Divide and Conquer
Prune and Search

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.

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: $O(n^{\log(2k-1)/\log k})$

Karatsuba Algorithm

- Let $x = x_H B + x_L$ and $y = y_H B + y_L$ where $x_L < B$, $y_L < B$, and $y \le x < B^2$.
- $xy = x_H y_H B^2 + x_L y_H B + x_H y_L B + x_L y_L$.
- 4 subproblems $x_H y_H$, $x_L y_H$, $x_H y_L$, $x_L y_L$.
- $T(n) = 4T(n/2) + O(n) = O(n^2)$
- 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

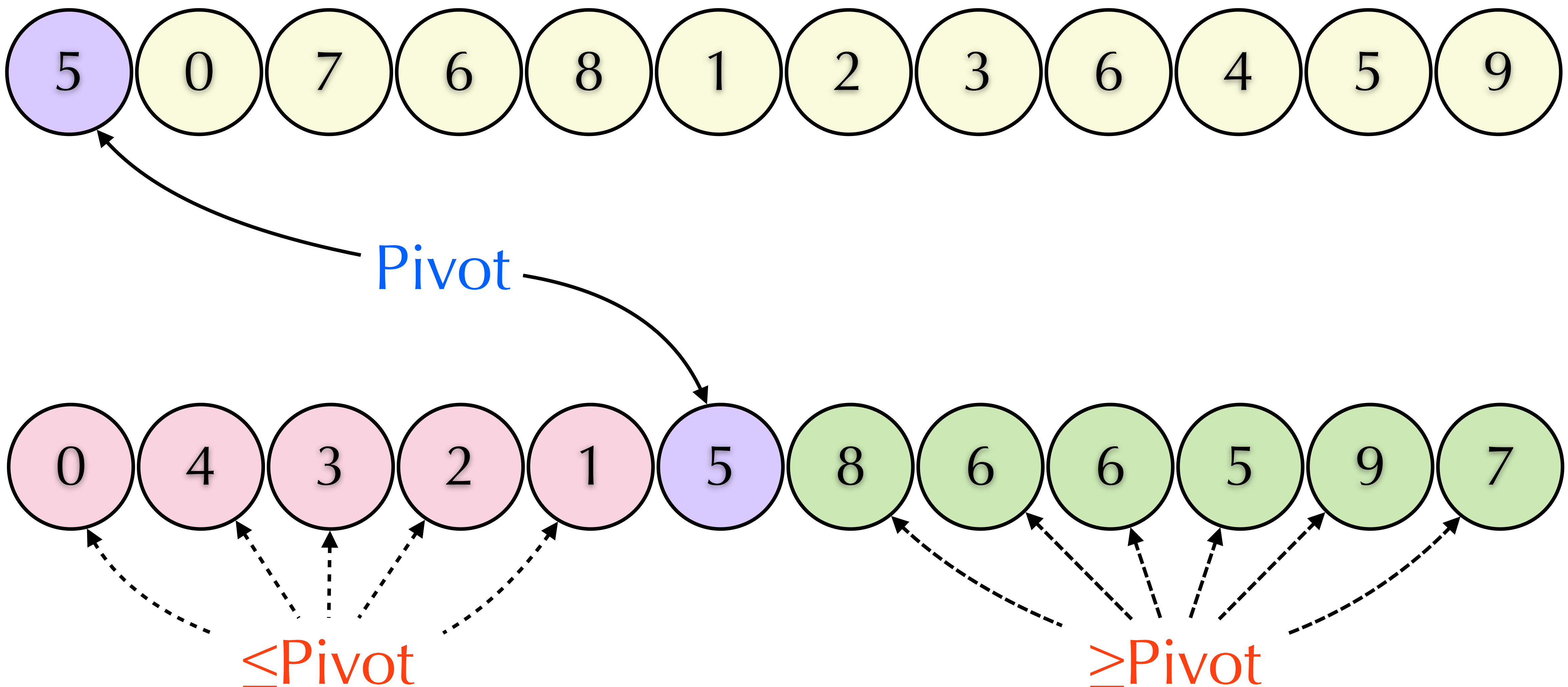
- Termination: If x and y are small, multiply them by long multiplication.
- Divide: $x_H \times y_H$, $x_L \times y_L$, $(x_H + x_L) \times (y_H + y_L)$
- Conquer: Solve the subproblems
- Combine: Let $z_H = x_H y_H$, $z_L = x_L y_L$, $z_M = (x_H + x_L) \times (y_H + y_L) - z_H - z_L$, and $x \times y = z_H B^2 + z_M B + z_L$.
- Time: $T(n) = 3T(n/2) + O(n) = O(n^{1.59})$

