Computer Architecture

Contents

구현시 주의사항

- **Lab 3의 bubble sorting 코드를 활용**
  - main C code에서 arrOri [8] = {5, 1, 4, 3, 2, 8, 7,6}으로 수정하여 그대로 사용
  - Assembly code를 quick sorting을 수행할 수 있도록 수정
- **수정시 주의사항**
  - Recursive call 할 때는 call 전 레지스터값을 스텍에 저장했다가 call 후 레지스터 값을 복구하는 과정이 필요
  - Lab 1의 recursive call 을 어셈블리코드로 구현한 P15 참조

보고서 준비사항

- 보고서에 포함할 내용
  - 결과내용을 캡쳐
  - 소스코드에 대한 주석/설명
- **HW10 & 11: 소스코드를 압축하여 이메일로 제출하세요.**
- **제출마감: 11월 29일(일) 23:59**

Lab 6 : quick sort

- Quick sort 동작 방식 설명
- 숫자 배열이 주어졌고 정렬하려고 한다.

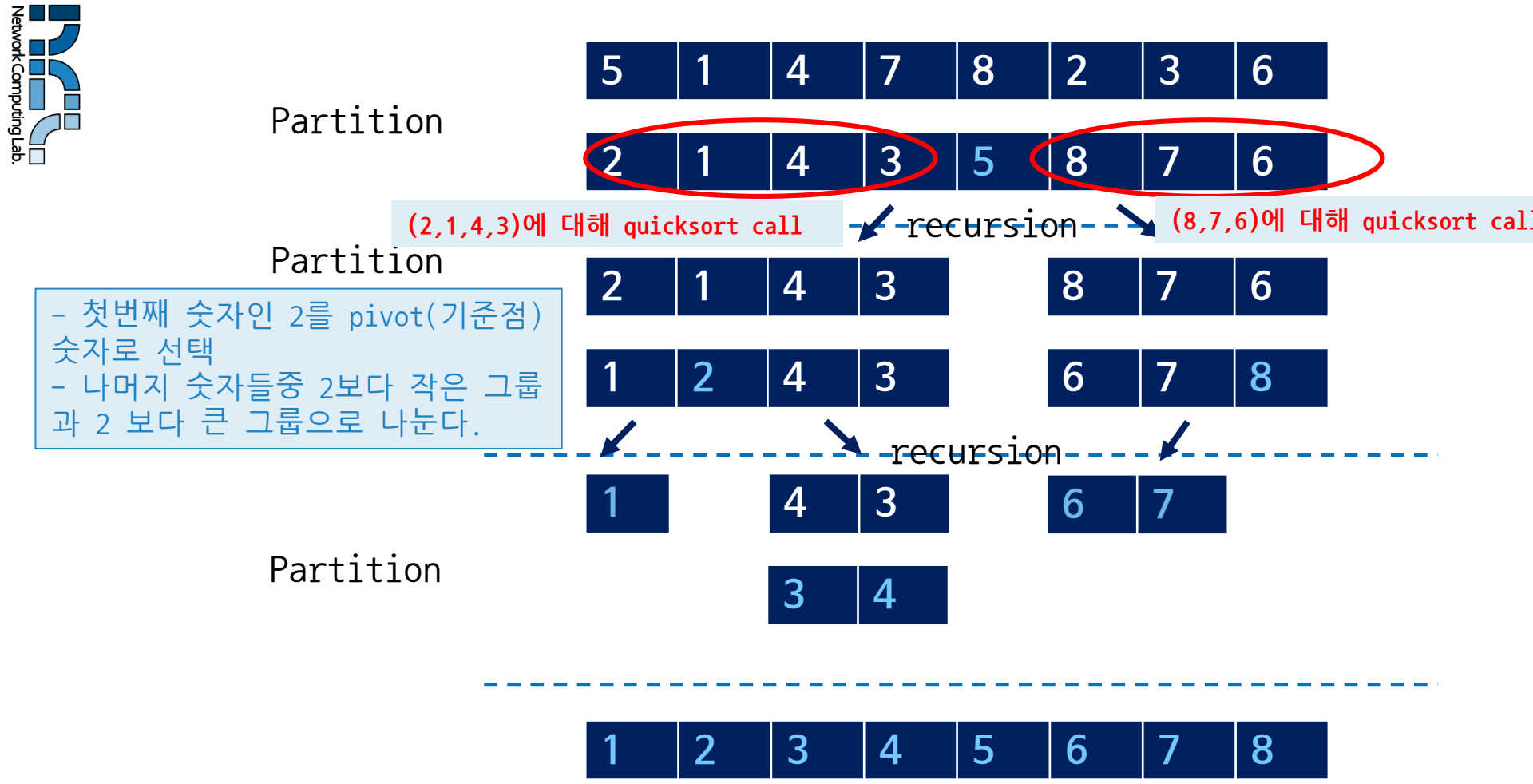
① 배열에서 기준점 pivot 숫자를 하나 선택한다. (현 예제에서는 첫번째 element)
② 우선 배열 내 숫자를 pivot 보다 작은 그룹과 pivot보다 큰 그룹으로 분리
③ 두 그룹 사이에 pivot을 위치시킨다.
④ pivot보다 작은 그룹에 대해 quick sort 적용
⑤ pivot보다 큰 그룹에 대해 quick sort 적용

