Chương 1
THUẬT TOÁN QUICK SORT

ThS. Nguyễn Tấn
Trần Minh Khang

1. TƯ TƯỞNG THUẬT TOÁN QUICK SORT

- Thuật toán quick sort chia không gian cần sắp xếp thành 2 không gian con là không gian con 1 và không gian con 2. Không gian con 1 là không gian mà tất cả các phần tử thuộc không gian này đều nhỏ hơn tất cả các phần tử thuộc không gian con 2.
  - Nếu không gian con thứ nhất có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Quick Sort.
  - Nếu không gian con thứ hai có nhiều hơn một phần tử thì sắp xếp không gian con này bằng thuật toán Quick Sort.

2. CÀI ĐẶT CẢI TIẾN

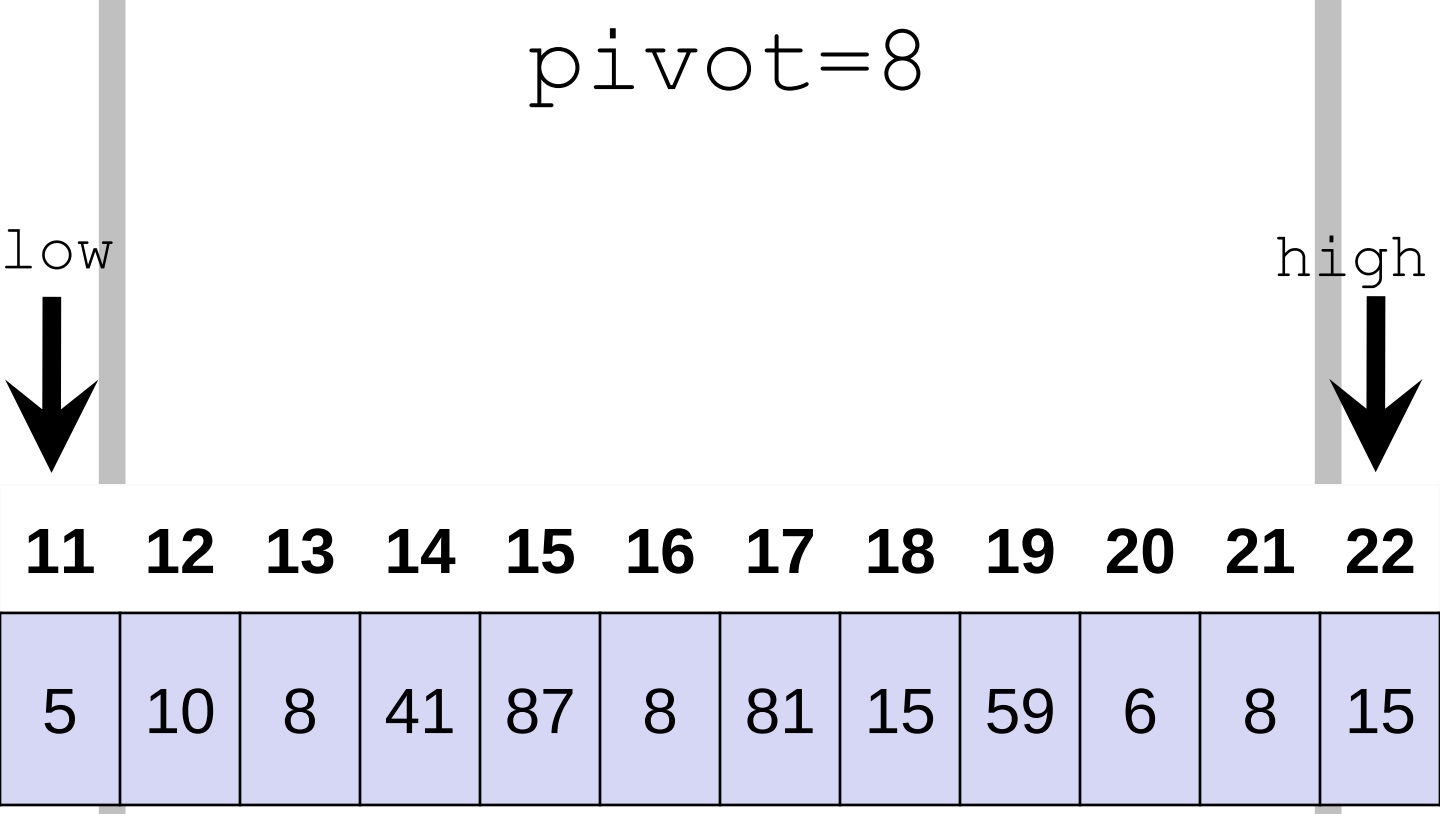
```
10. void SapTang(int a[], int n)
11. {
12.     QuickSort(a,0,n-1);
13. }
14. void HoanVi(int &a,int &b)
15. {
16.     int temp = a;
17.     a = b;
18.     b = temp;
19. }
```

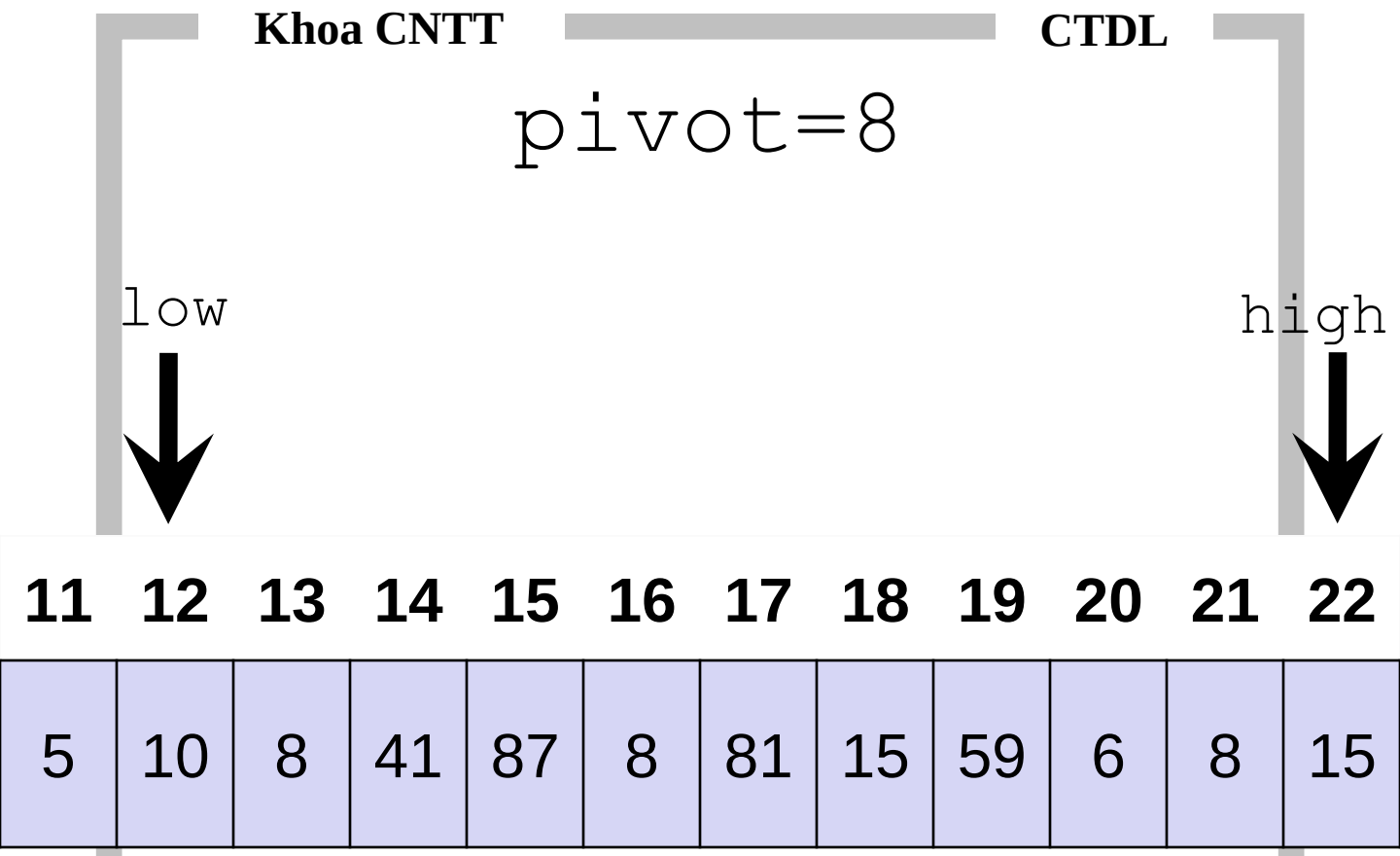
2. CÀI ĐẶT CẢI TIẾN

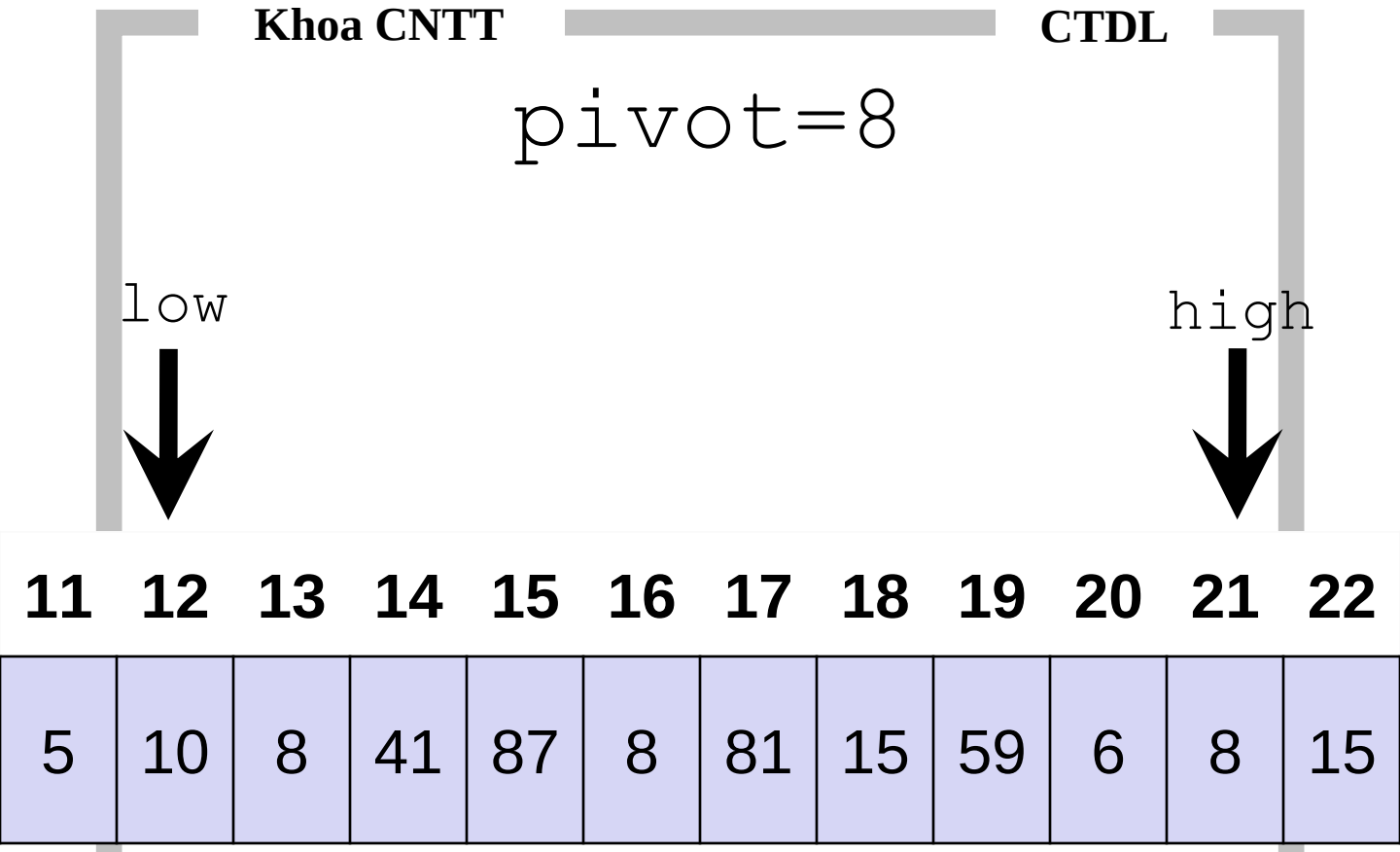
```
10. void QuickSort(int a[],
                   int Left,int Right)
11. {
12.   if(Left<Right)
13.   {
14.     int LastLeft,FirstRight;
15.     Partition(a,Left,Right,
            LastLeft,FirstRight);
16.     QuickSort(a,Left,LastLeft);
17.     QuickSort(a,FirstRight,Right);
18.   }
19. }
```