Quick Sort

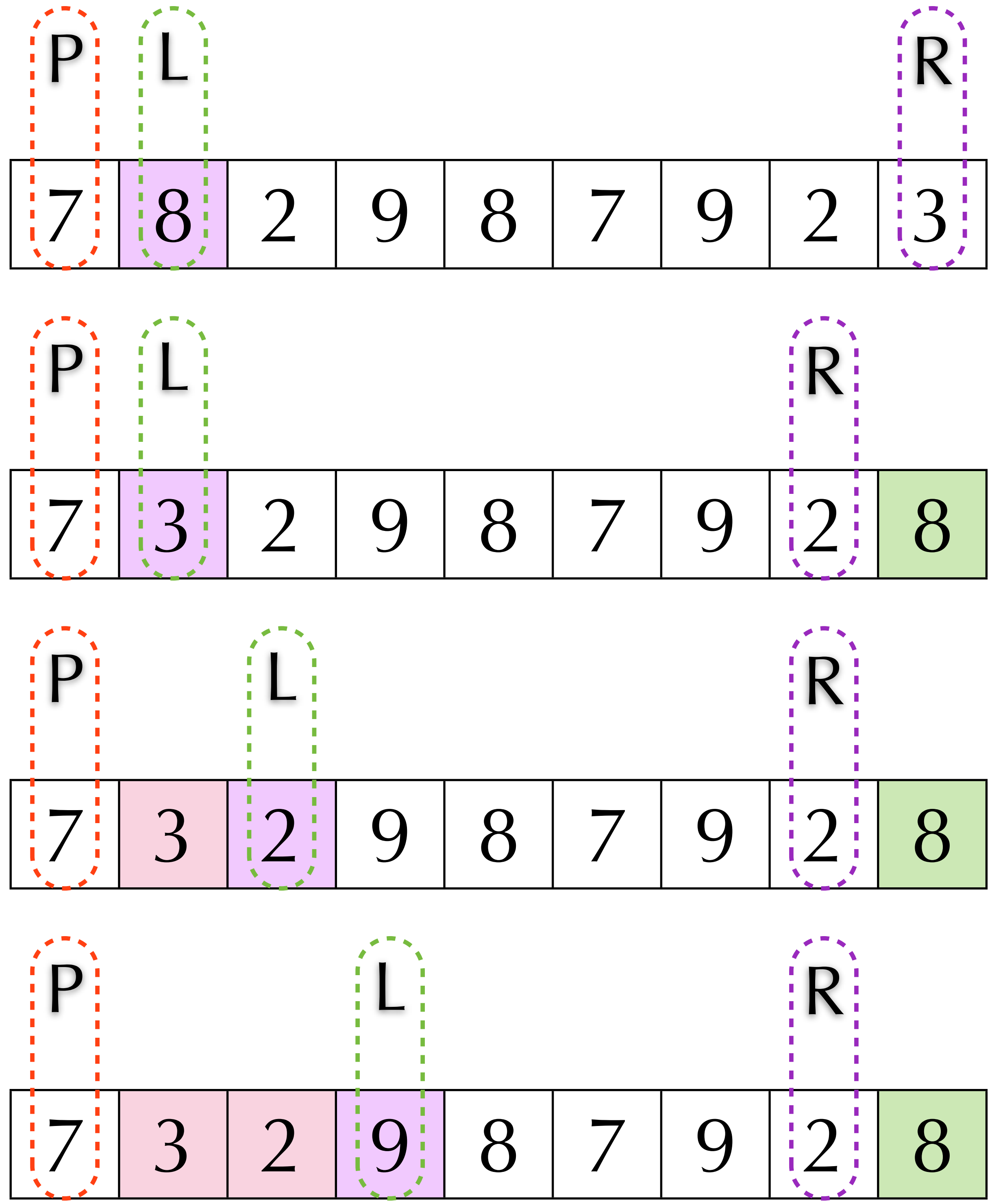
$\boxed{\text{Sort } A[1],\ldots,A[n]}$

- 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



Partition

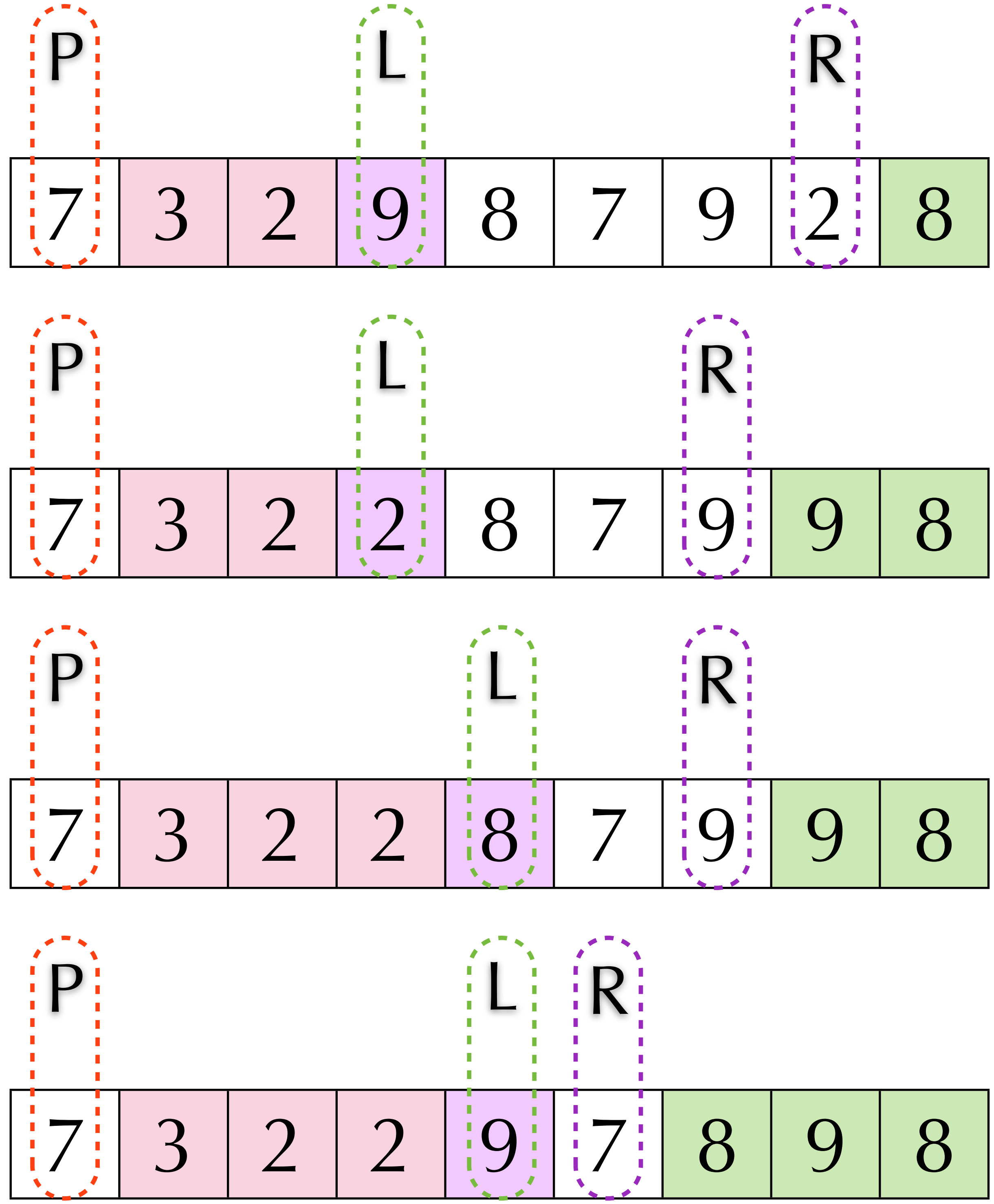


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If A[P]≥A[L]:
L=L+1

If A[P]≥A[L]:
L=L+1

Partition

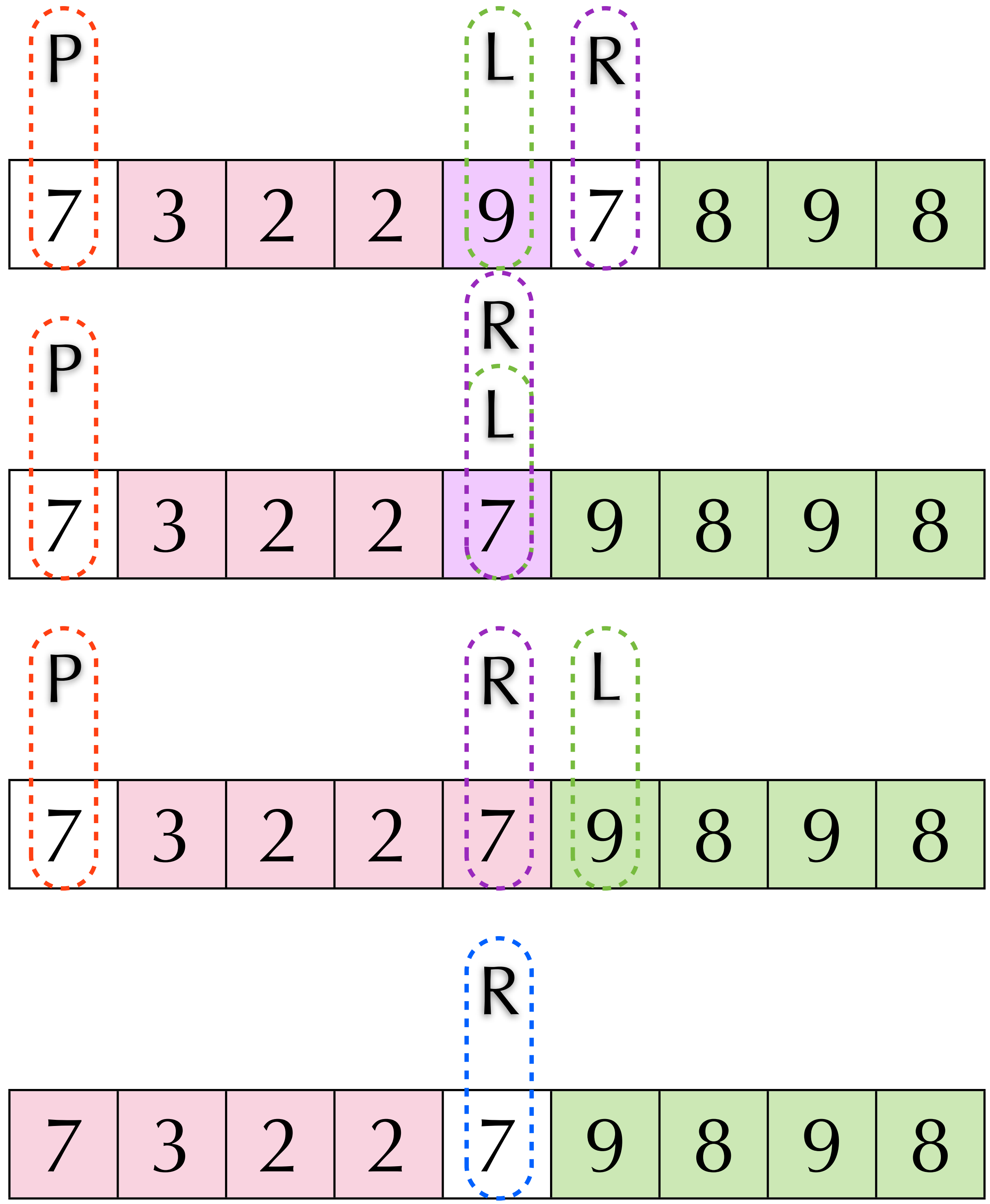


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

If A[P]≥A[L]:
L=L+1

If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

Partition

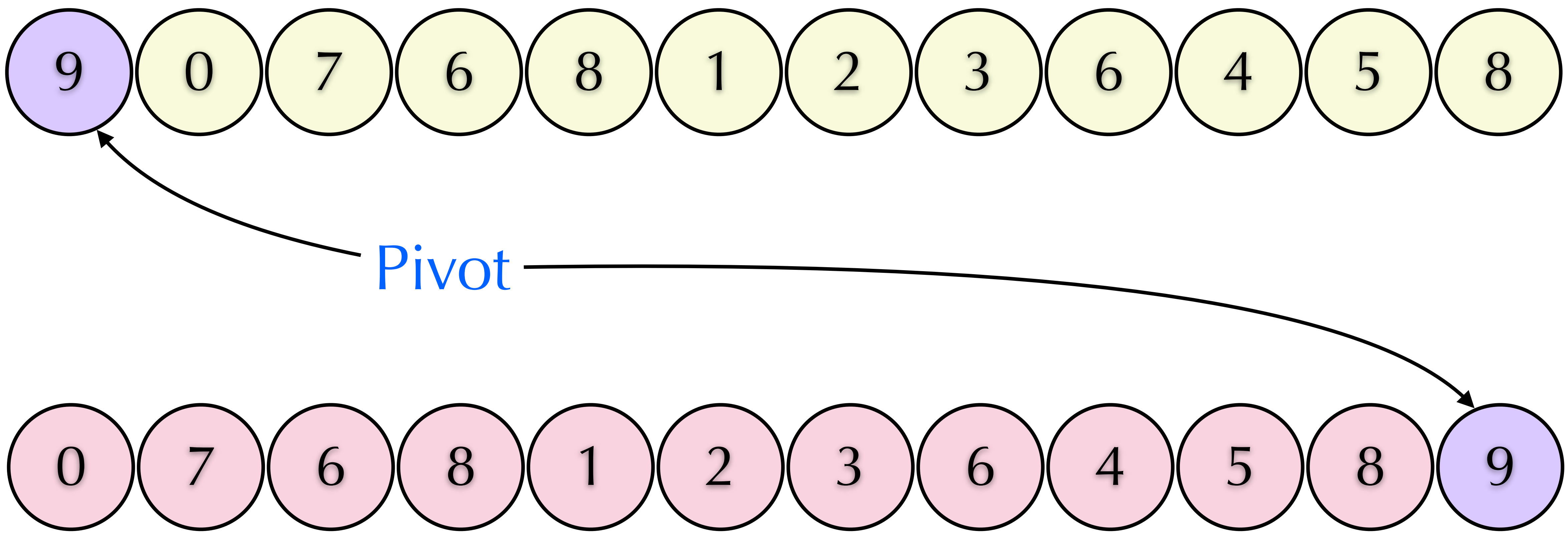


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

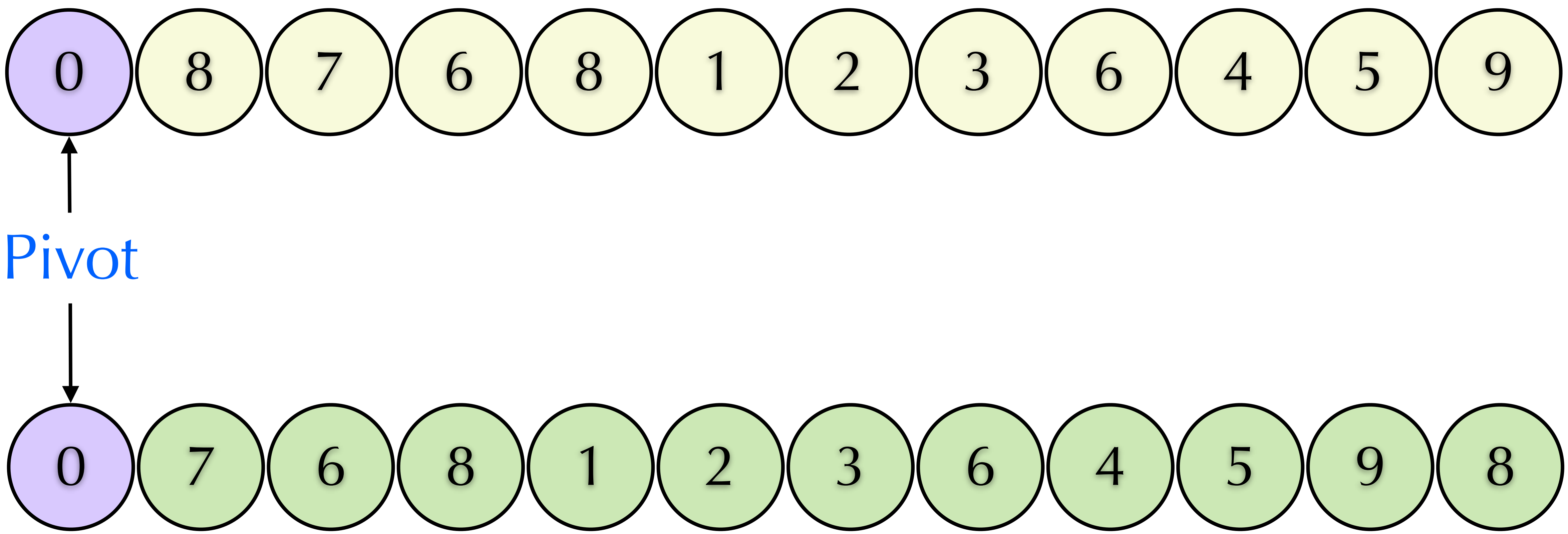
If A[P]≥A[L]:
L=L+1

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

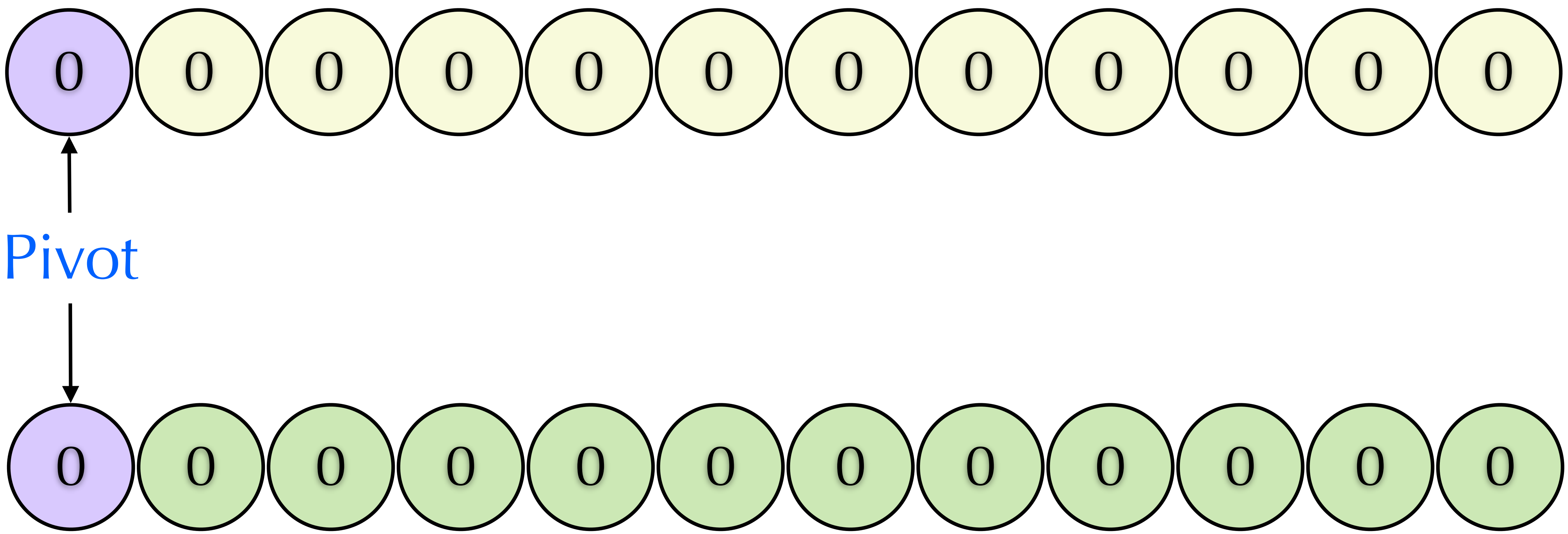
Partition: Worst Case 1



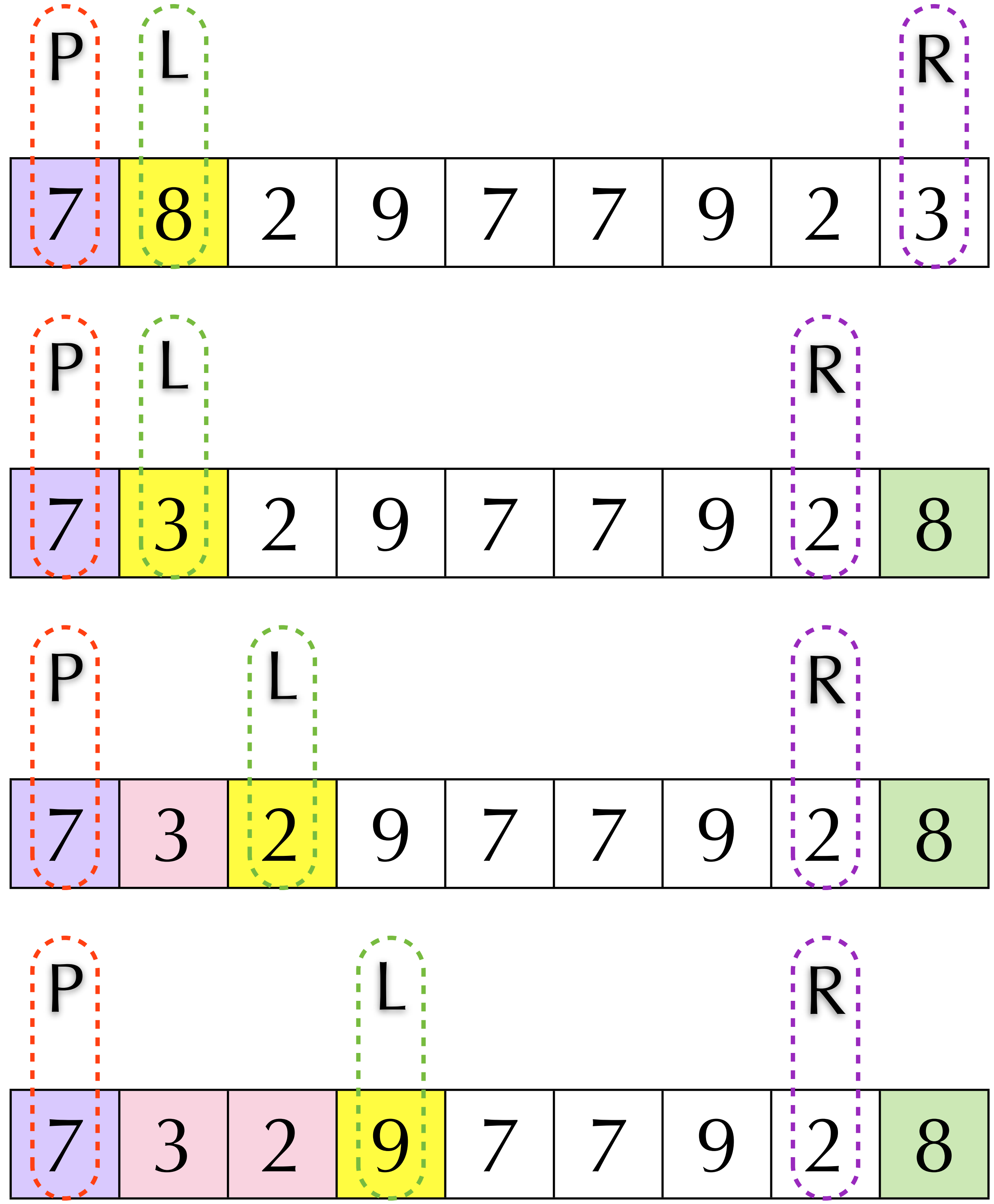
Partition: Worst Case 2



Partition: Worst Case 3



Modified Partition

