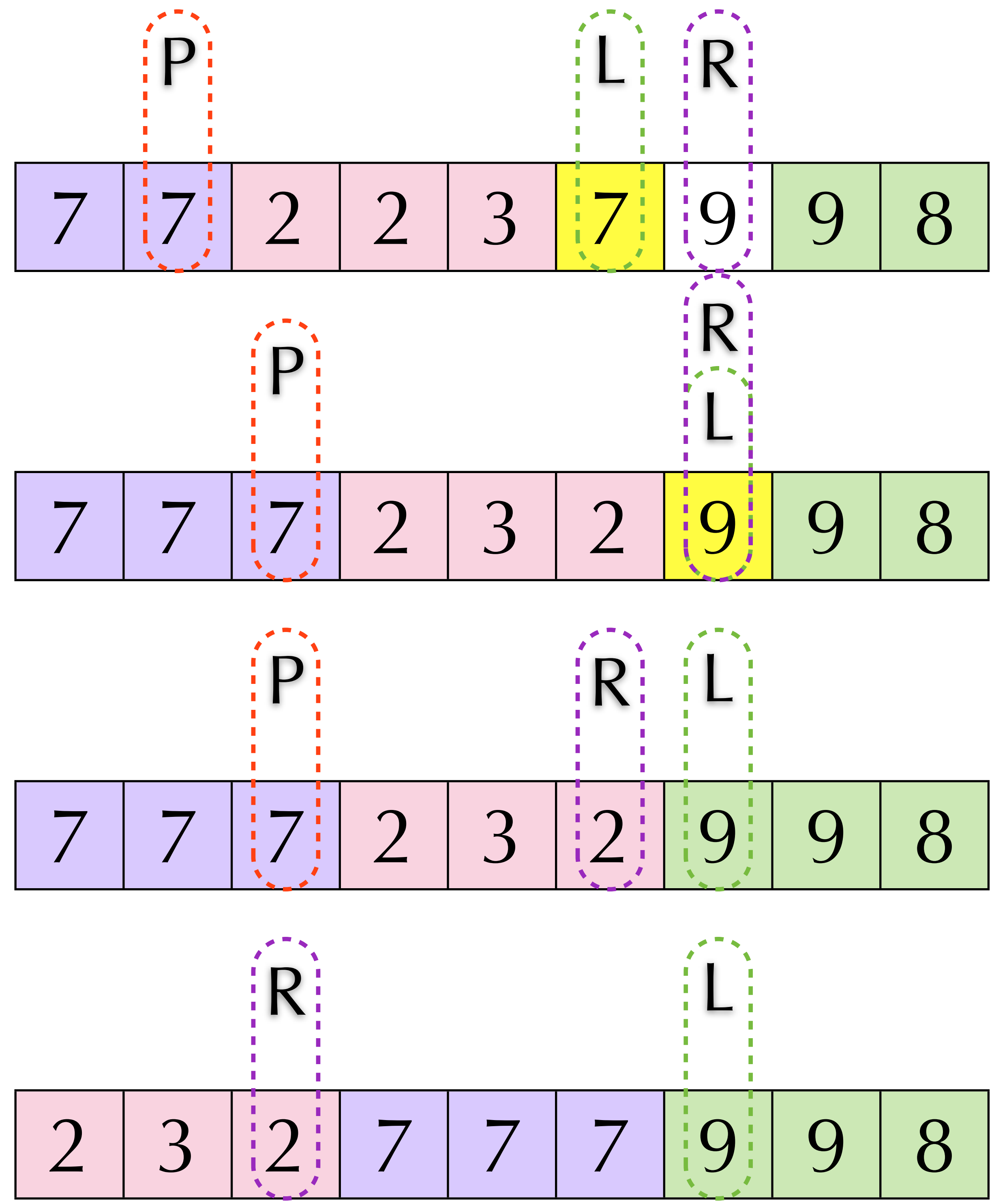


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If A[P]>A[L]:
L=L+1

If A[P]=A[L]:
P=P+1
swap(A[P],A[L])
L=L+1

Modified Partition



If A[P]=A[L]:
P=P+1
swap(A[P],A[L])
L=L+1

If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If R<L:
while(P>0)
  swap(A[P],A[R])
  P=P–1, R=R–1

qsort(A[1..R])
qsort(A[L..n])

Quick Sort

- Worst case:
  - $T(n)=T(n-1)+O(n)=O(n^2)$
- Average case:
  - What is average? The input sequence is uniformly randomly sampled.
  - $T(n)=(2/n)(T(1)+\ldots+T(n-1))+O(n)$ $=O(n\log n)$ … err, why?

Decrease and Conquer

- A special case of divide and conquer
  - There is only one subproblem.
- For example: Greatest common divisor
  - GCD(a,0)=a
  - GCD(a,b)=GCD(b,a) ... use this if a<b
  - GCD(a,b)=GCD(a−b,b)

Prune and Search

- A special case of decrease and conquer
- $T(n)=T(pn)+O(f(n))$ where $p<1$
- Example:
  - Binary search
  - Golden section search
  - Euclidean algorithm
  - Extended Euclidean algorithm