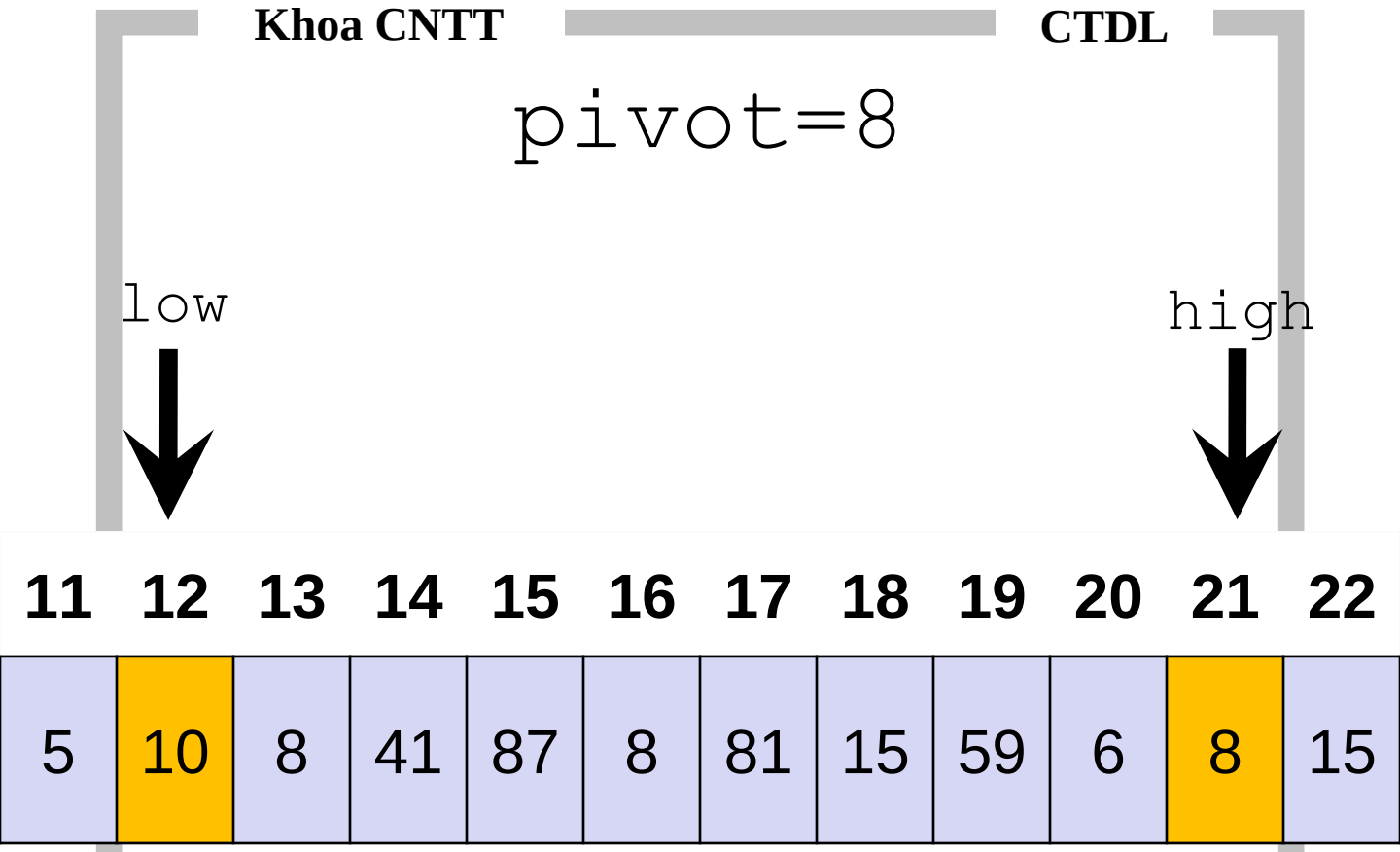
```
10. void Partition(int a[],
        int Left, int Right,
        int&LastLeft,int&FirstRight)
11. {
12.     int pivot=a[(Left+Right)/2];
13.     int low = Left;
14.     int high = Right;
15.     while(low<high)
16.     {
17.         while (a[low]<pivot)
18.             low++;
19.         while (a[high]>pivot)
20.             high--;
21.         if(low<=high)
22.         {
23.             HoanVi(a[low],a[high]);
24.             low++;
25.             high--;
26.         }
27.     }
28.     LastLeft = high;
29.     FirstRight = low;
30. }
```

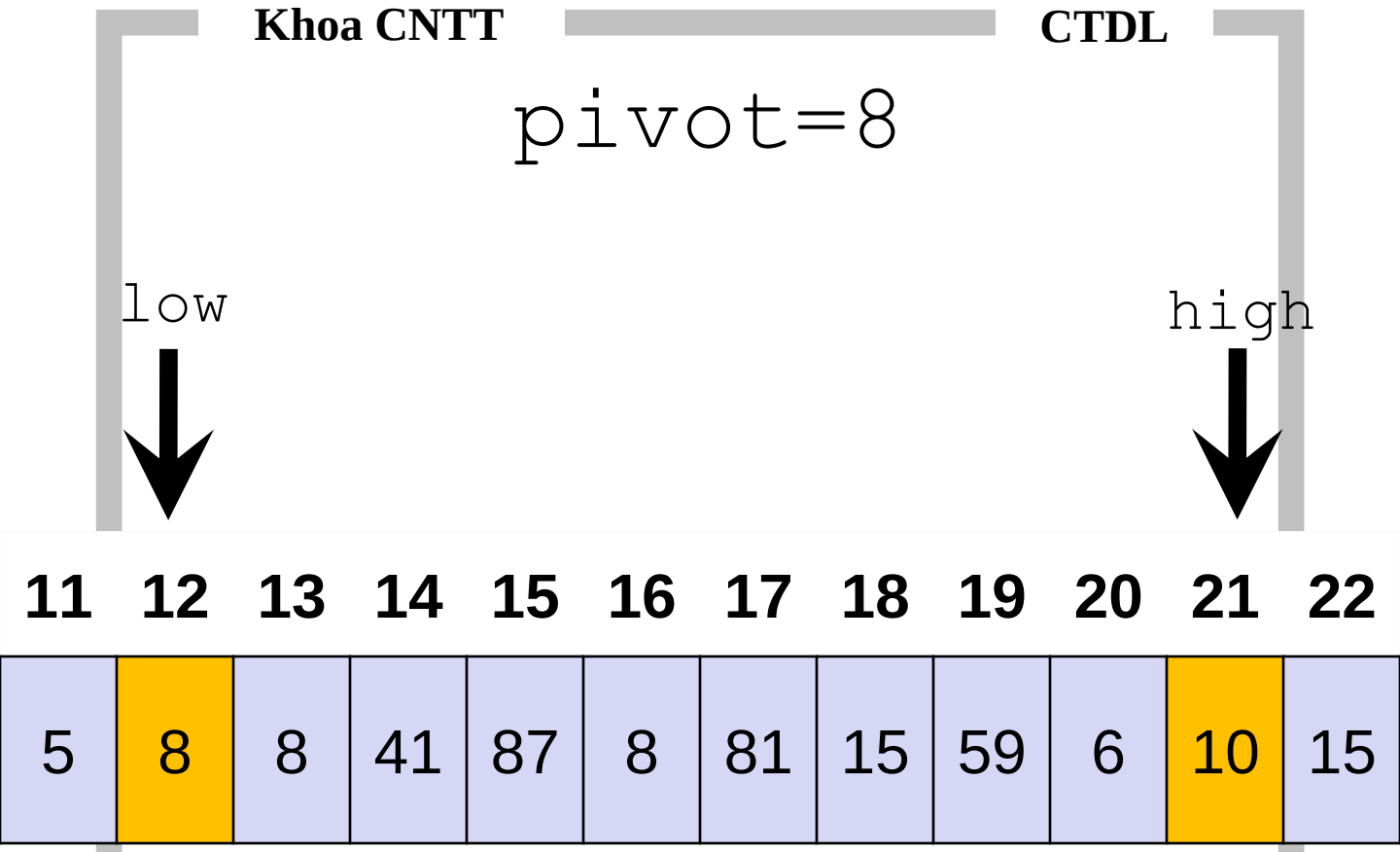
11	12	13	14	15	16	17	18	19	20	21	22
5	10	8	41	87	8	81	15	59	6	8	15