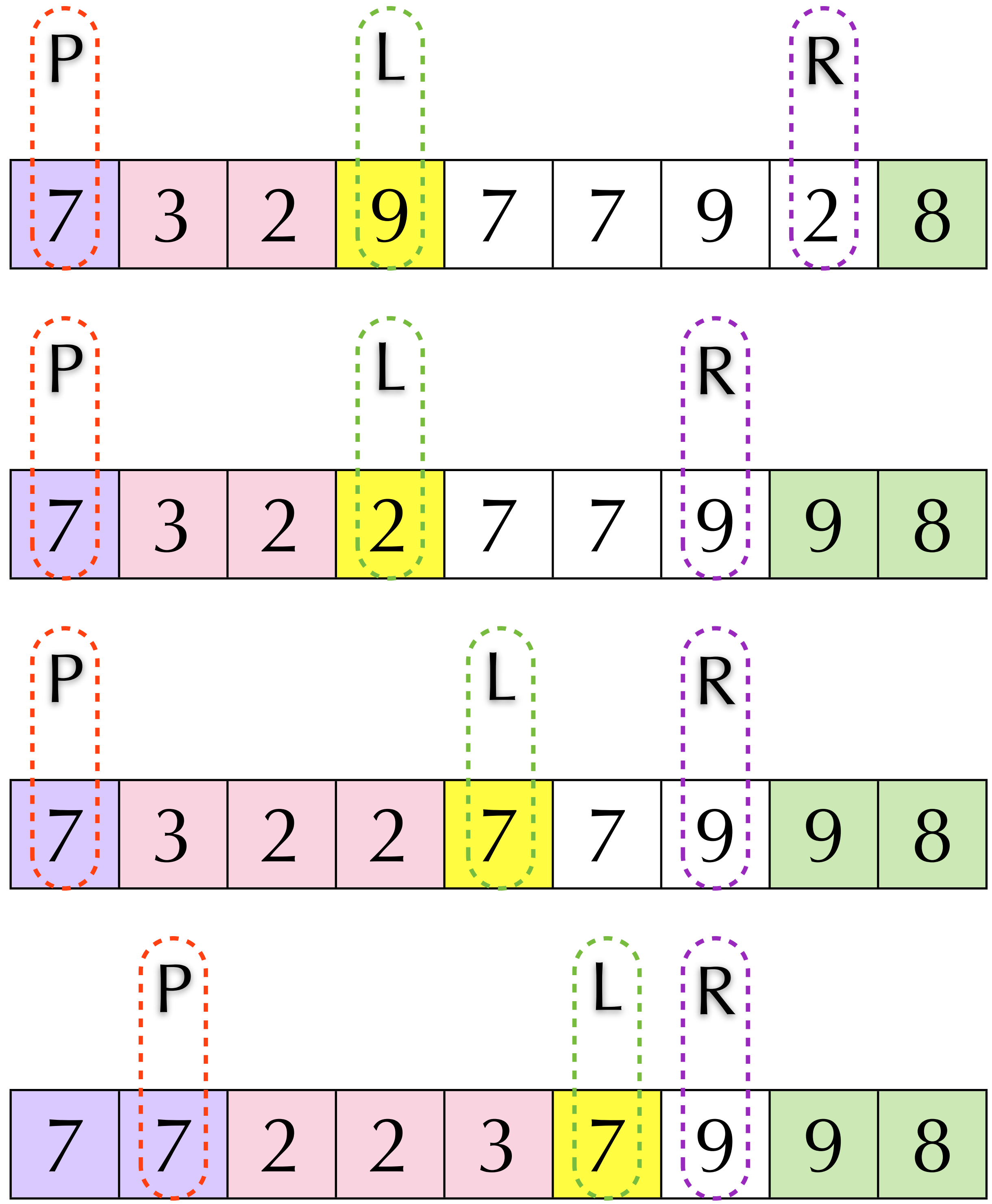


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

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

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

Modified Partition

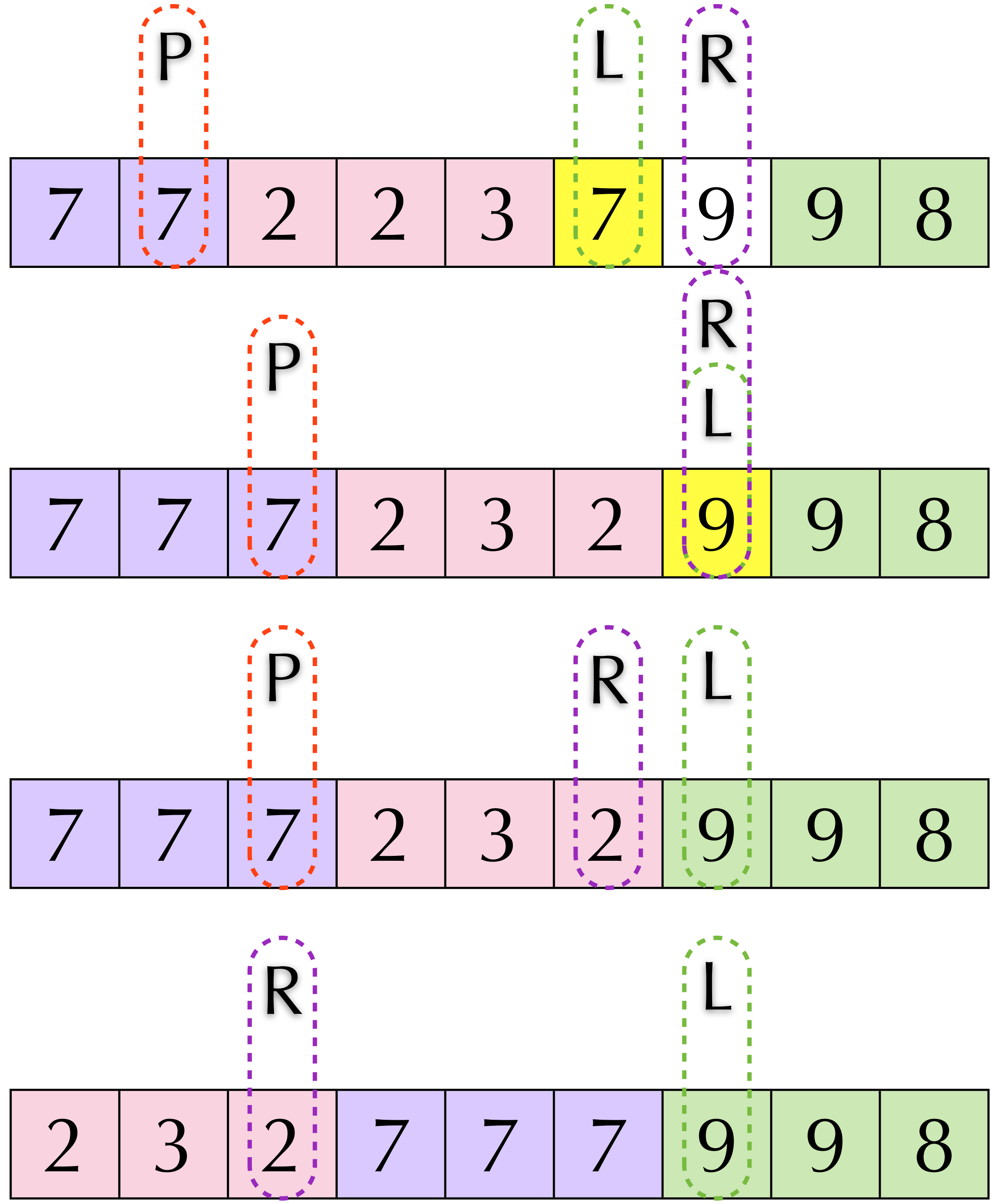


If A[P]<A[L]:
swap(A[L],A[R])
R=R–1

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])

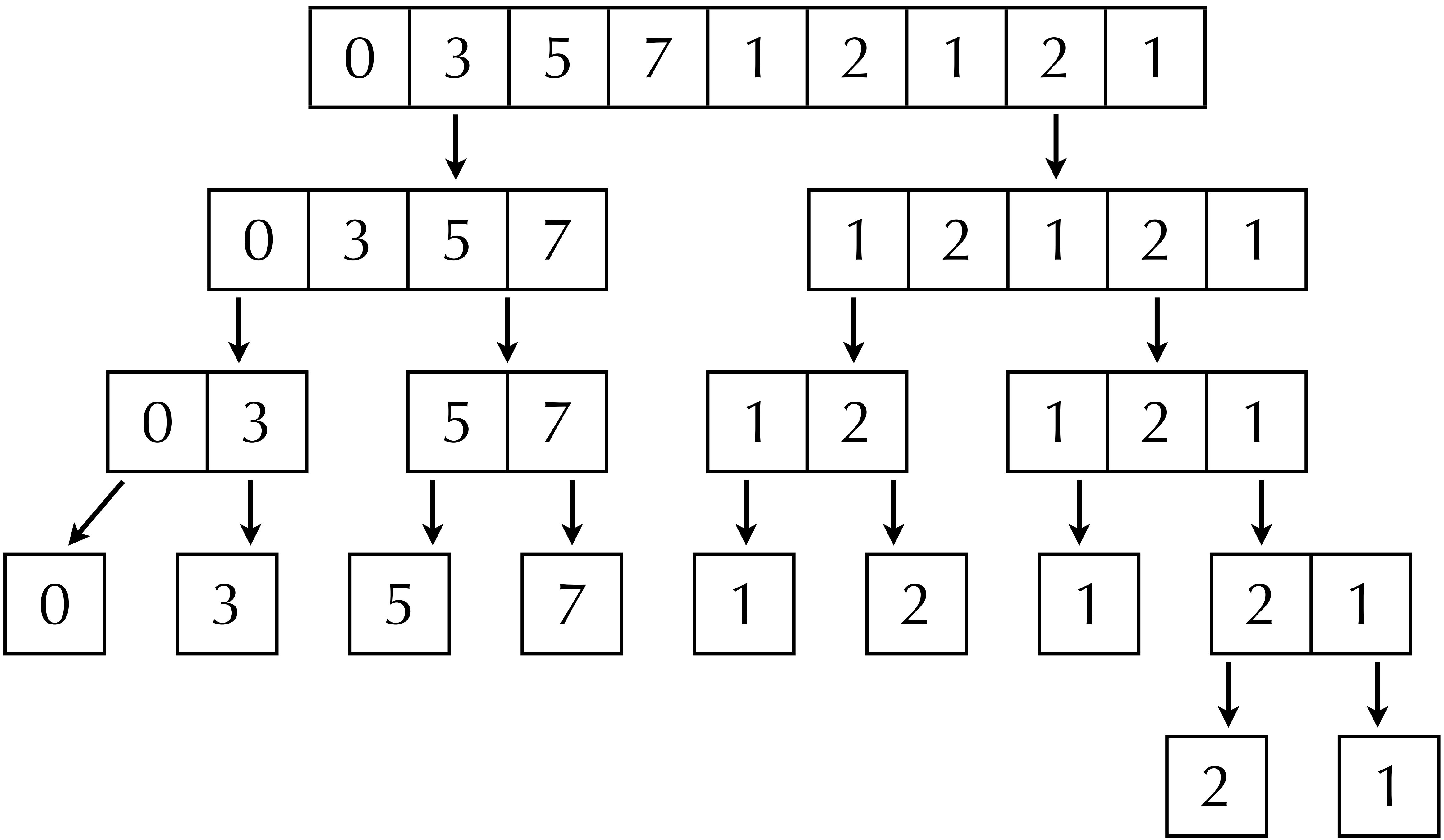
Quick Sort

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

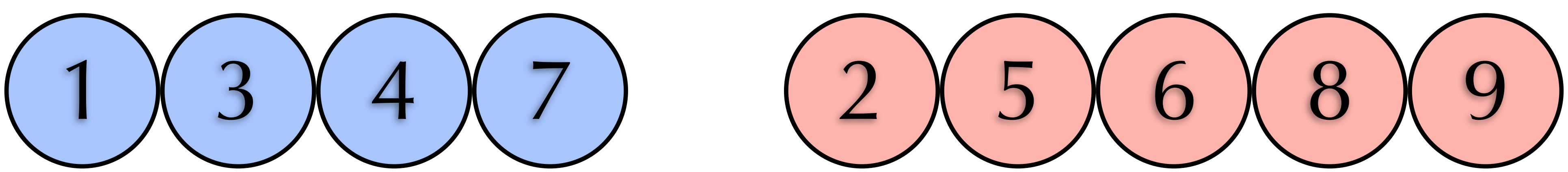
Merge Sort

- Input: $\langle a_1,\ldots,a_n\rangle$
- Termination: n=1. $\langle a_1\rangle$ is sorted.
- Divide: split $\langle a_1,\ldots,a_n\rangle$ into $\langle a_1,\ldots,a_{n/2}\rangle$ and $\langle a_{1+n/2},\ldots,a_n\rangle$.
- Conquer: Sort $\langle a_1,\ldots,a_{n/2}\rangle$ and $\langle a_{1+n/2},\ldots,a_n\rangle$
- Combine: Merge two sorted lists into a sorted list $\langle b_1,\ldots,b_n\rangle$

Merge Sort

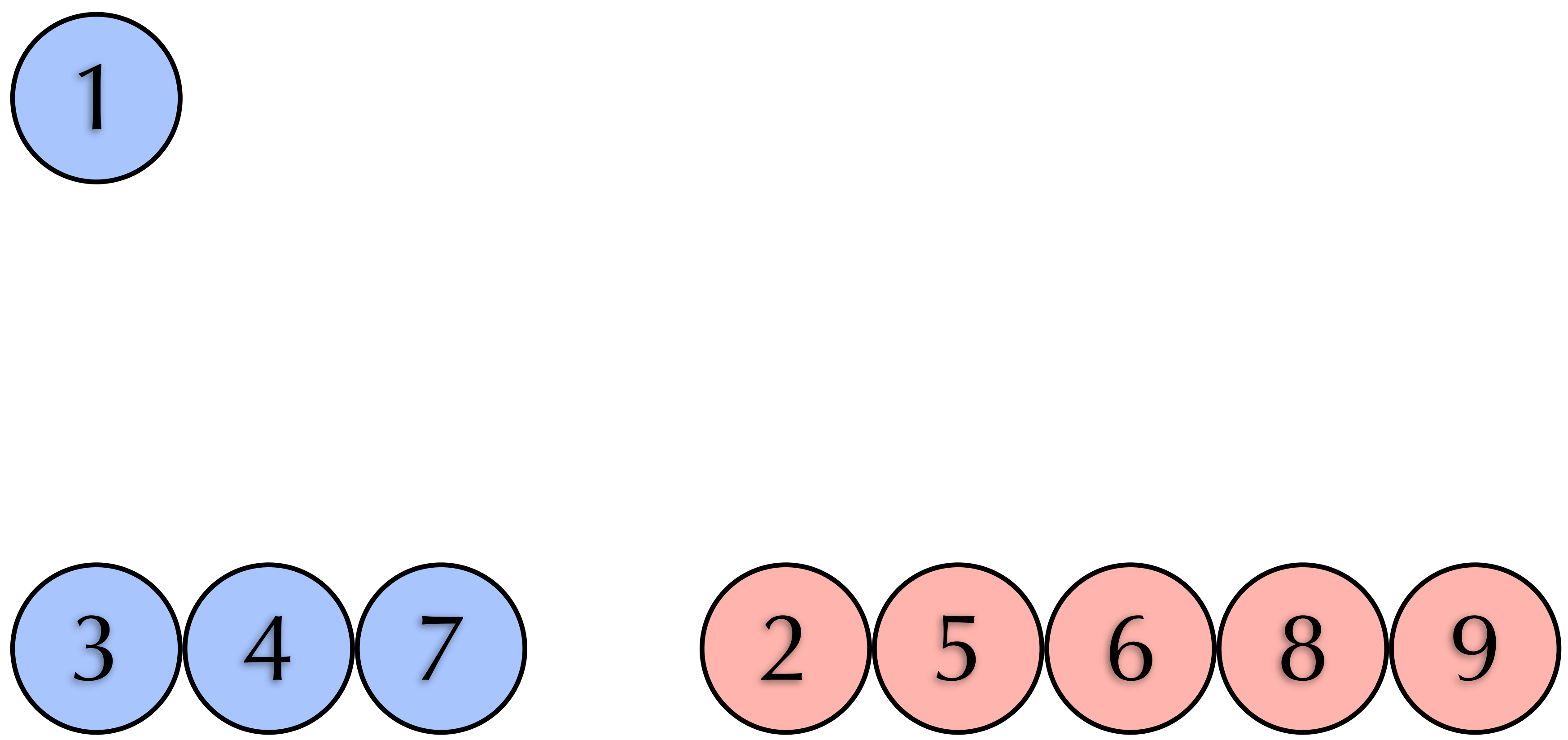


Merge



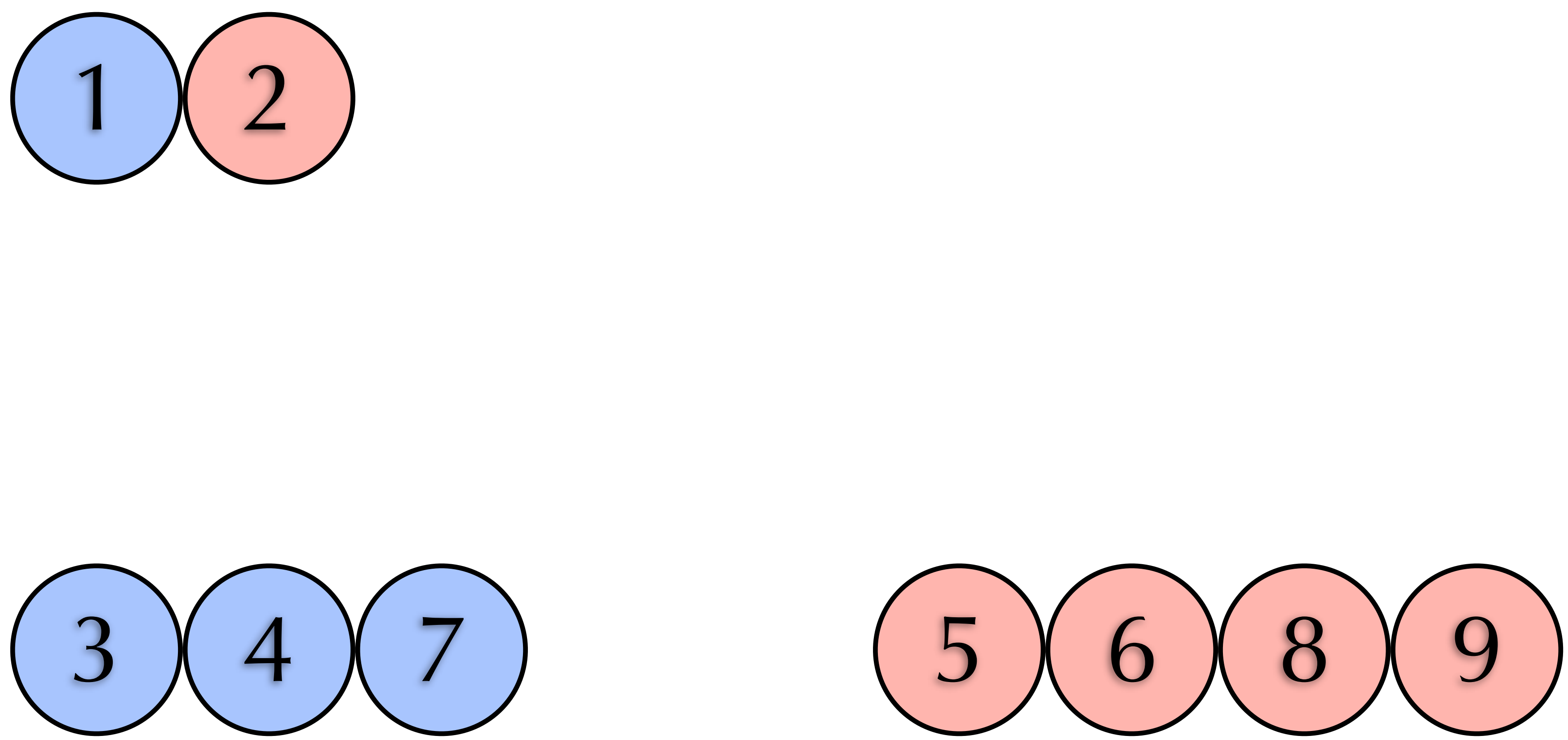
1<2

Merge



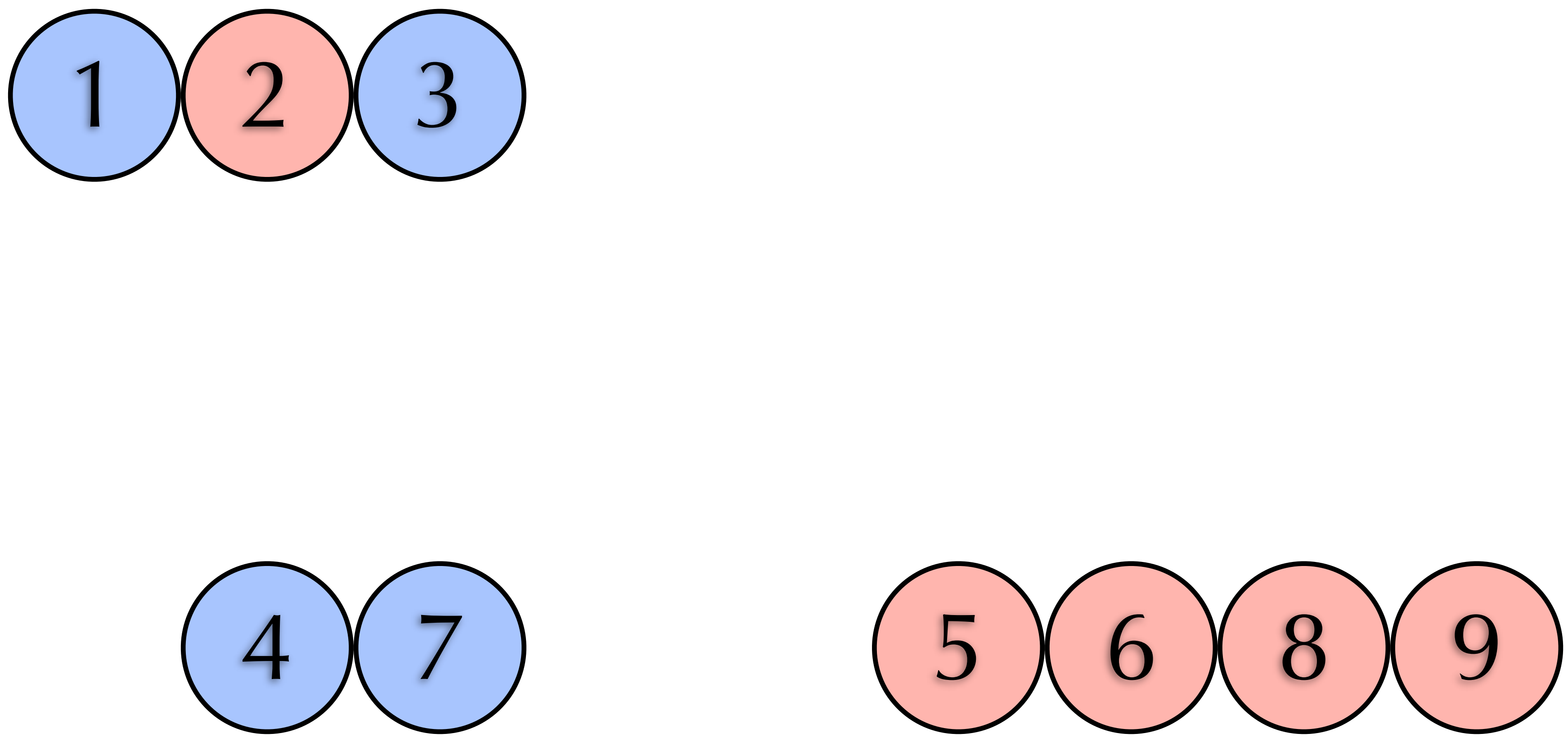
3>2

Merge



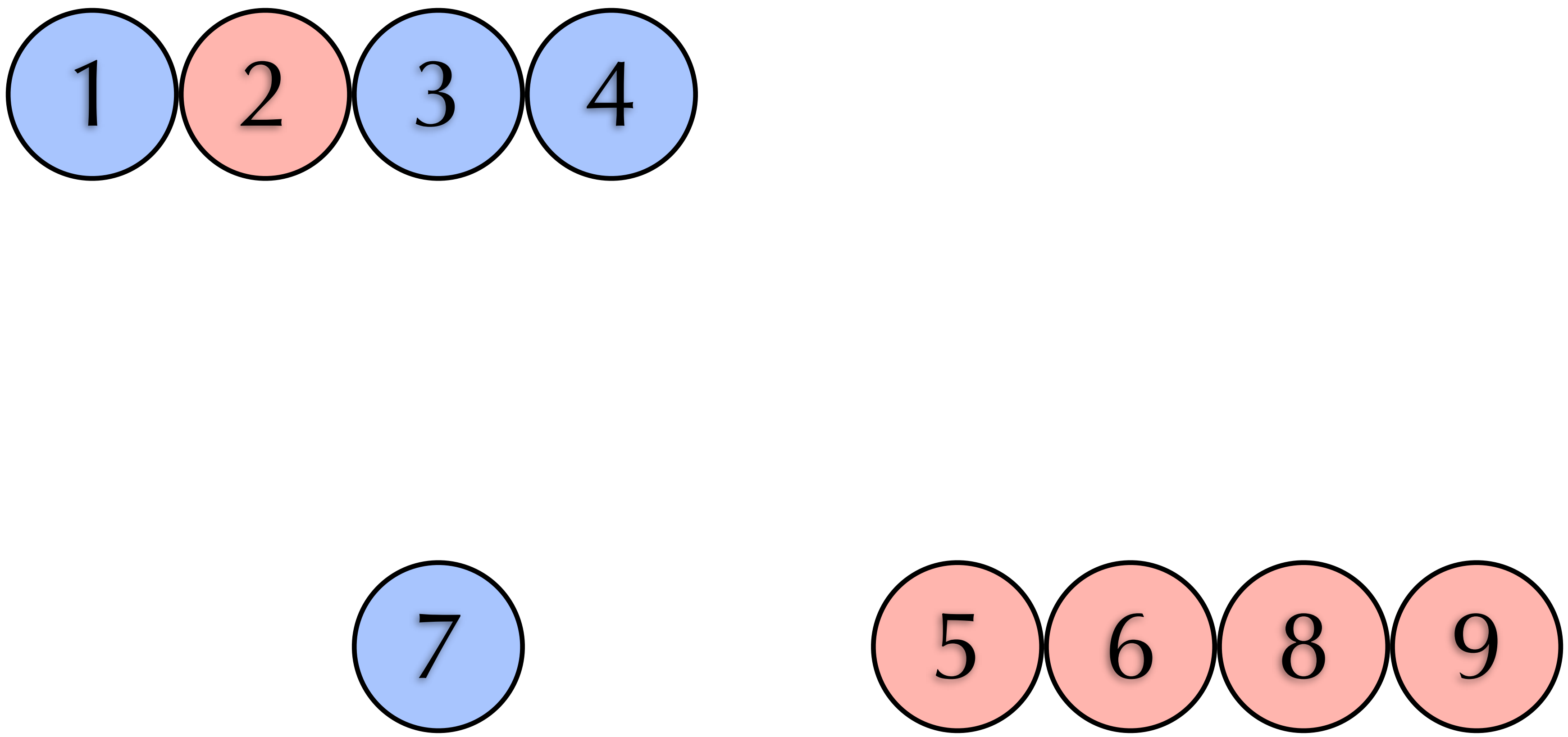
$3<5$

Merge



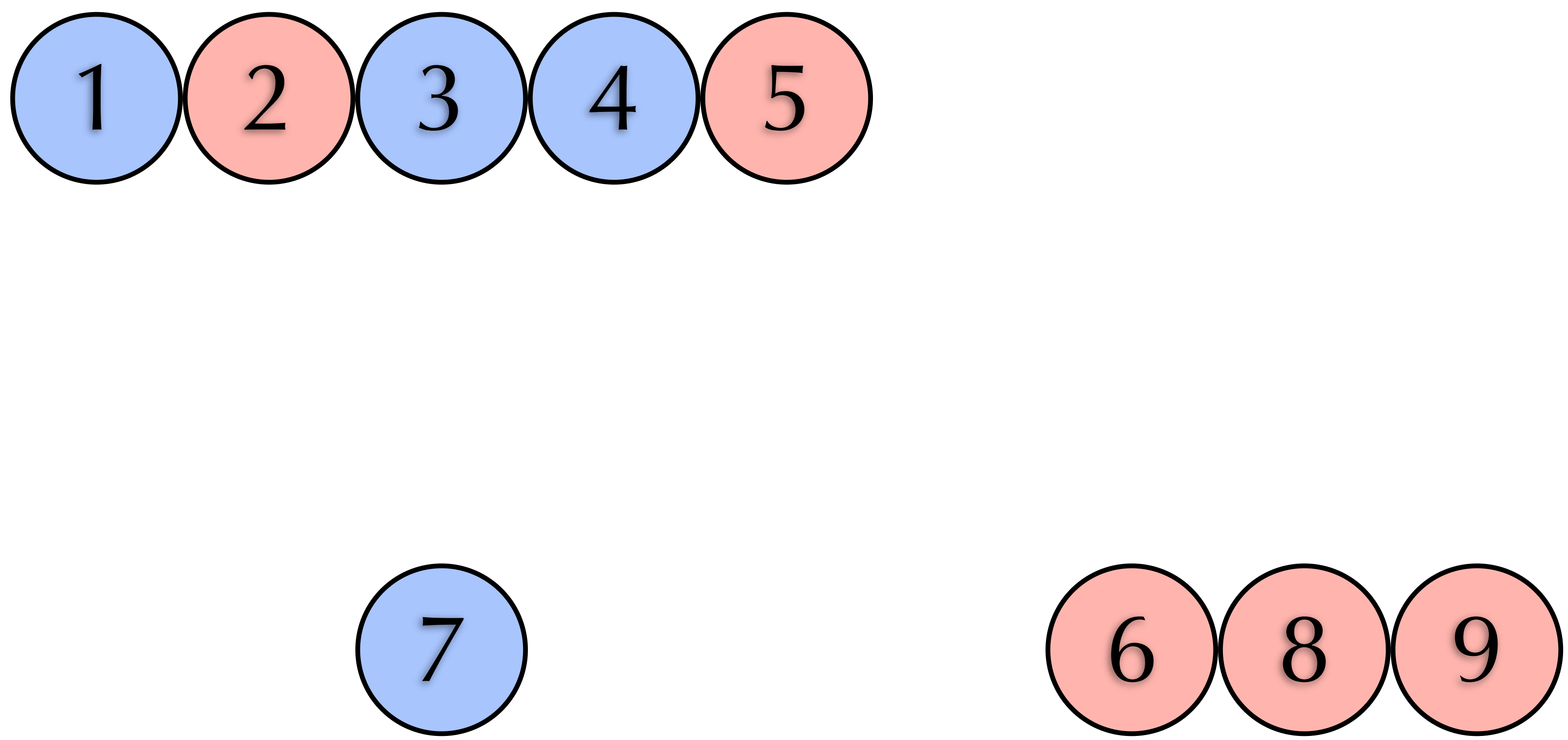