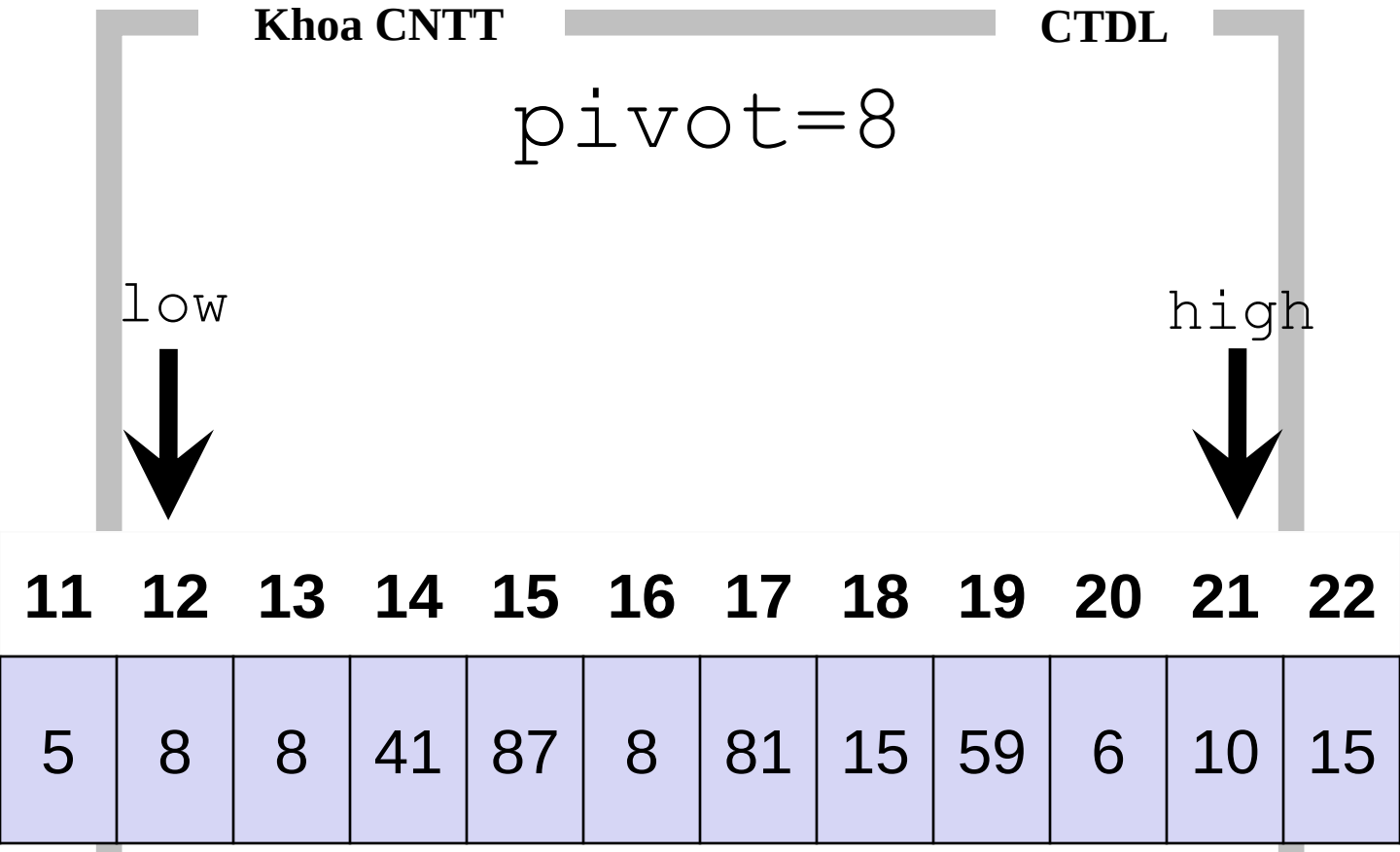

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 10 8 41 87 8 81 15 59 6 8 15

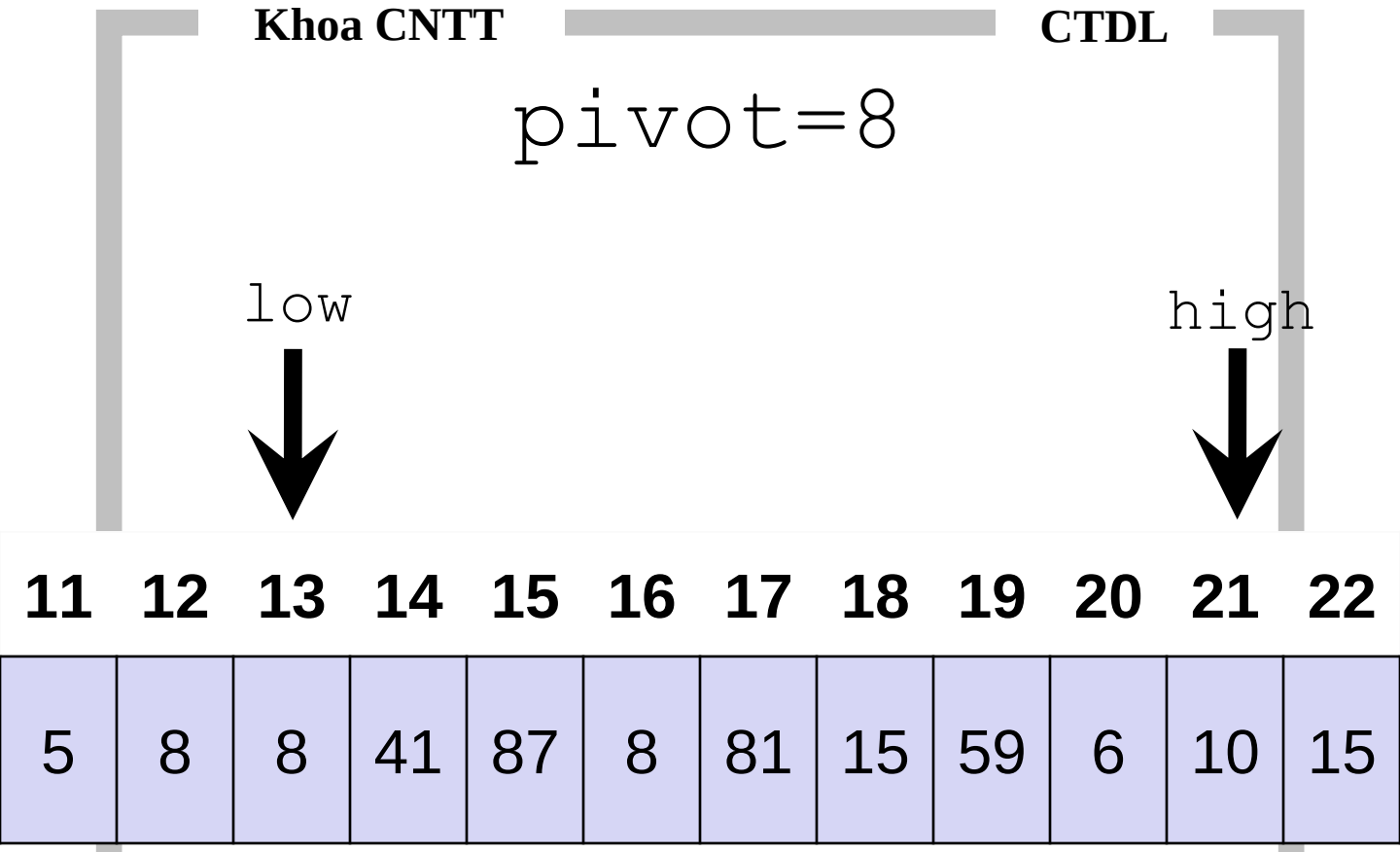

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 10 8 41 87 8 81 15 59 6 8 15

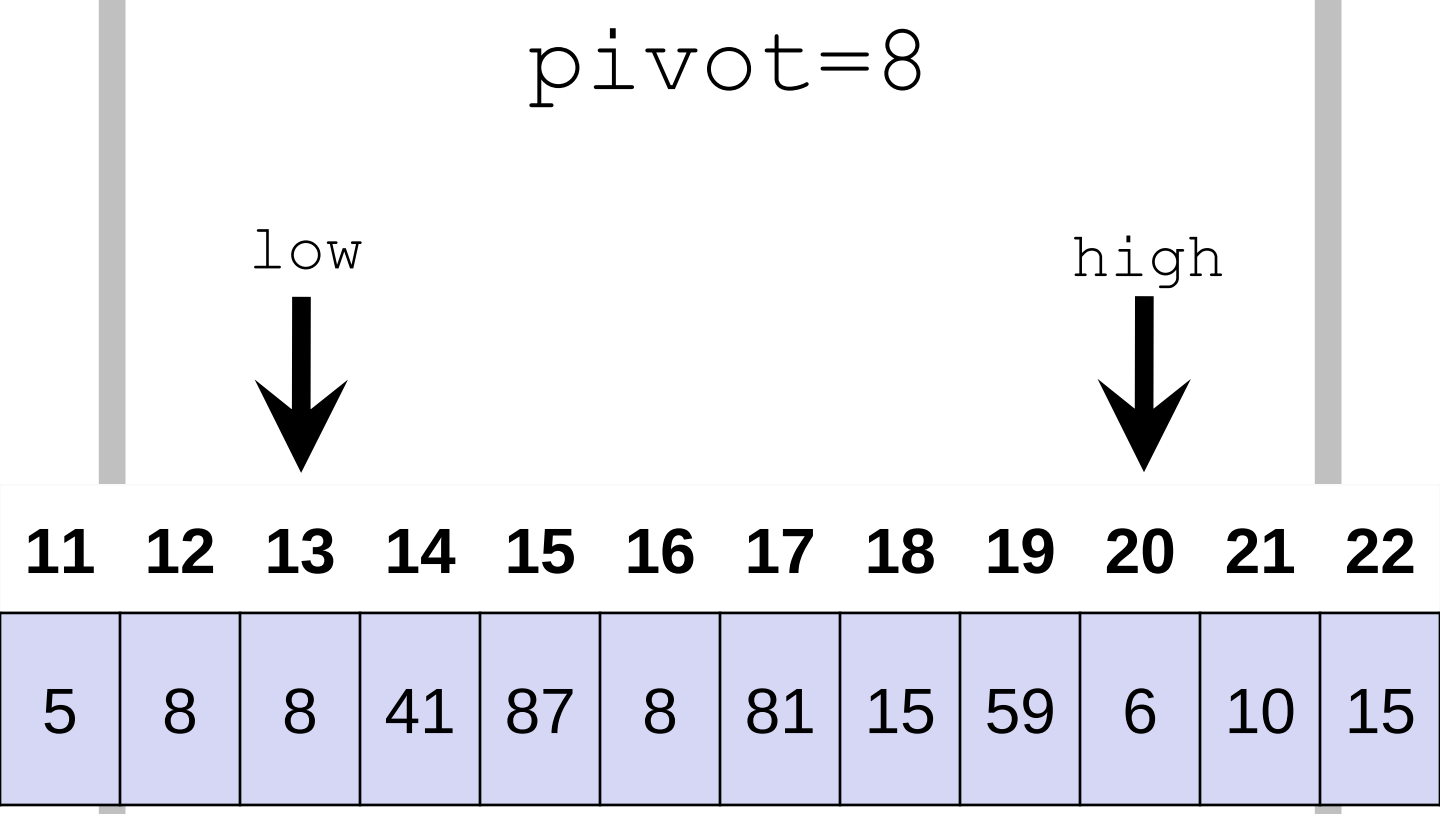

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 10 8 41 87 8 81 15 59 6 8 15