4<5

Merge



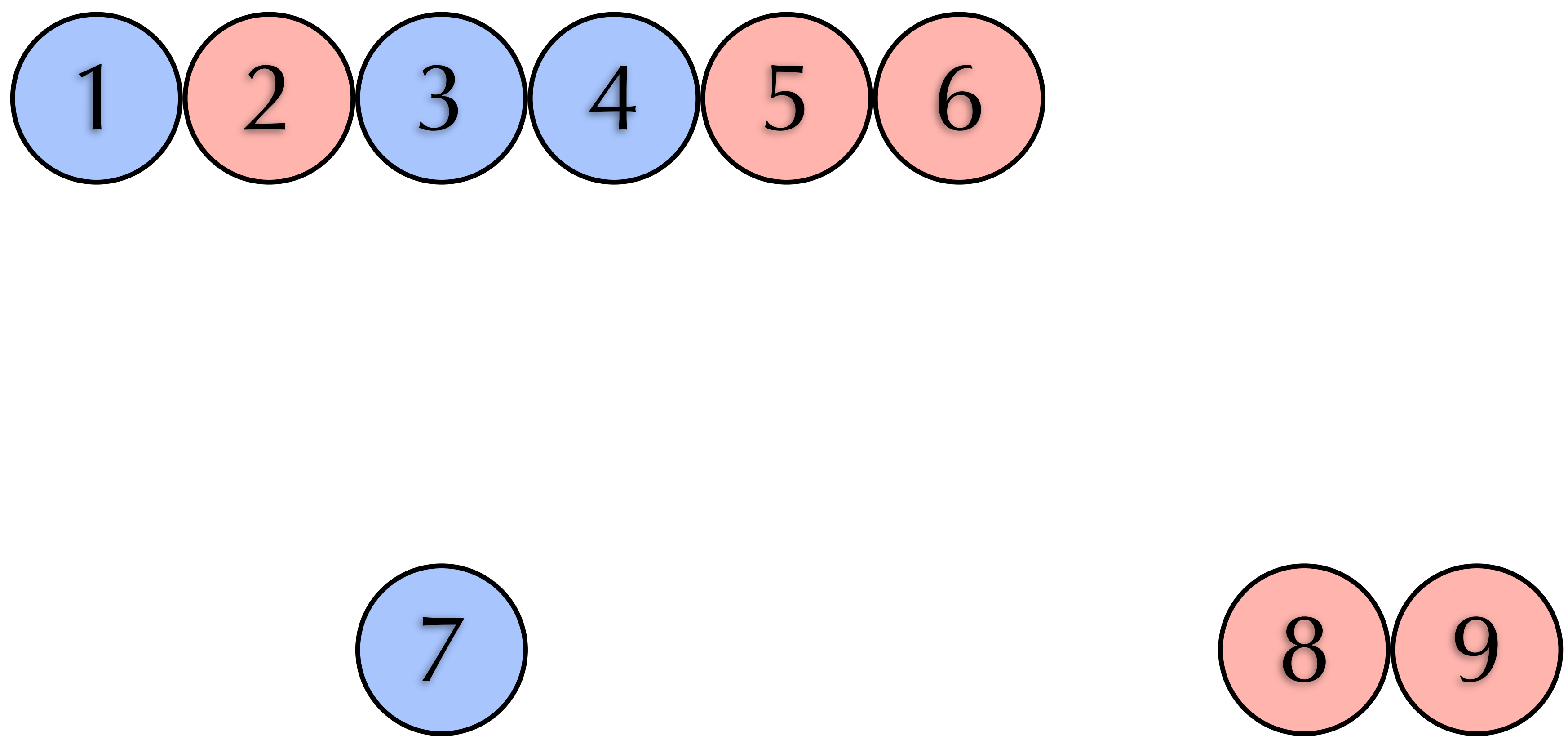
7>5

Merge



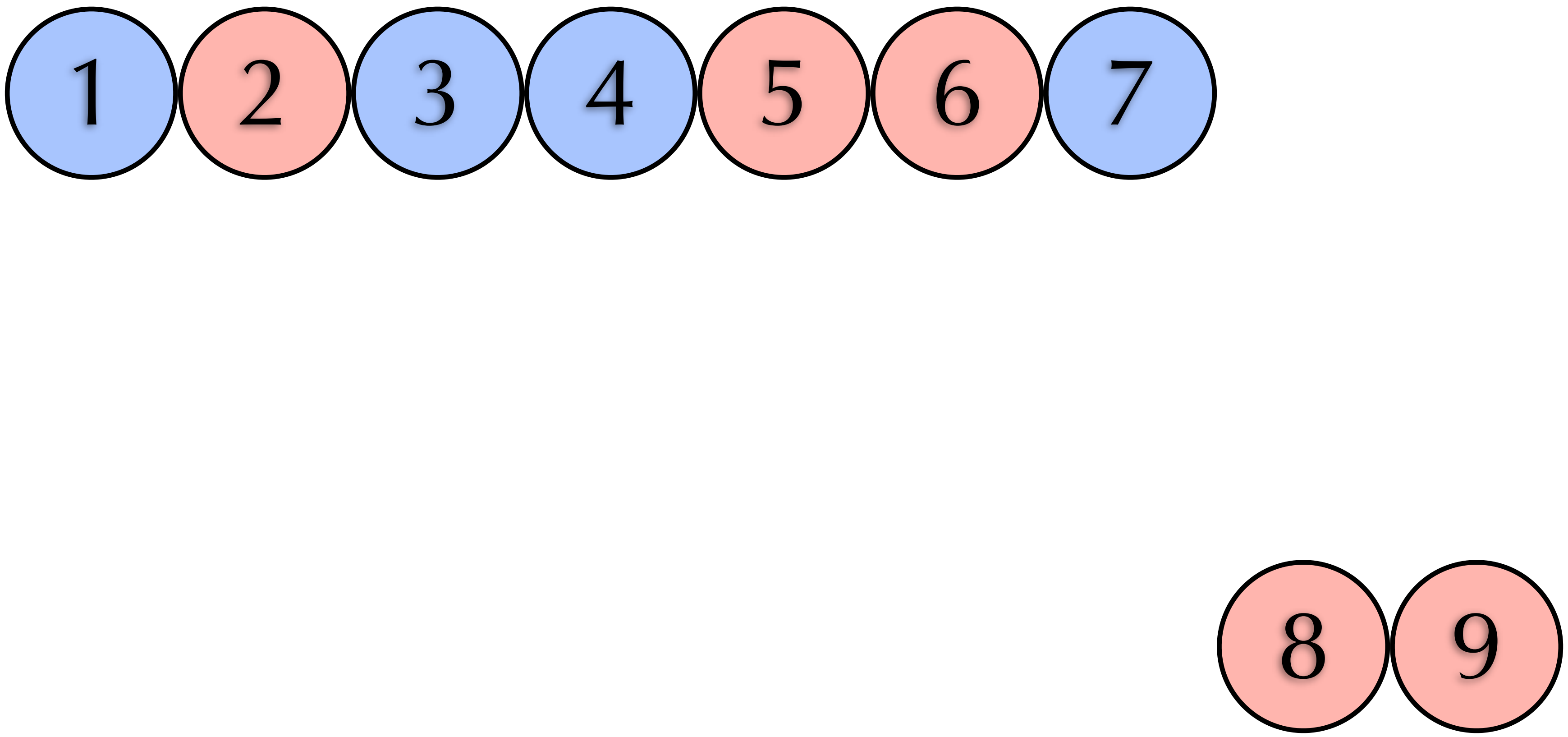
7>6

Merge



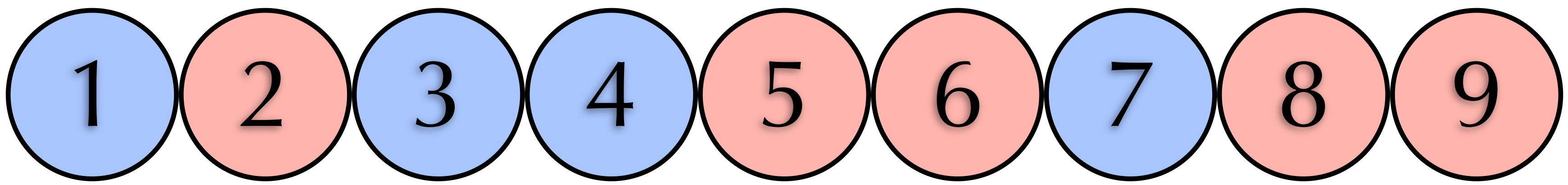
7<8

Merge

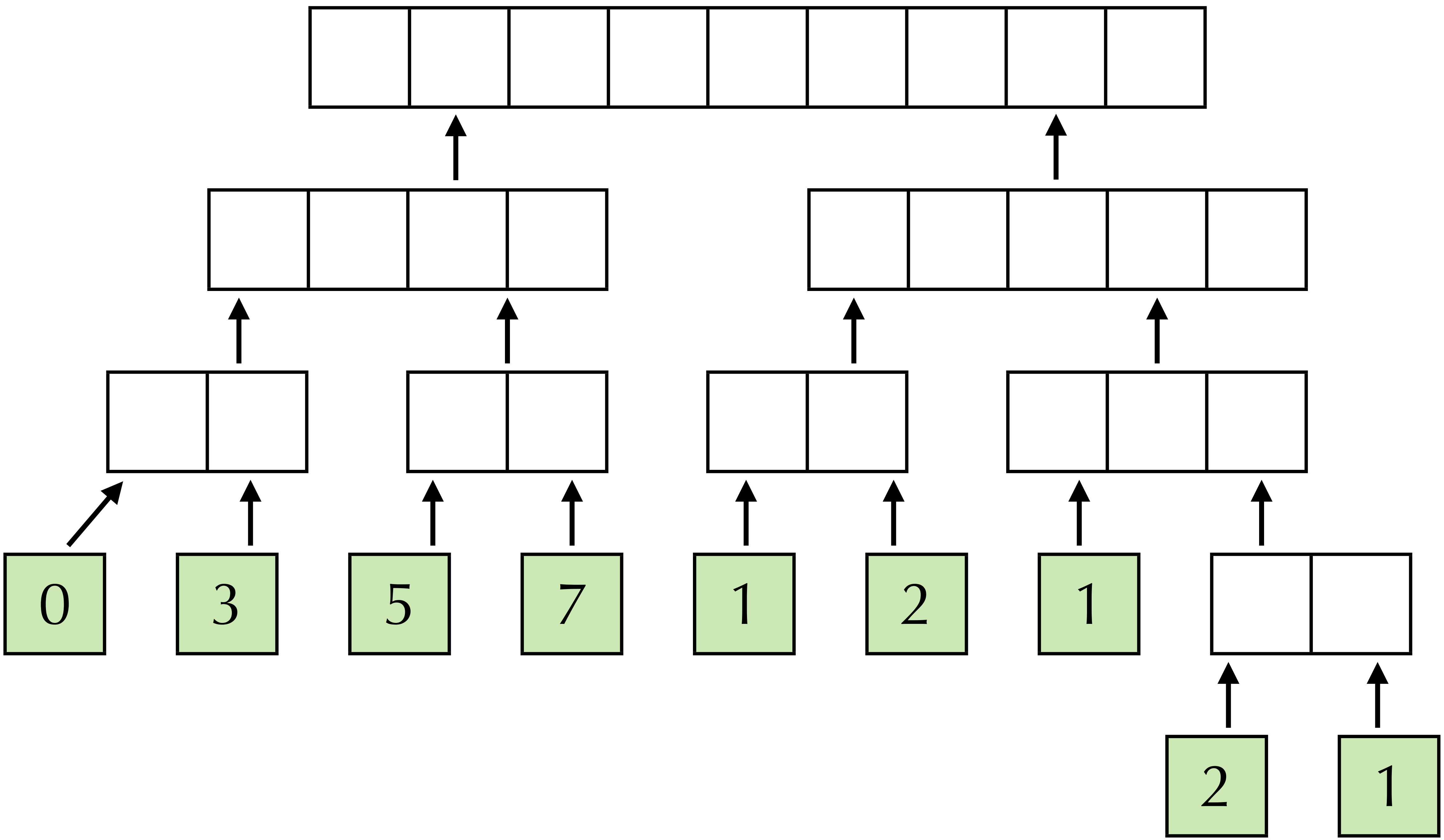


Left half is empty!

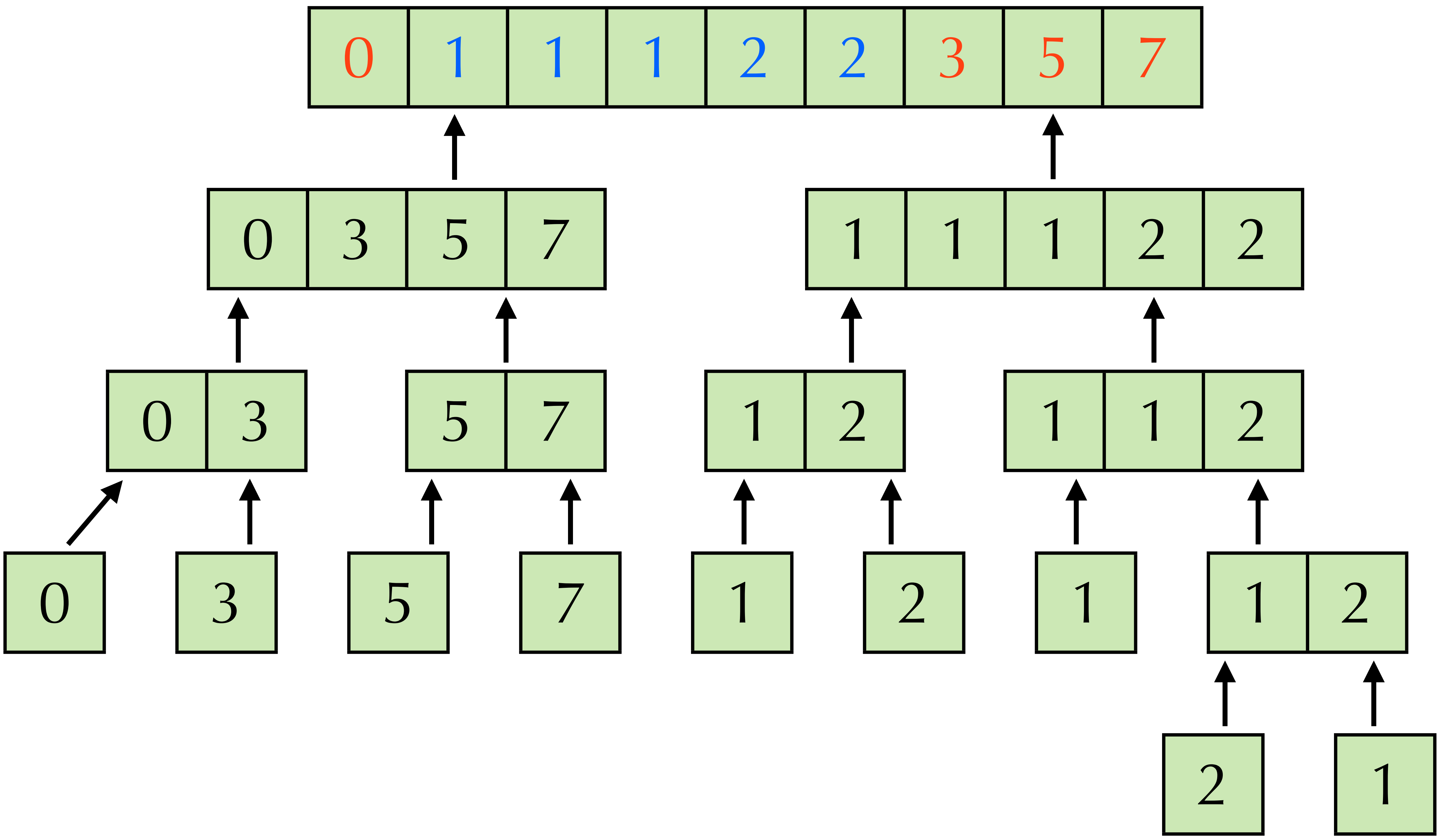
Merge



Merge Sort



Merge Sort



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
  - Bisection method
  - Golden section search
  - Extended Euclidean algorithm

Binary search

- Given x and a sorted array A[1..n].
- If $x \in A$, then find out k such that $x = A[k]$.
- If $x \notin A$, then find out k such that $A[k-1] < x < A[k]$.
(Suppose $A[0] = -\infty$ and $A[n+1] = \infty$.)

Binary search

- Strategy:
  - If n=1, then return if A[1]=x.
  - Suppose there are n elements in A, check if A[n/2]=x.
  - If A[n/2]≠x, then check if A[n/2]<x.
    - Y: k=bSearch(A[0..(n/2)],x)
    - N: k=n/2+bSearch(A[(n/2)+1..(n+1)],x)
- Can be done in O(logn)
- Iterative implementation?

Find 101

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
L							M								R

Find 101

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
							L				M				R

Find 101

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
							L		M		R				

Find 101

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
							L		M		R				

Find 101

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199

<101	101	>101

Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
L							M								R

Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
							L				M				R

Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
							L		M		R				

Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
							L	M	R						

Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199	∞
								L	R						

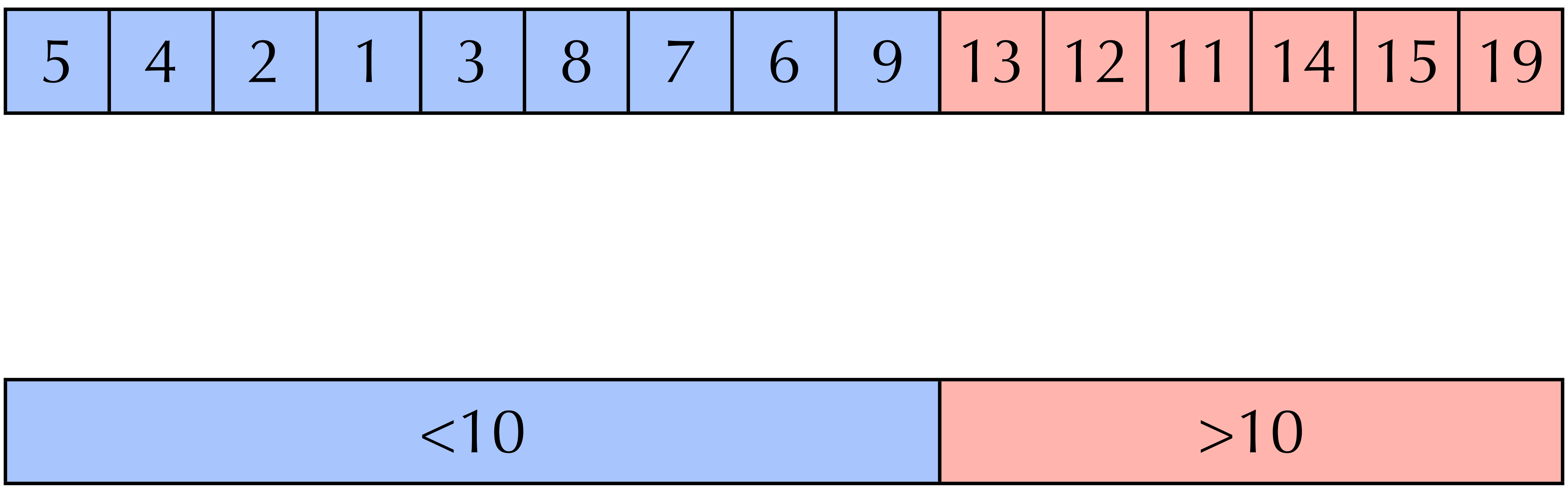
Done!!

Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199

<100	>100

Is Sorted Needed?



Alternative Approach
Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199
L							T							

Step=7

T=L+Step

Alternative Approach
Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199
							L							T

Step=7

T=L+Step

Alternative Approach
Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199
							L			T				

Step=3

T=L+Step

Alternative Approach
Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199
							L	T						

Step=1

T=L+Step

Alternative Approach
Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199
								L	T					

Step=1

T=L+Step

Alternative Approach
Find 100

1	12	23	41	52	67	71	84	91	101	102	103	105	121	199
								L						

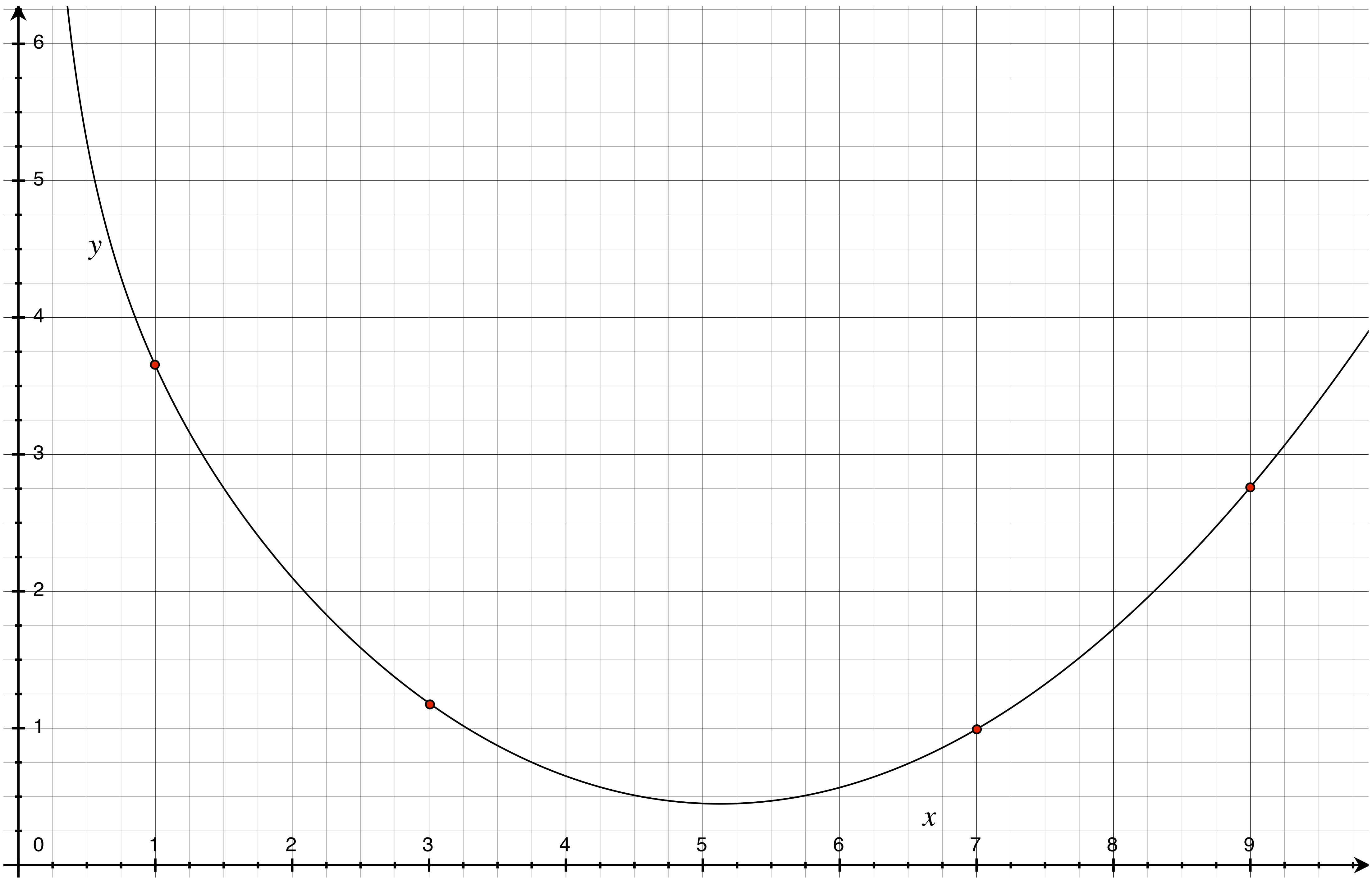
Step=0

Done!

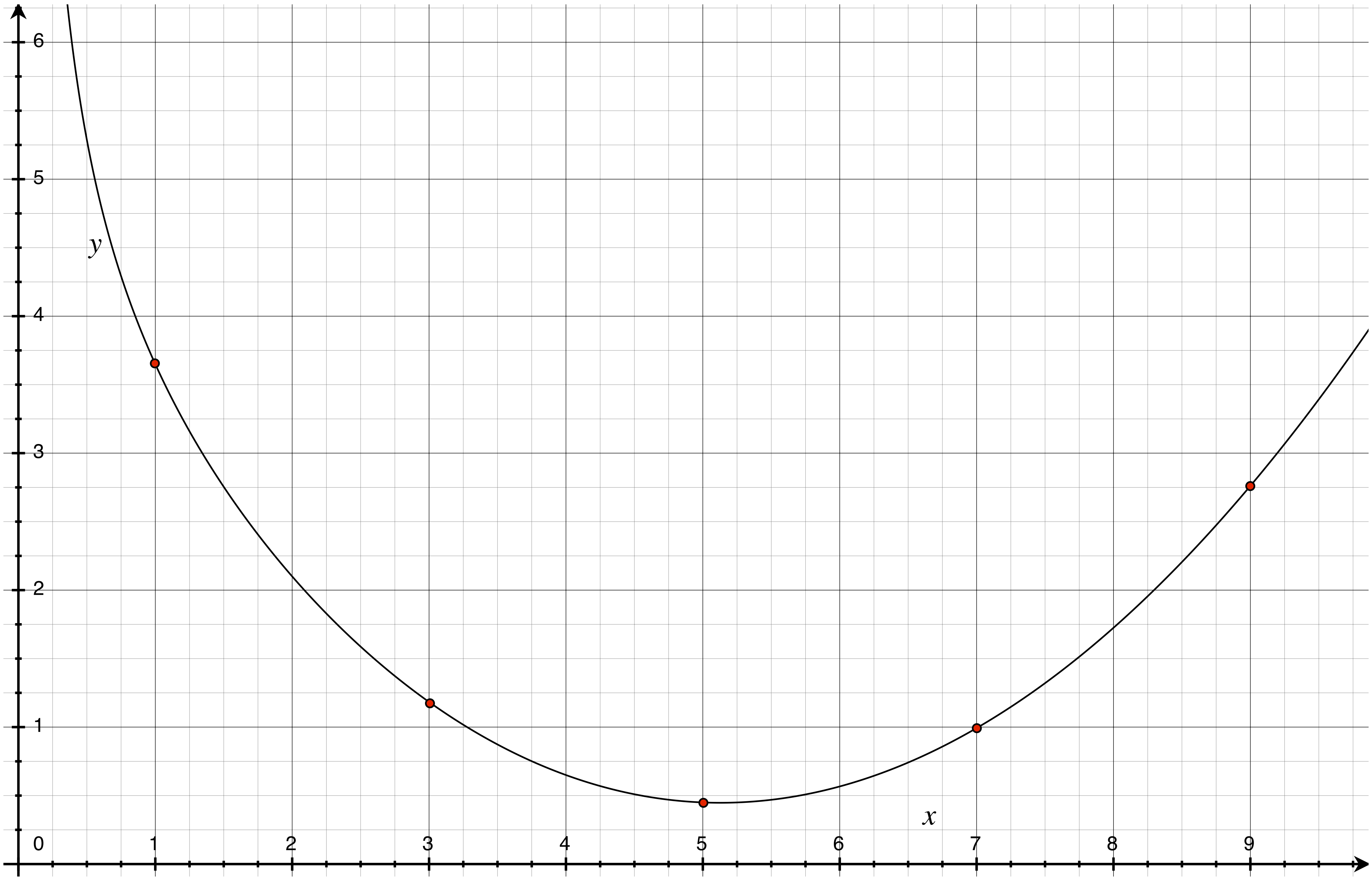
Golden section search

- Given a convex array A[1..n], find out the minimum element in A.
- Convex array: $cA[i]+(1-c)A[j] \geq A[ci+(1-c)j]$ for $ci+(1-c)j$ is an integer between i and j.
- How many elements have to be queried?
  - O(logn)
  - How?