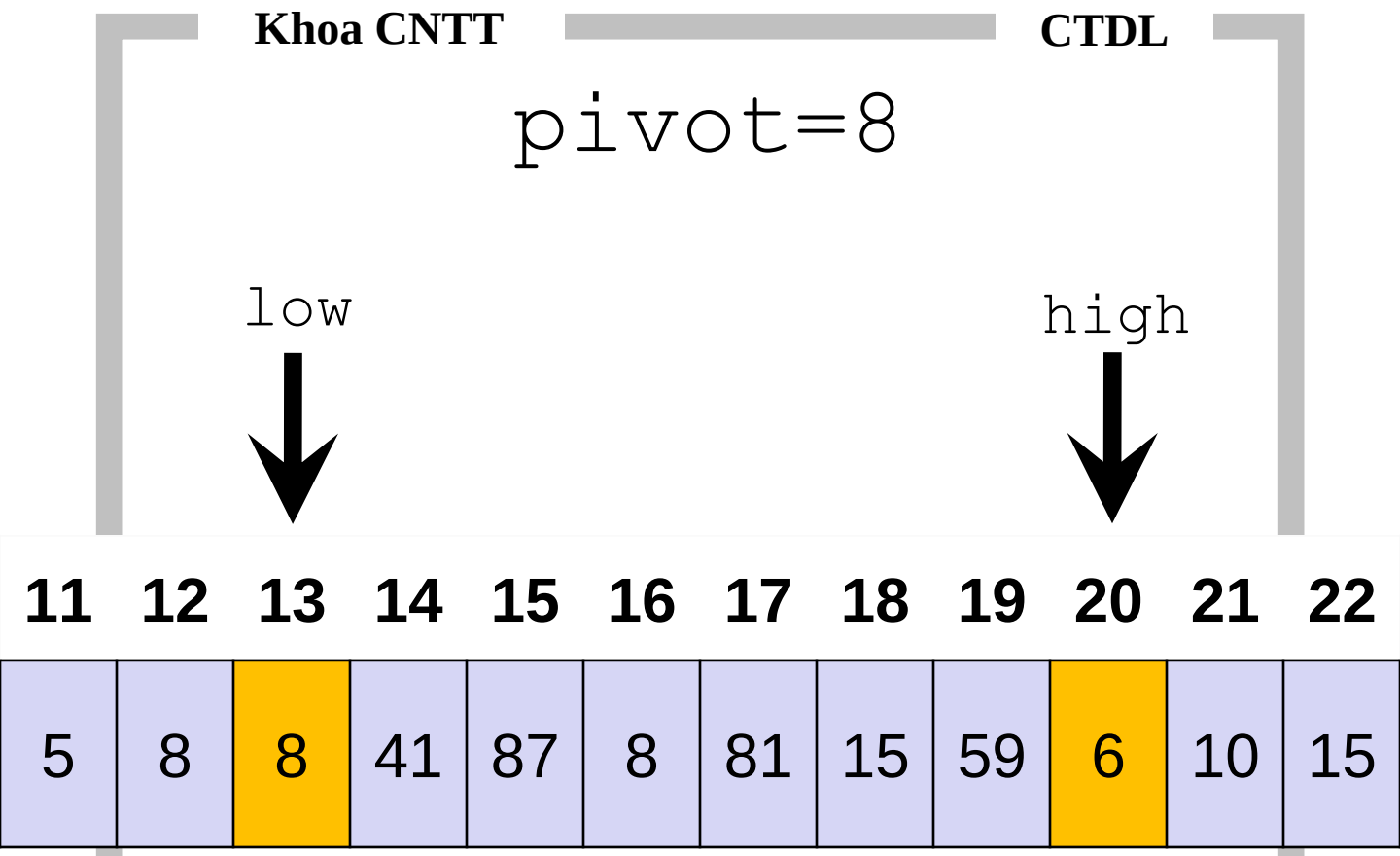

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 10 8 41 87 8 81 15 59 6 8 15