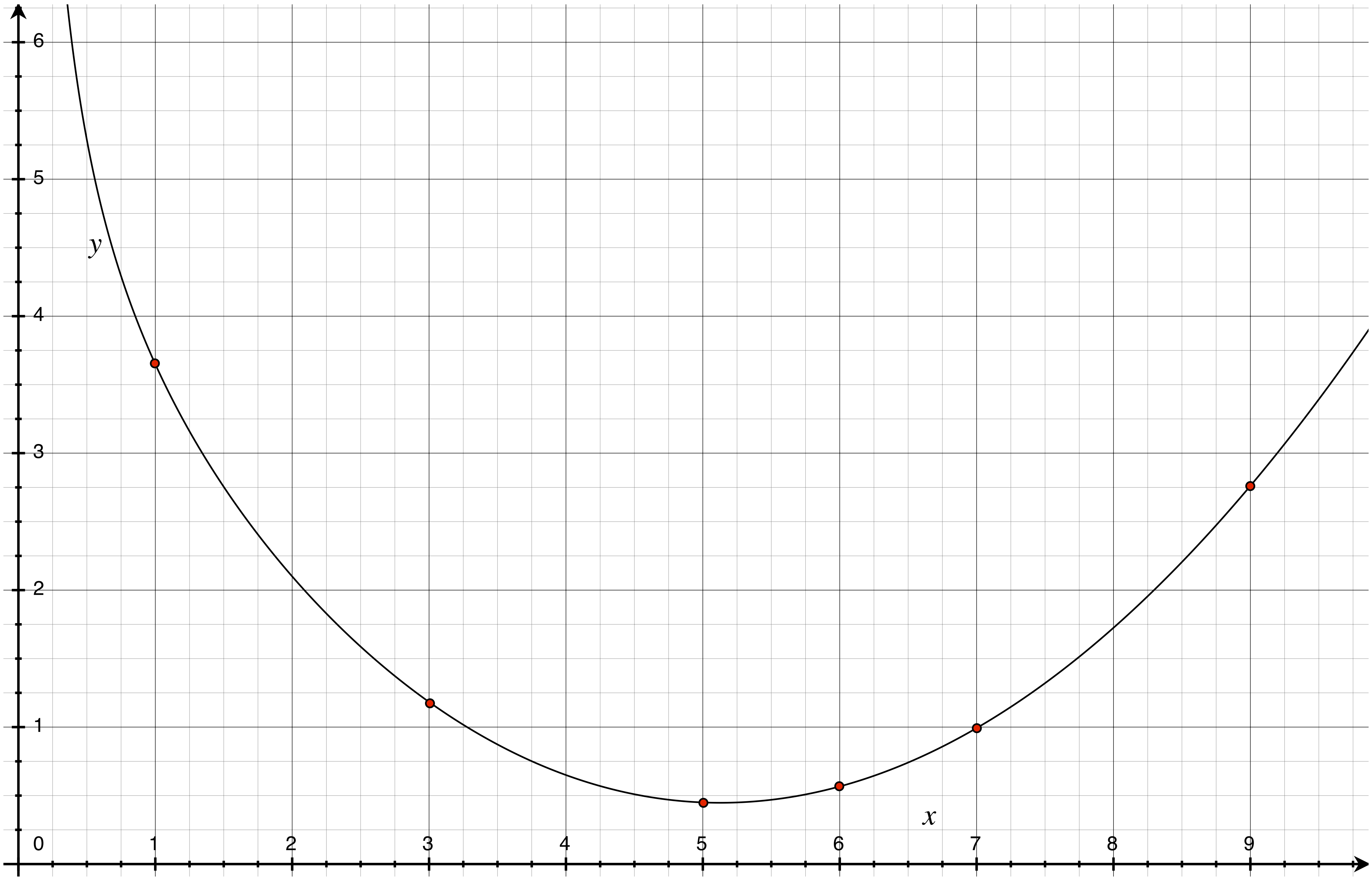
Example 1



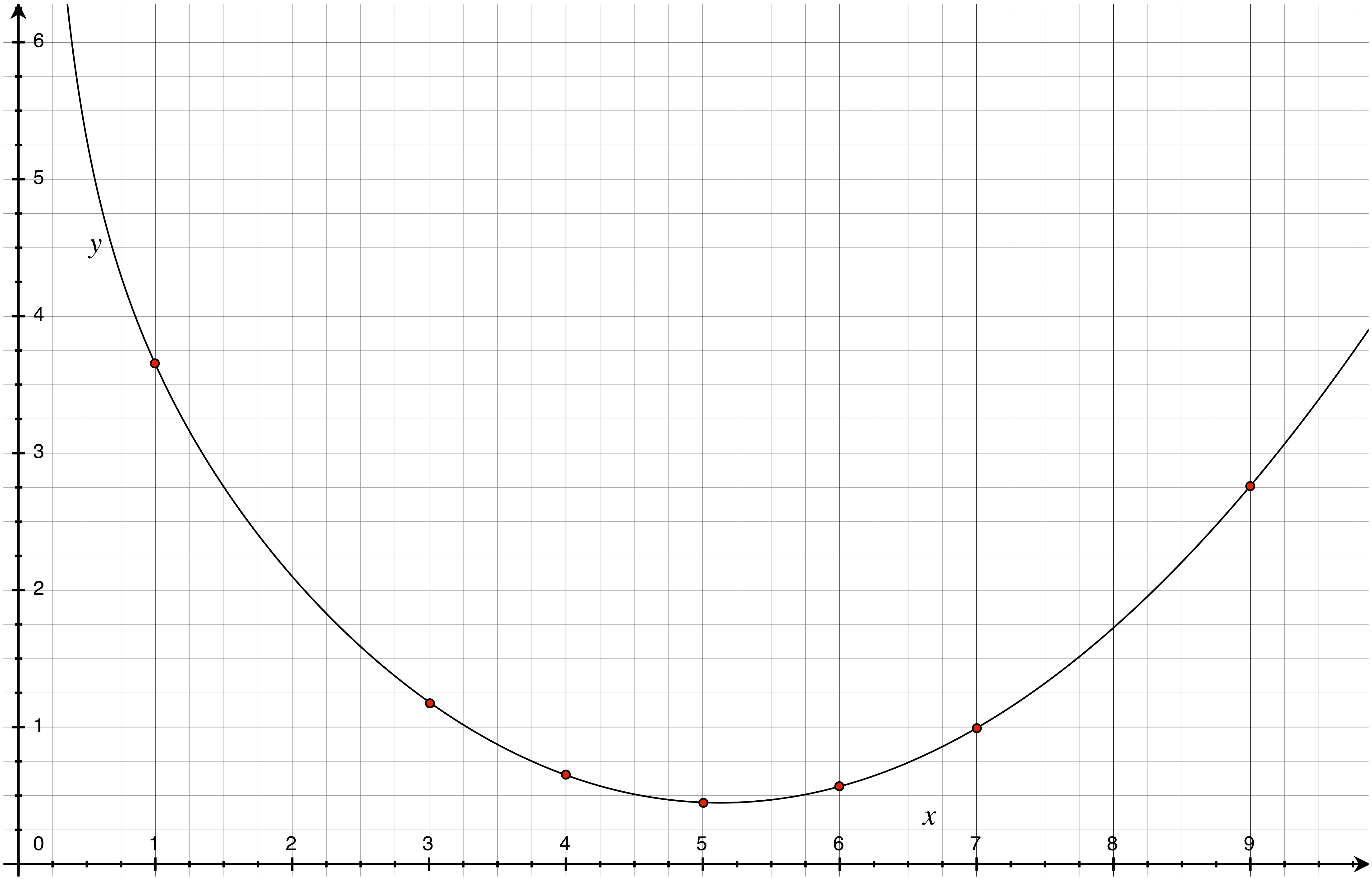
Example 1



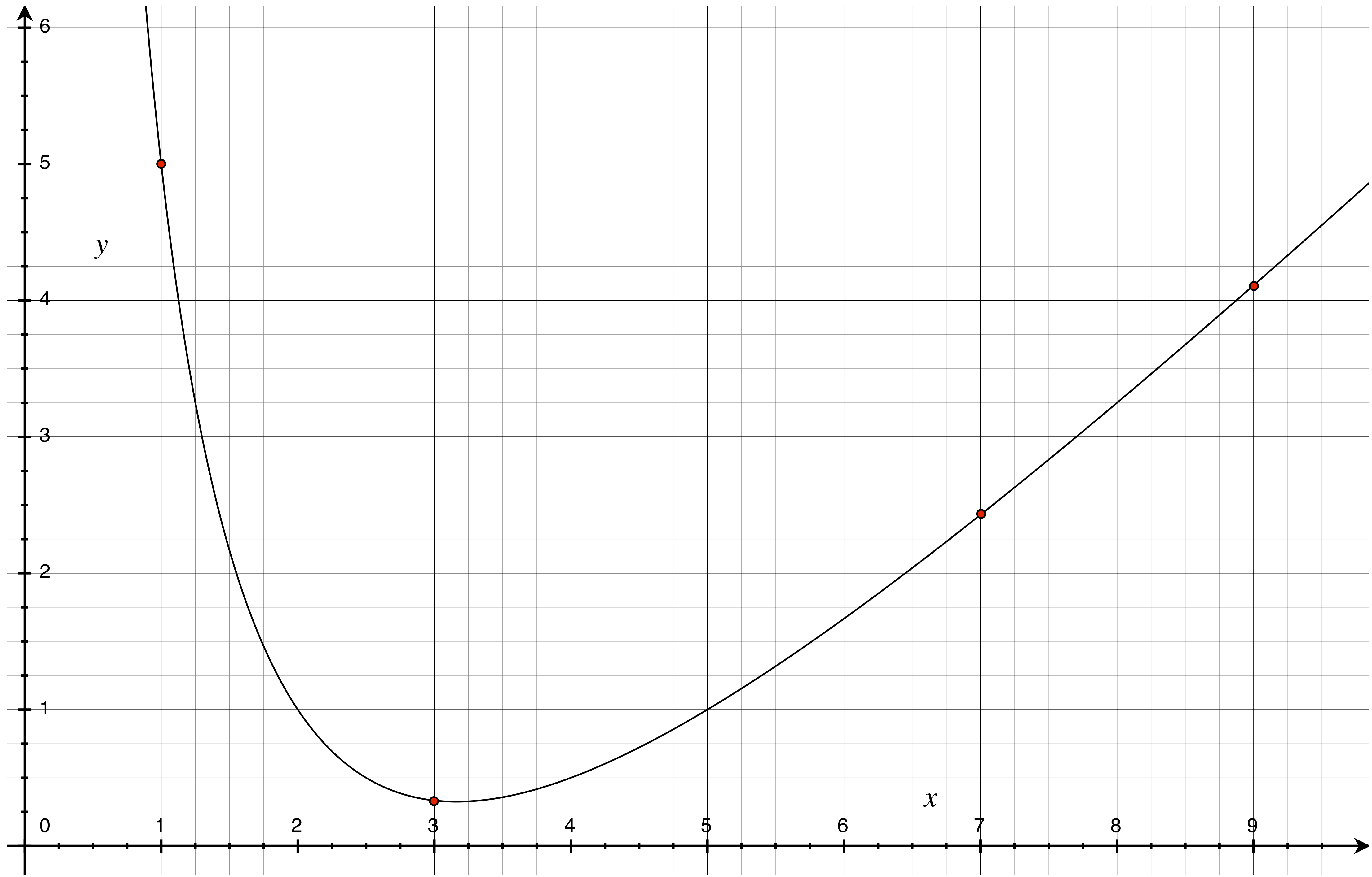
Example 1



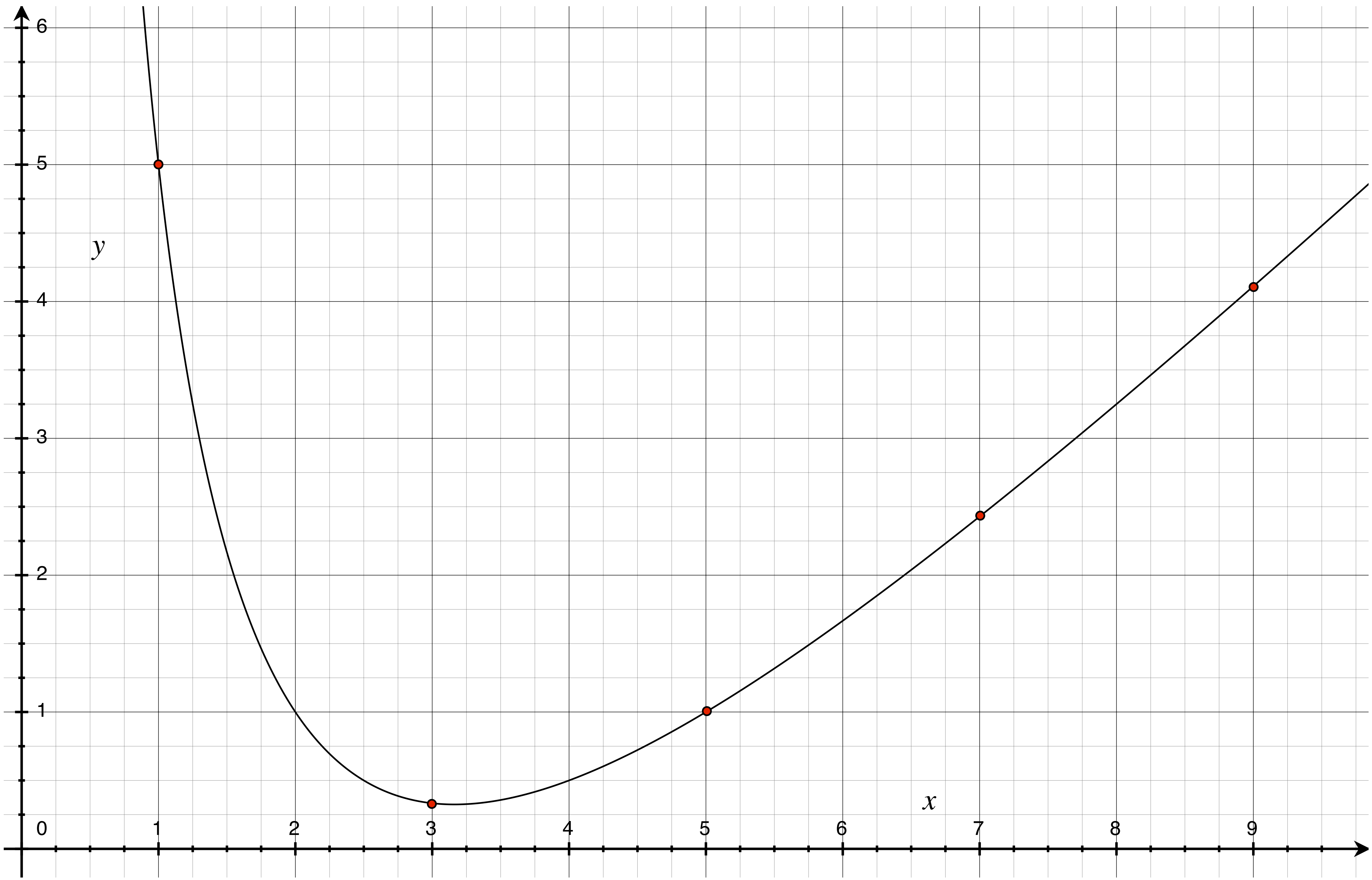
Example 1