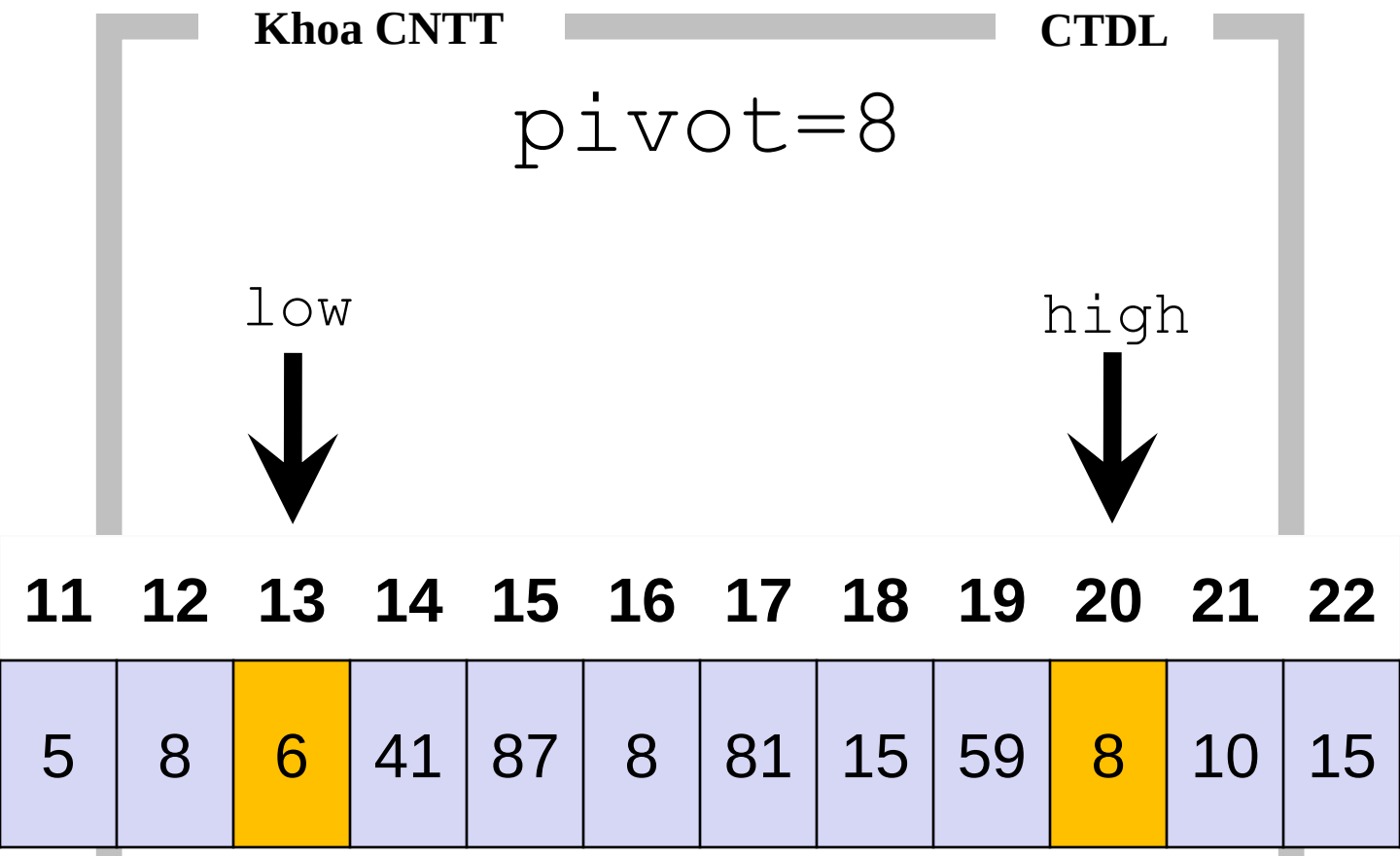

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 8 41 87 8 81 15 59 6 10 15

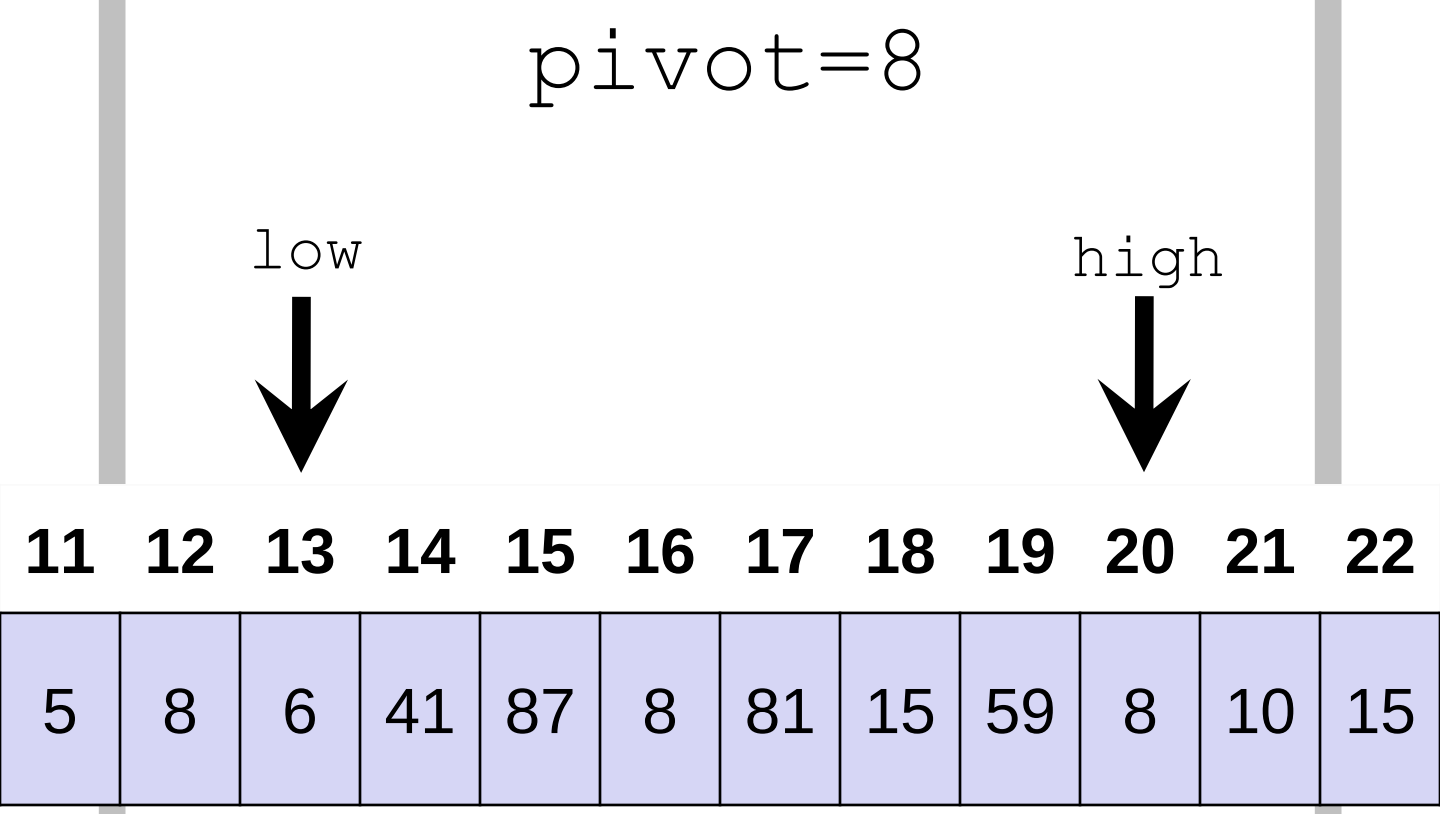

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 8 41 87 8 81 15 59 6 10 15

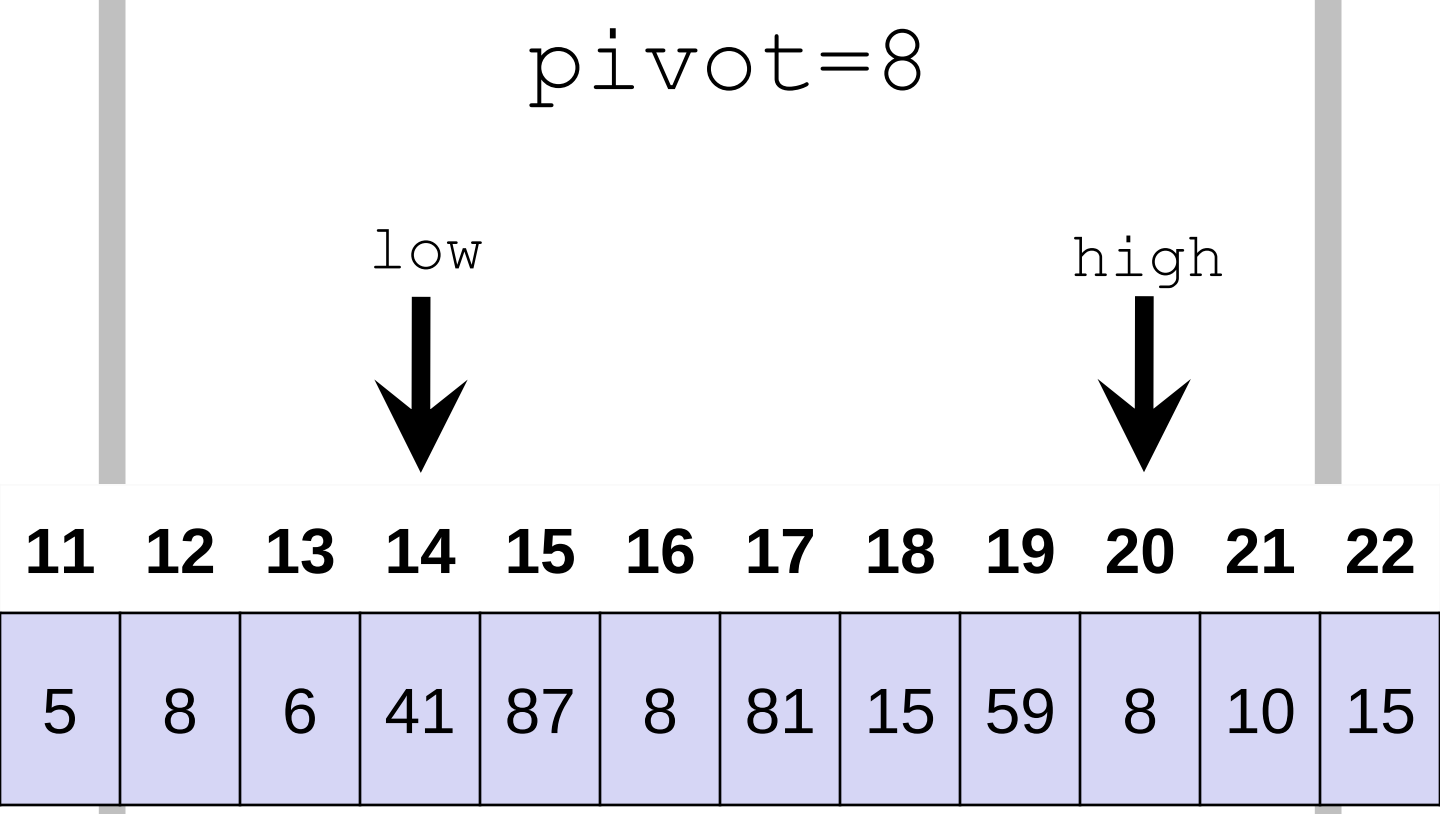

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 8 41 87 8 81 15 59 6 10 15

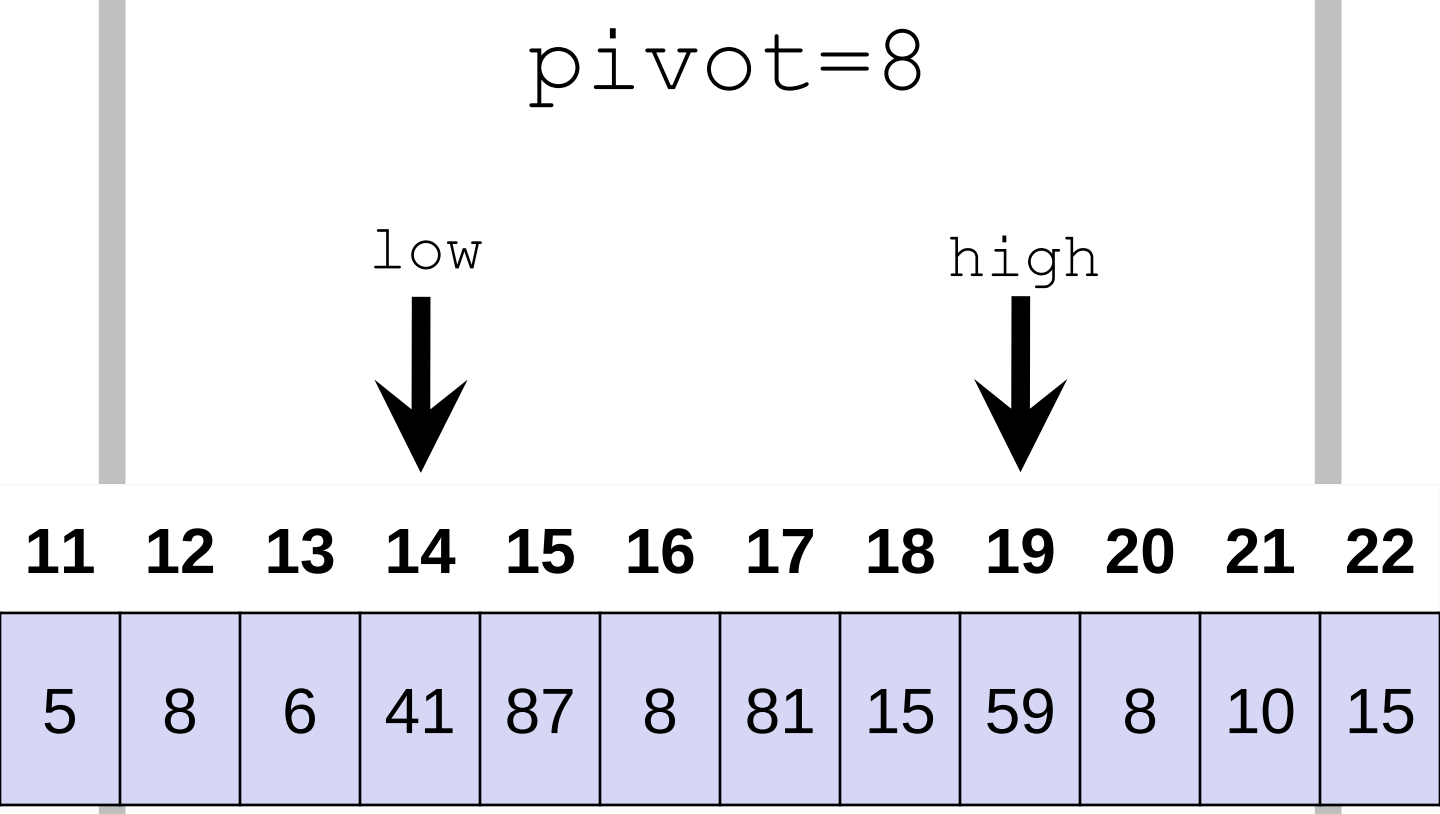

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 8 41 87 8 81 15 59 6 10 15

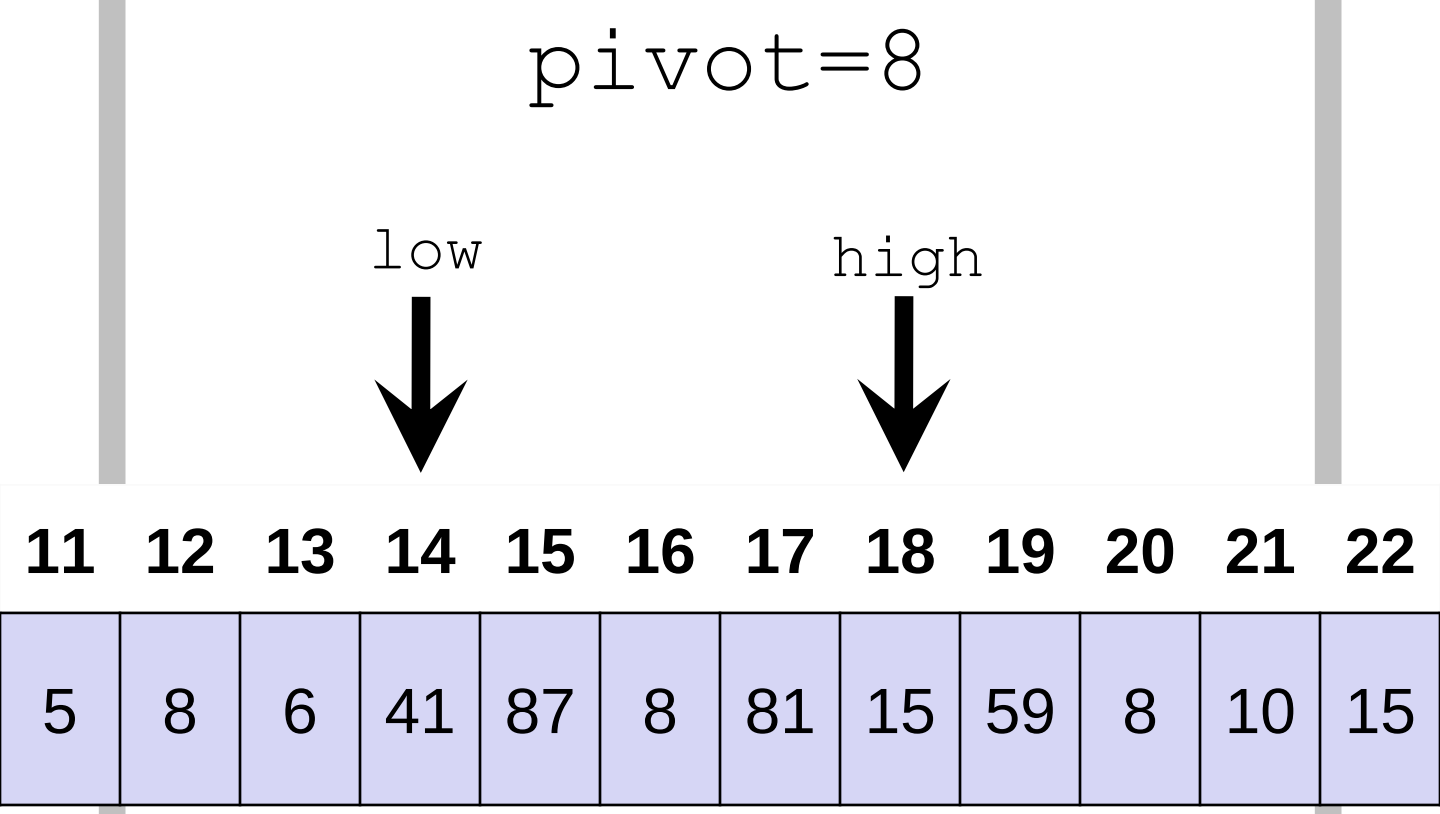

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 8 41 87 8 81 15 59 6 10 15

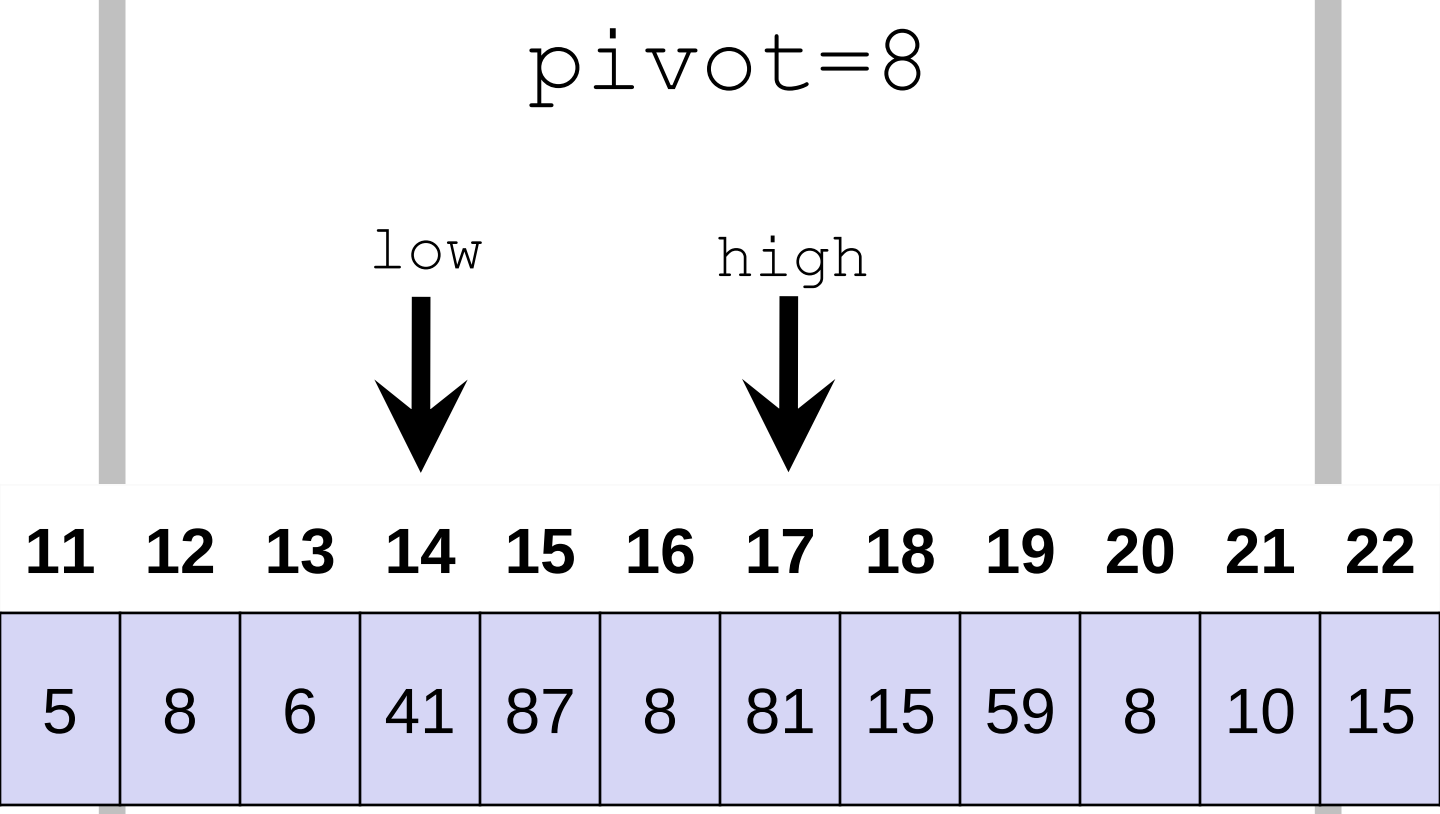

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15

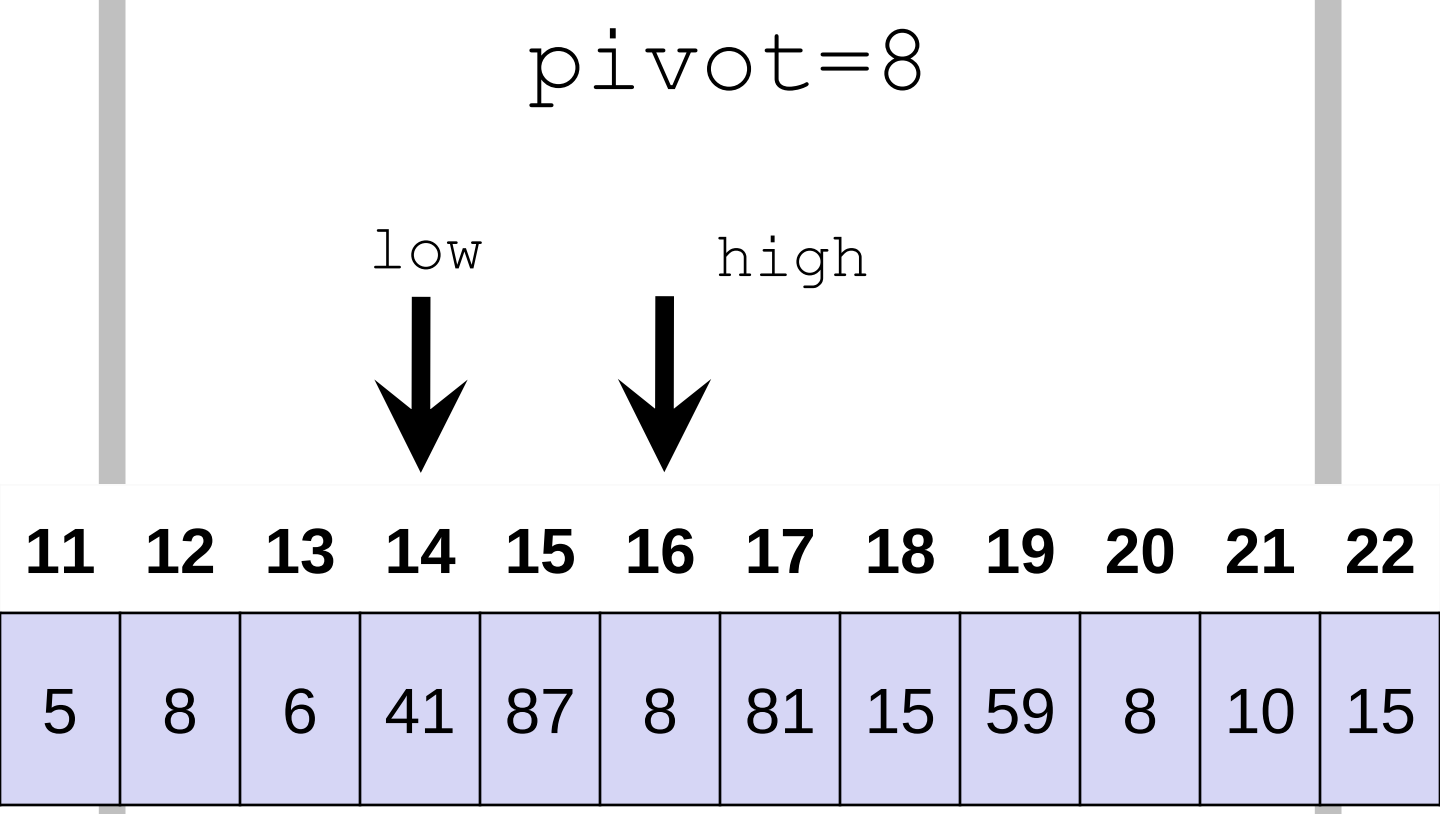

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15

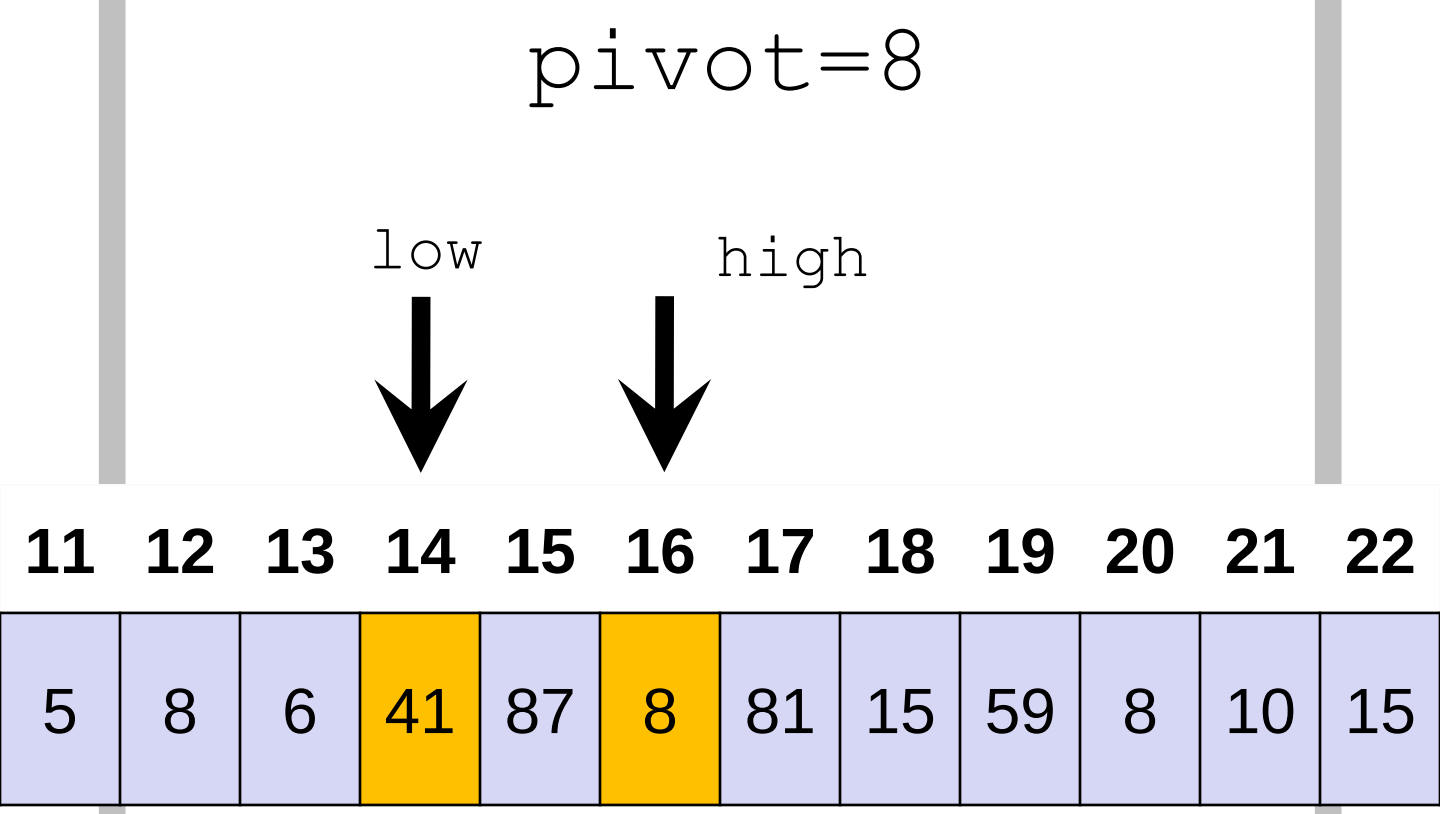

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15

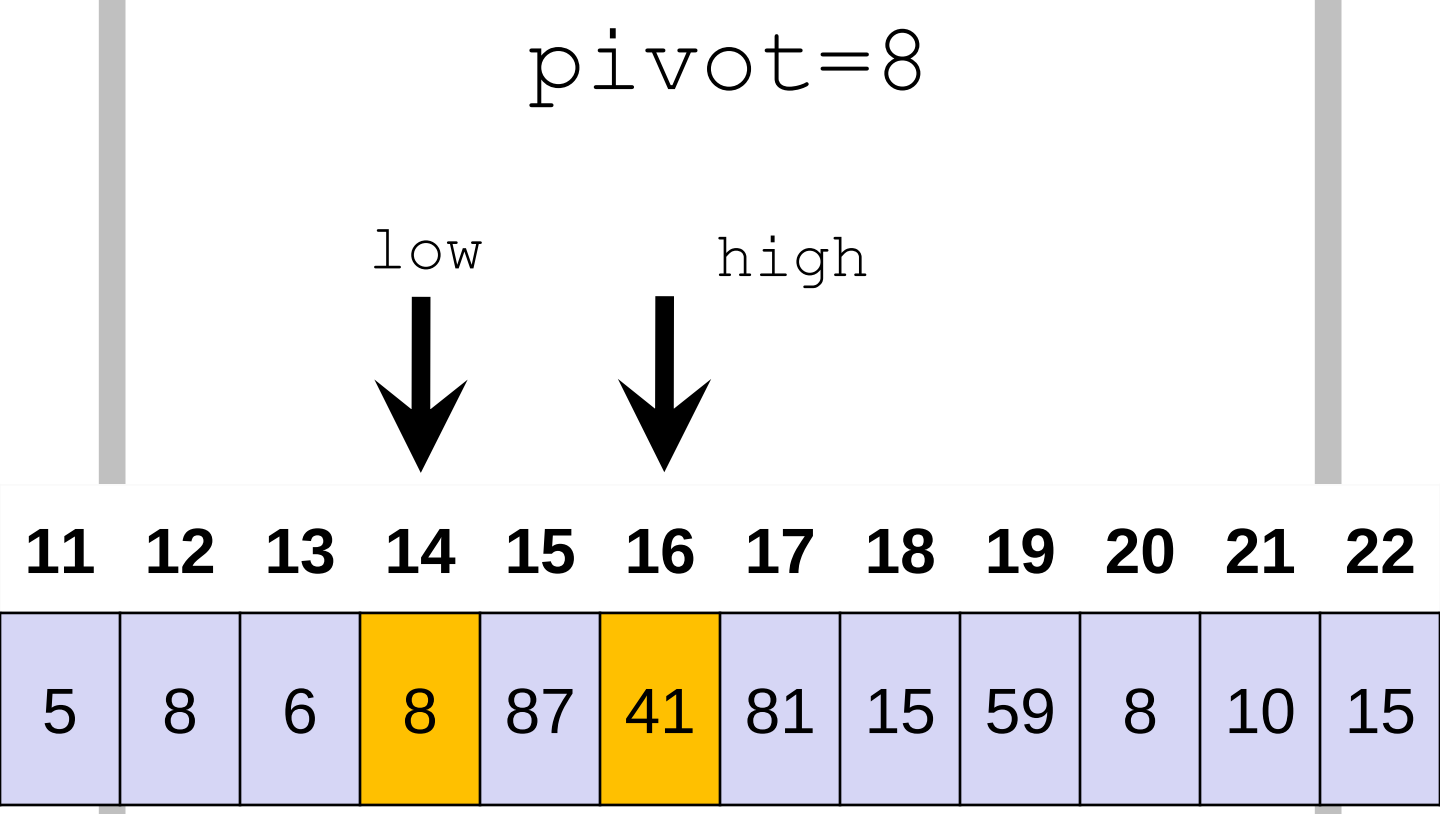

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15

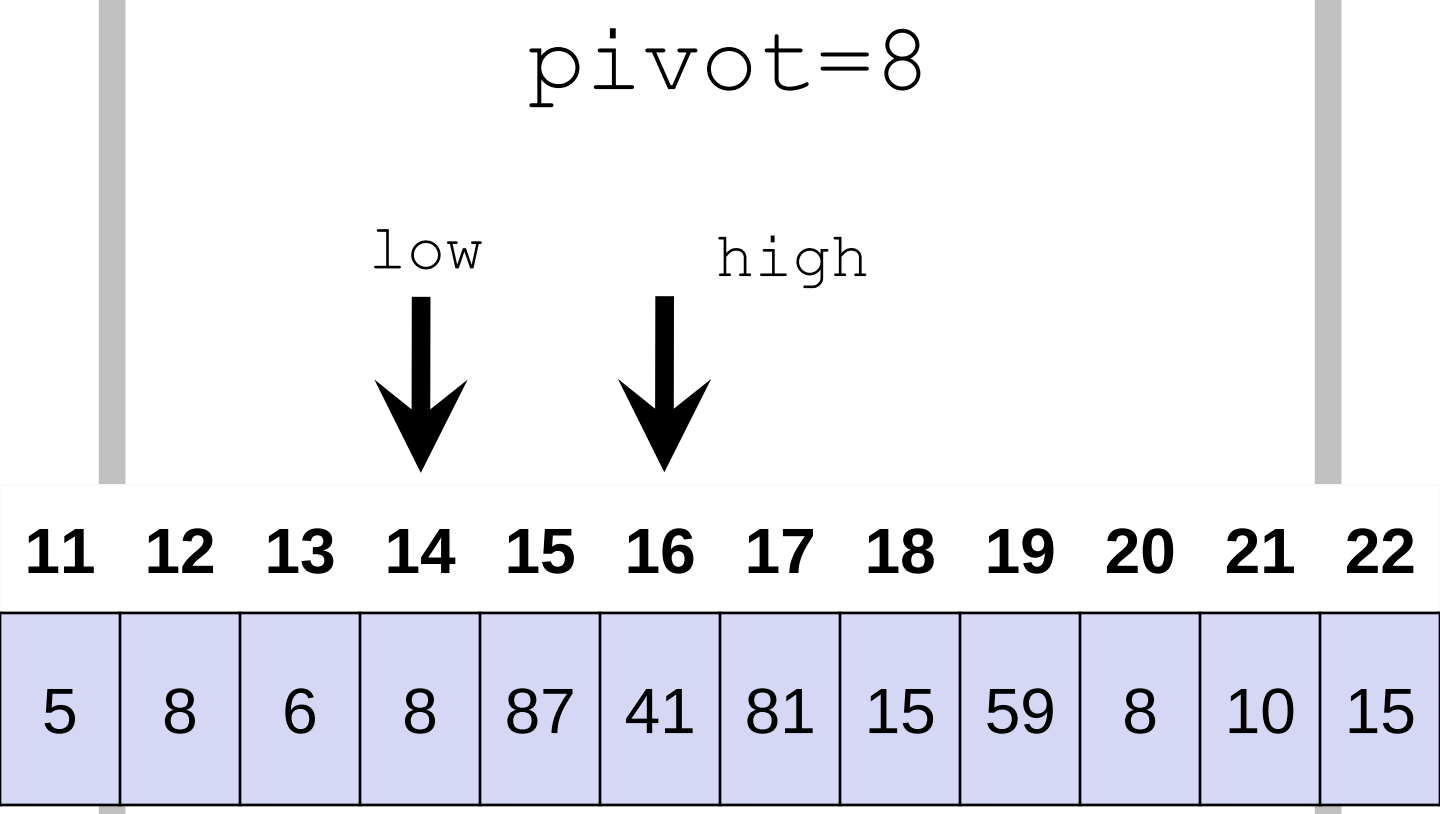

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15


pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15

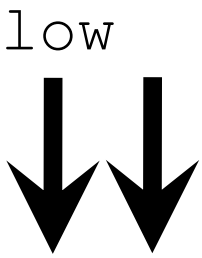

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15


pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 41 87 8 81 15 59 8 10 15

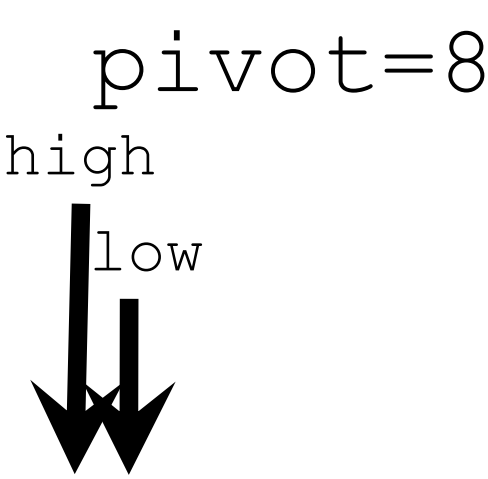

pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 8 87 41 81 15 59 8 10 15


pivot=8
low
high
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 8 87 41 81 15 59 8 10 15