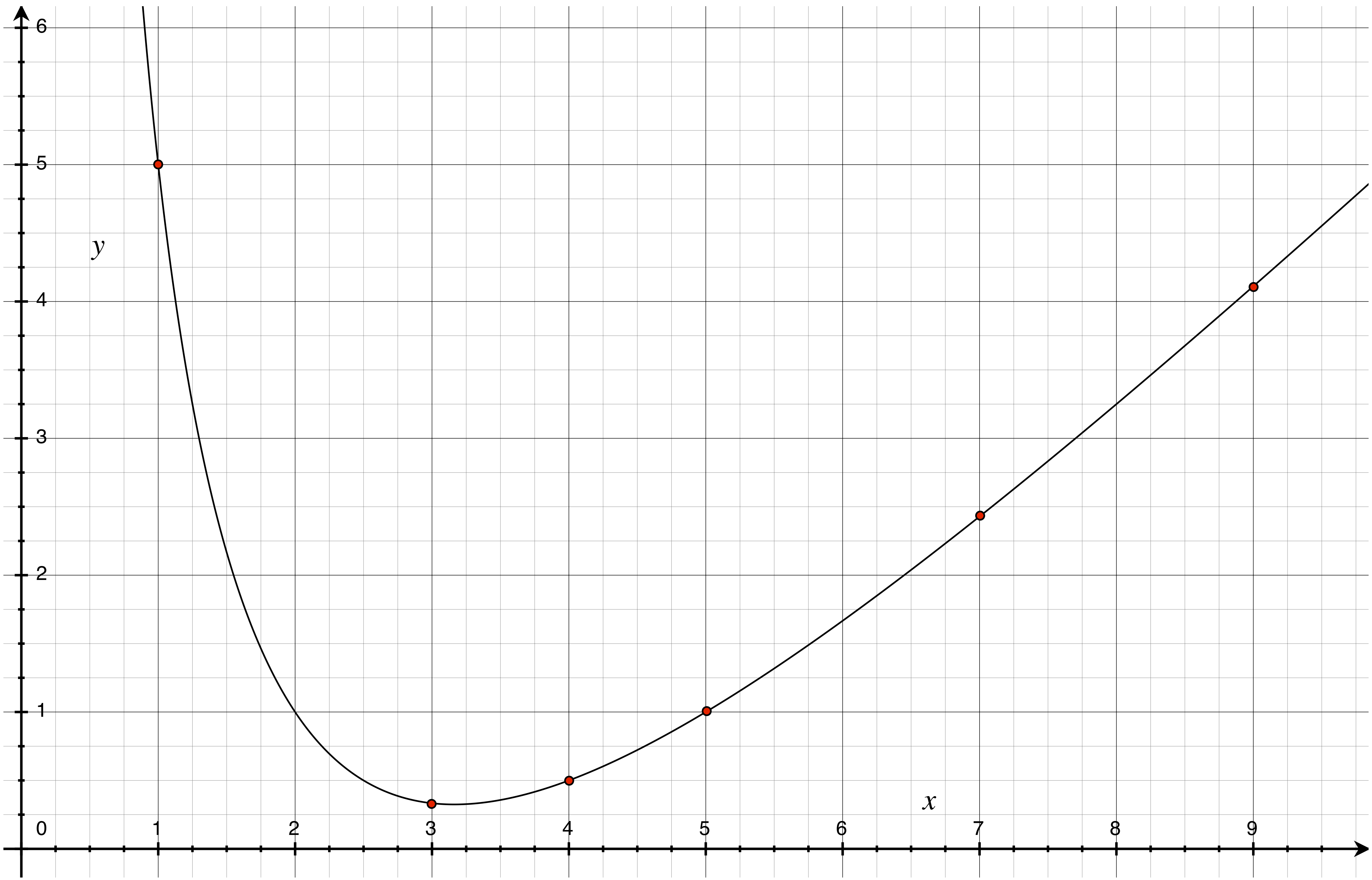
Example 2



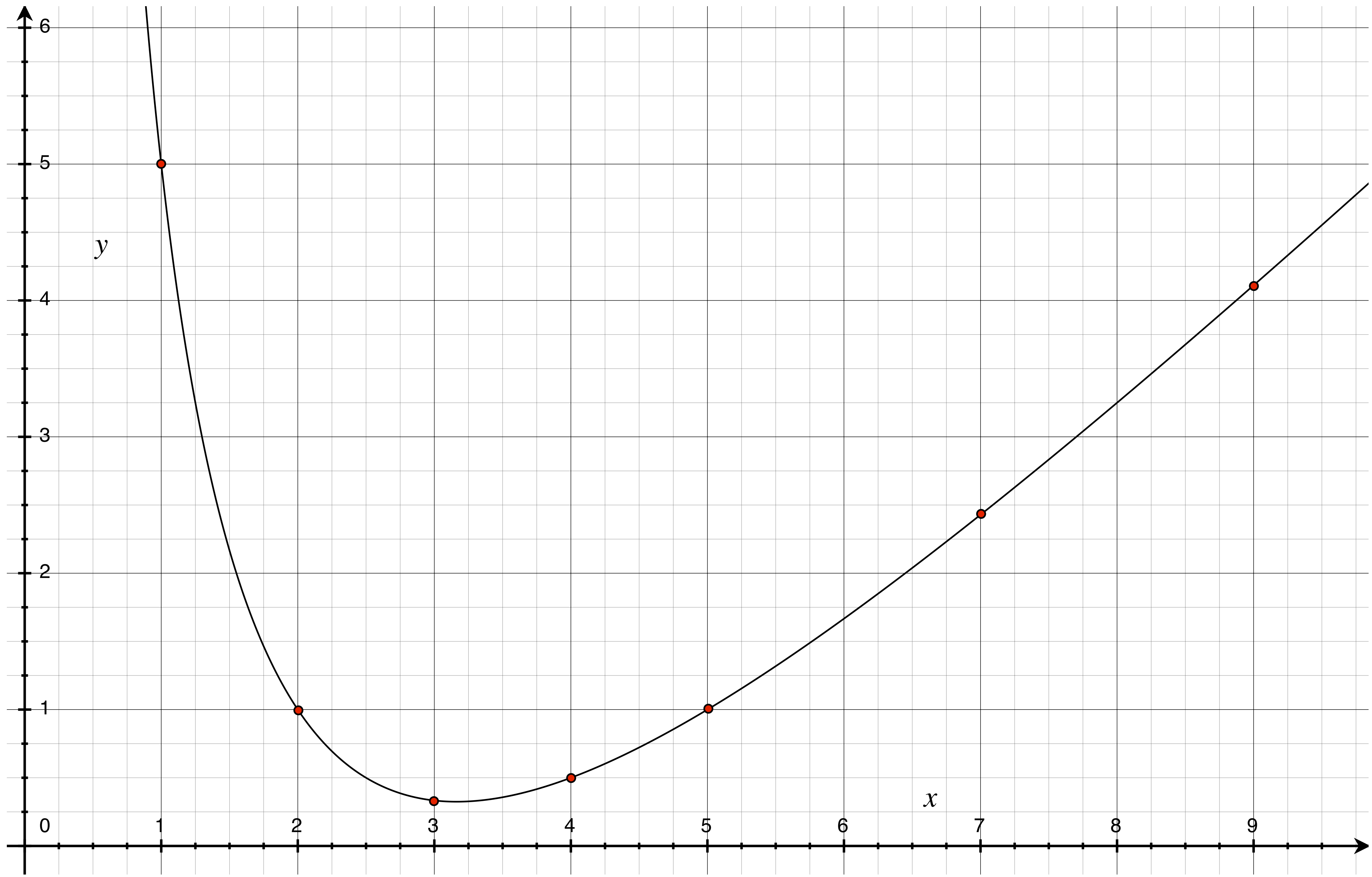
Example 2



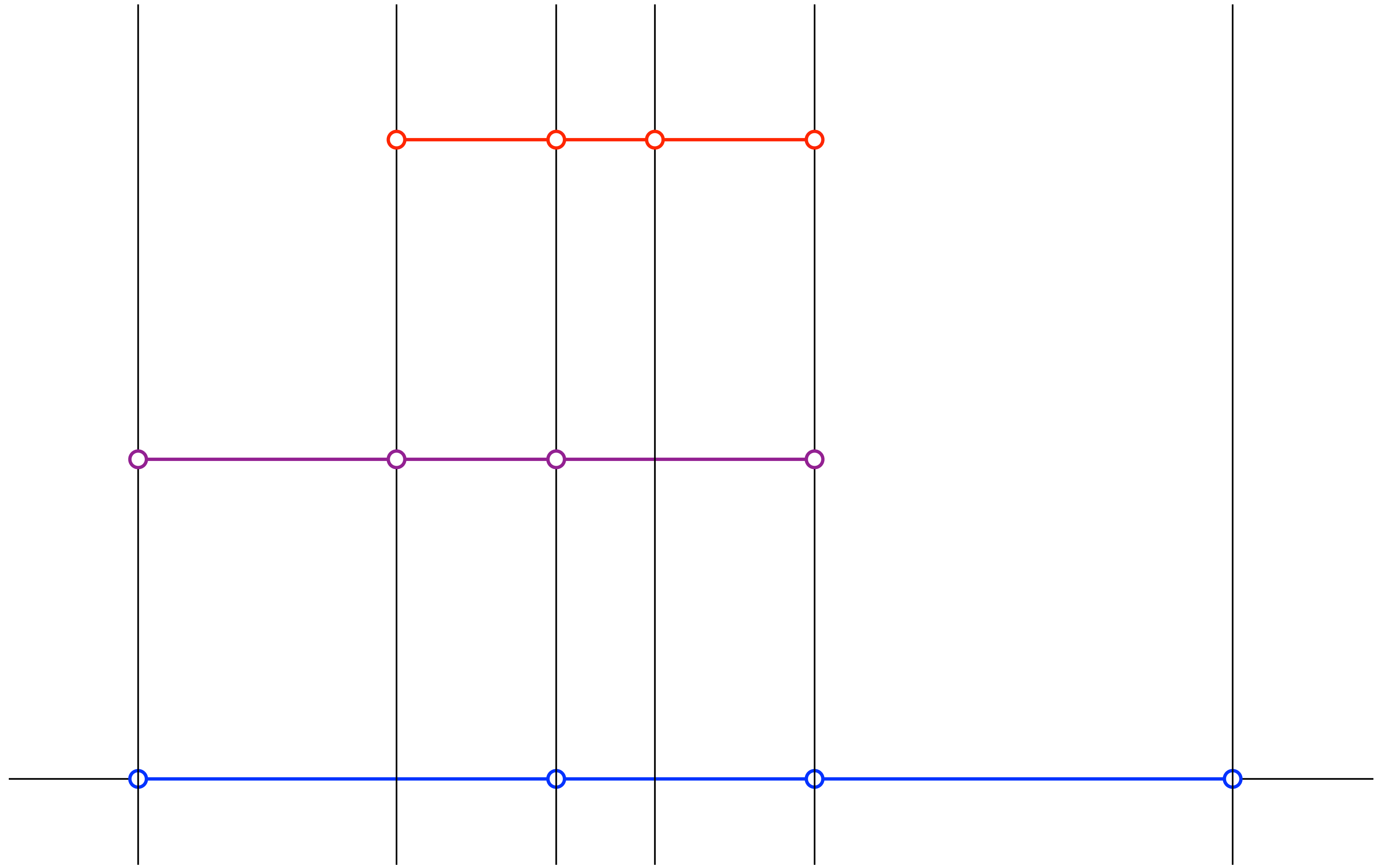
Example 2



Example 2



Golden section



Recycle the samples by golden ratio sampling