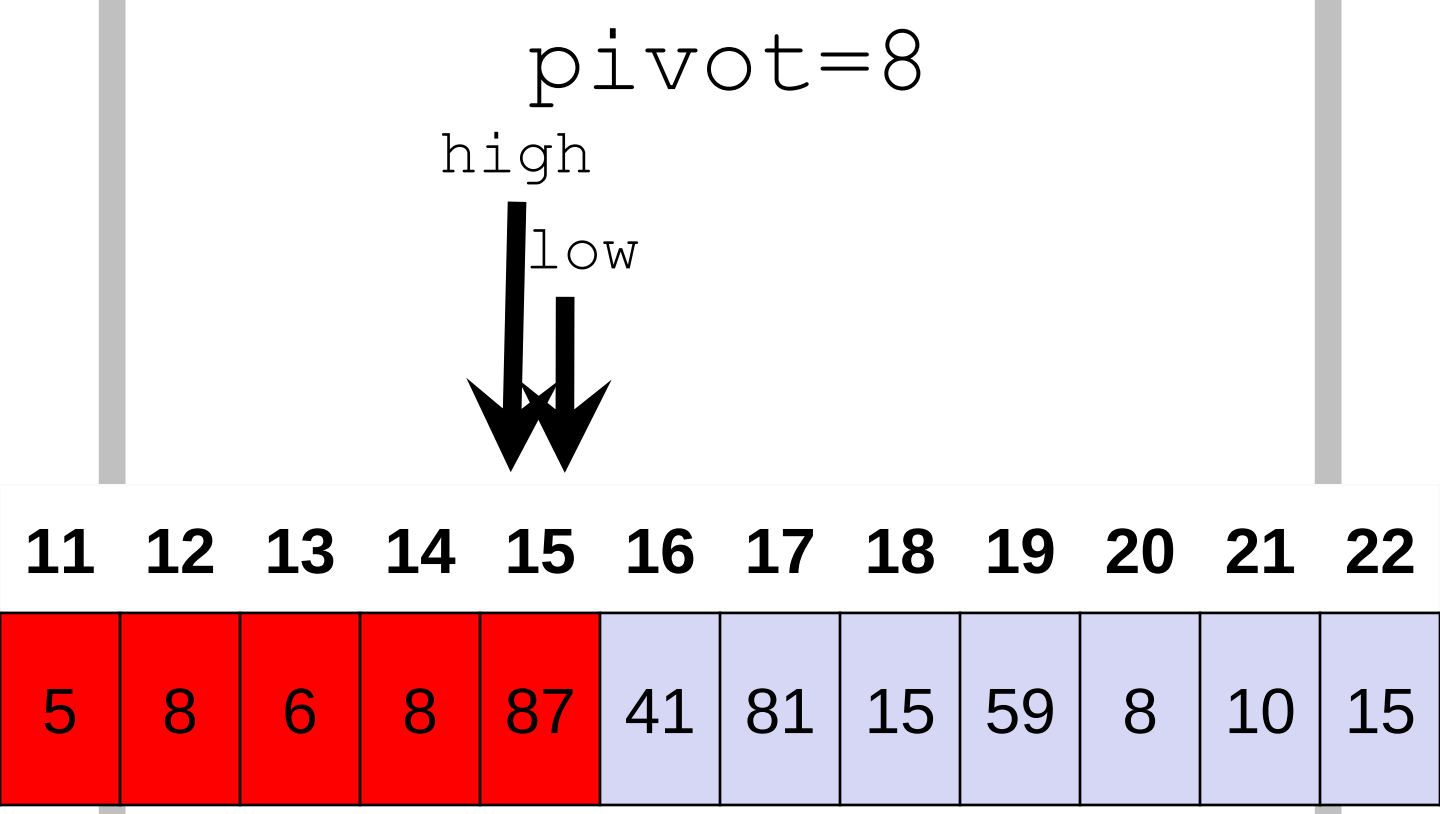
pivot=8

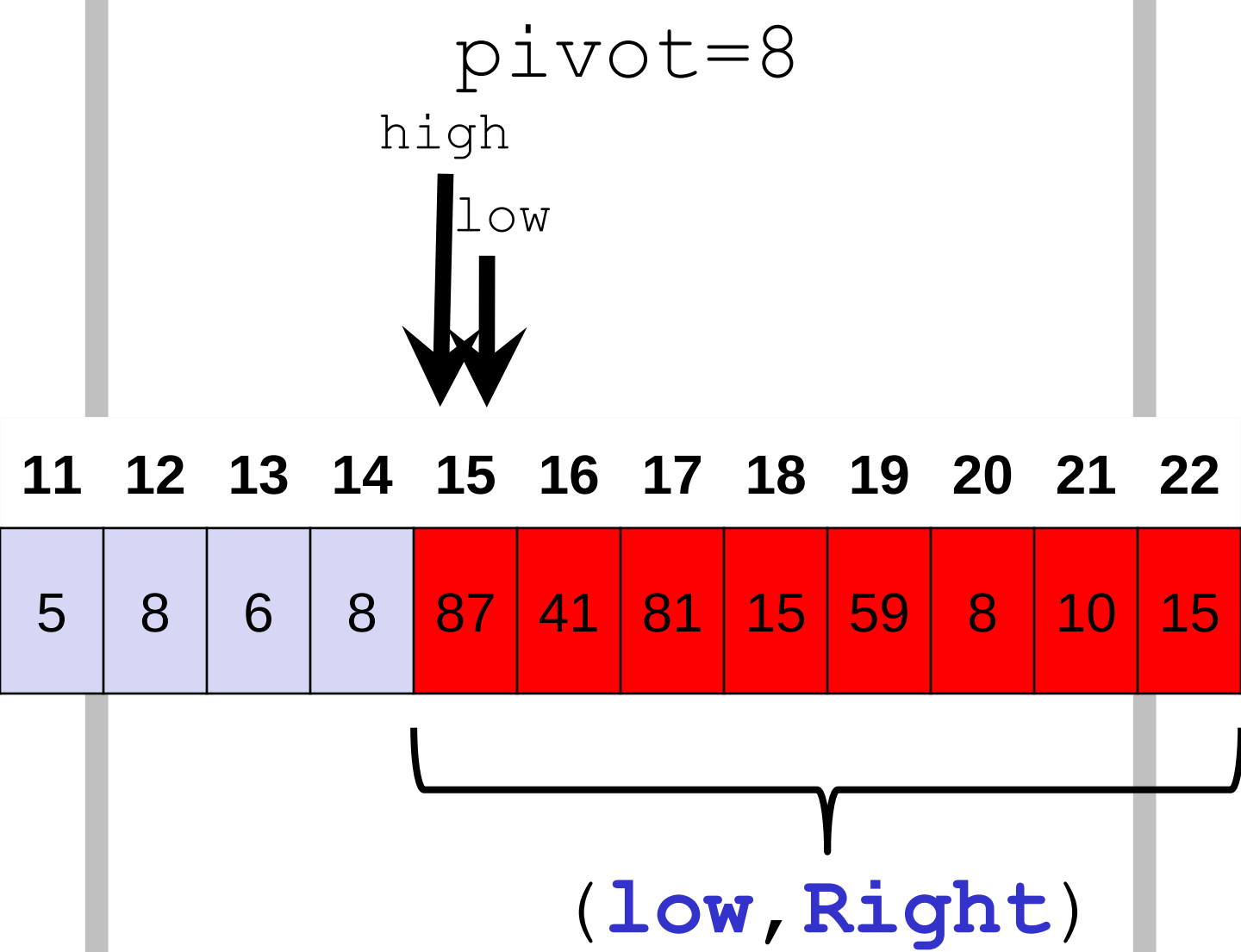


11	12	13	14	15	16	17	18	19	20	21	22
5	8	6	8	87	41	81	15	59	8	10	15



11	12	13	14	15	16	17	18	19	20	21	22
5	8	6	8	87	41	81	15	59	8	10	15


pivot=8
high
low
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 8 87 41 81 15 59 8 10 15
(Left,high)


pivot=8
high
low
11 12 13 14 15 16 17 18 19 20 21 22
5 8 6 8 87 41 81 15 59 8 10 15
(low,Right)

```
10. void Partition(int a[],
        int Left, int Right,
        int&LastLeft,int&FirstRight)
11. {
12.     int pivot=a[(Left+Right)/2];
13.     int low = Left;
14.     int high = Right;
15.     while(low<high)
16.     {
17.         while (a[low]<pivot)
18.             low++;
19.         while (a[high]>pivot)
20.             high--;
21.         if(low<=high)
22.         {
23.             HoanVi(a[low],a[high]);
24.             low++;
25.             high--;
26.         }
27.     }
28.     LastLeft = high;
29.     FirstRight = low;
30. }
```

Chương 1
THUẬT TOÁN QUICK SORT

ThS. Nguyễn Tấn
Trần Minh Khang

```
11. void QuickSort(int a[],
                   int left,int right)
12. {
13.     if(left>=right)
14.         return;
15.     int low = left;
16.     int high = right;
17.     int pivot=a[(left+right)/2];
18.     do{
19.         while(a[low]<pivot)
20.             low++;
21.         while(a[high]>pivot)
22.             high--;
23.         if(low<=high)
24.         {
25.             HoanVi(a[low],a[high]);
26.             low++;
27.             high--;
28.         }
29.     }while(low<high);
30.     if(left<high)
31.         QuickSort(a,left,high);
32.     if(right>low)
33.         QuickSort(a,low,right);
34. }
```

4. BÀI TẬP LUYỆN TẬP 1

BÀI TẬP LUYỆN TẬP

- Bài toán: Định nghĩa hàm sắp mảng một chiều các số thực tăng dần bằng thuật toán quick sort.
- Hàm cài đặt

```
10. void SapTang(float a[],
                 int n)
11. {
12.     QuickSort(a,0,n-1);
13. }
14. void HoanVi(float&a,float&b)
15. {
16.     float temp=a;
17.     a=b;
18.     b=temp;
19. }
```

```
11. void QuickSort(float a[],
                   int left,int right)
12. {
13.     if(left>=right)
14.         return;
15.     int low = left;
16.     int high = right;
17.     int pivot=a[(left+right)/2];
18.     do{
19.         while(a[low]<pivot)
20.             low++;
21.         while(a[high]>pivot)
22.             high--;
23.         if(low<=high)
24.         {
25.             HoanVi(a[low],a[high]);
26.             low++;
27.             high--;
28.         }
29.     }while(low<high);
30.     if(left<high)
31.         QuickSort(a,left,high);
32.     if(right>low)
33.         QuickSort(a,low,right);
34. }
```

5. BÀI TẬP LUYỆN TẬP 2

- **Bài toán: Định nghĩa hàm sắp mảng một chiều các phân số tăng dần bằng thuật toán quick sort.**
- Cấu trúc dữ liệu

```
10. struct phanso
11. {
12.     int tu;
13.     int mau;
14. };
15. typedef struct phanso PHANSO;
16. void HoanVi(PHANSO&a,PHANSO&b)
17. {
18.     PHANSO temp=a;
19.     a=b;
20.     b=temp;
21. }
```

BÀI TẬP LUYỆN TẬP

- Hàm cài đặt

```
1. int SoSanh(PHANSO x, PHANSO y)
2. {
3.     float a=(float)x.tu/x.mau;
4.     float b=(float)y.tu/y.mau;
5.     if(a>b)
6.         return 1;
7.     if(a<b)
8.         return -1;
9.     return 0;
10. }
11. void SapTang(PHANSO a[], int n)
12. {
13.     QuickSort(a,0,n-1);
14. }
```

```
11. void QuickSort(float a[],
                   int left,int right)
12. {
13.     if(left>=right)
14.         return;
15.     int low = left;
16.     int high = right;
17.     int pivot=a[(left+right)/2];
18.     do{
19.         while(a[low]<pivot)
20.             low++;
21.         while(a[high]>pivot)
22.             high--;
23.         if(low<=high)
24.         {
25.             HoanVi(a[low],a[high]);
26.             low++;
27.             high--;
28.         }
29.     }while(low<high);
30.     if(left<high)
31.         QuickSort(a,left,high);
32.     if(right>low)
33.         QuickSort(a,low,right);
34. }
```

```
11. void QuickSort(PHANSO a[],
                   int left,int right)
12. {
13.     if(left>=right)
14.         return;
15.     int low = left;
16.     int high = right;
17.     int pivot=a[(left+right)/2];
18.     do{
19.         while(a[low]<pivot)
20.             low++;
21.         while(a[high]>pivot)
22.             high--;
23.         if(low<=high)
24.         {
25.             HoanVi(a[low],a[high]);
26.             low++;
27.             high--;
28.         }
29.     }while(low<high);
30.     if(left<high)
31.         QuickSort(a,left,high);
32.     if(right>low)
33.         QuickSort(a,low,right);
34. }
```

```
11. void QuickSort(PHANSO a[],
                   int left,int right)
12. {
13.     if(left>=right)
14.         return;
15.     int low = left;
16.     int high = right;
17.     PHANSO pivot=a[(left+right)/2];
18.     do{
19.         while(a[low]<pivot)
20.             low++;
21.         while(a[high]>pivot)
22.             high--;
23.         if(low<=high)
24.         {
25.             HoanVi(a[low],a[high]);
26.             low++;
27.             high--;
28.         }
29.     }while(low<high);
30.     if(left<high)
31.         QuickSort(a,left,high);
32.     if(right>low)
33.         QuickSort(a,low,right);
34. }
```

```
11. void QuickSort(PHANSO a[],
                   int left,int right)
12. {
13.     if(left>=right)
14.         return;
15.     int low = left;
16.     int high = right;
17.     PHANSO pivot=a[(left+right)/2];
18.     do{
19.         while(sosanh(a[low],
                         pivot)==-1
20.             low++;
21.         while(sosanh(a[high],
                         pivot)==1)
22.             high--;
23.         if(low<=high)
24.         {
25.             HoanVi(a[low],a[high]);
26.             low++;
27.             high--;
28.         }
29.     }while(low<high);
30.     if(left<high)
31.         QuickSort(a,left,high);
32.     if(right>low)
33.         QuickSort(a,low,right);
34. }
```

6. BÀI TẬP

- Hãy nêu một ưu điểm và một khuyết điểm mà theo bạn là tiêu biểu nhất của phương pháp sắp xếp Quick Sort khi **cài đặt bằng đệ quy**.
- Ưu điểm của cài đặt thuật toán QuickSort bằng đệ quy: Rõ ràng, dễ hiểu và dễ cài đặt.
- Khuyết điểm của cài đặt thuật toán QuickSort bằng đệ quy: Tràn stack trong tình huống có nhiều lời gọi hàm đệ quy khi kích thước của mảng cần sắp xếp lớn.