) Recursive call

- 다음 페이지에서 예제를 그림으로 설명

Lab 6 : quick sort

- 첫번째 숫자인 5를 pivot(기준점) 숫자로 선택
- 나머지 숫자들중 5보다 작은 그룹과 5보다 큰 그룹으로 나눈다.



- 첫번째 숫자인 2를 pivot(기준점) 숫자로 선택
- 나머지 숫자들중 2보다 작은 그룹과 2 보다 큰 그룹으로 나눈다.

Lab 6 : quick sort

단순하게 보면 pivot보다 작은 그룹과 큰 그룹 2개로 나누고 각 그룹에 대해 다시 quicksort를 recursive call 하는 방식입니다.

```
quicksort( array, first, last ) {
partition( array, first, last );
quicksort( array, first, j-1 );
quicksort( array, j+1, last);
}
```

왼쪽의 partition 부분이 실제 C code로 봤을 때 오른 쪽에 해당하는 부분이라는 거죠.

그럼 이제 어떻게 두 개의 그룹으로 분할하는지 다음 페이지부터 상세히 이해해봅시다.

```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){                                  Partition
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```

Lab 6 : quick sort 동작설명 1

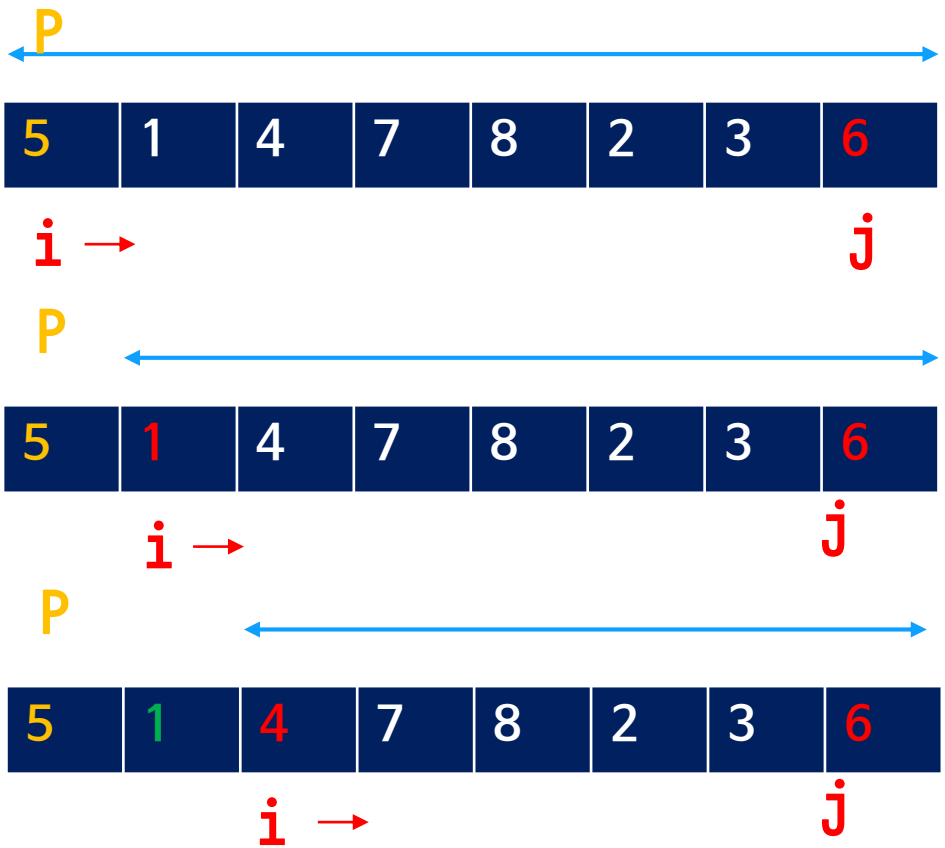
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 2

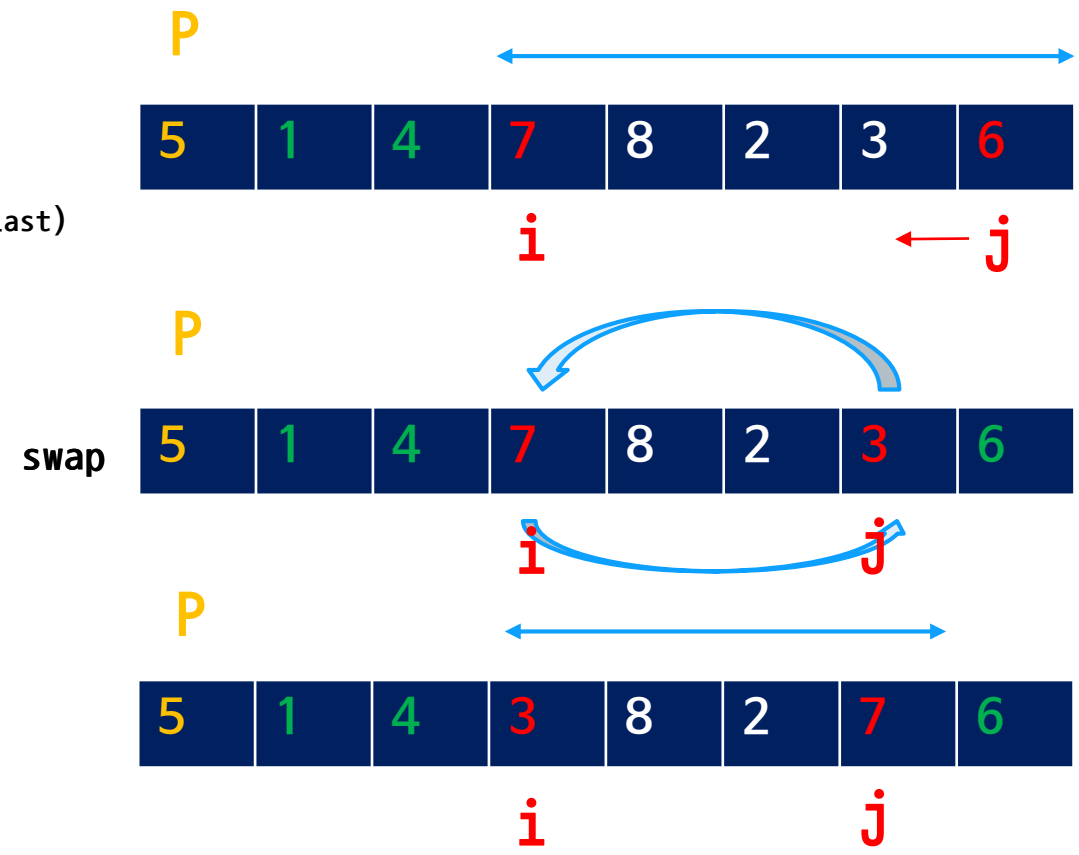
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 3

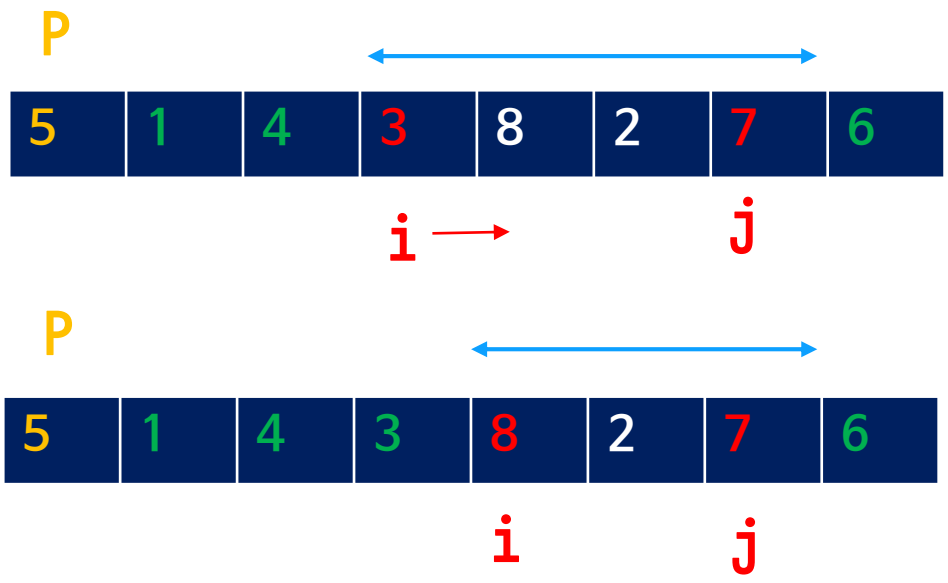
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 4

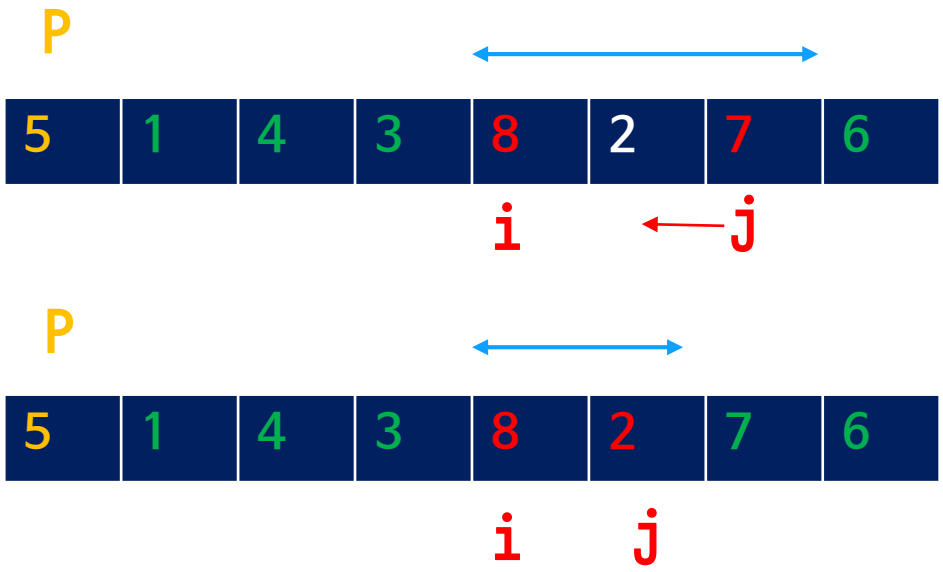
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 5

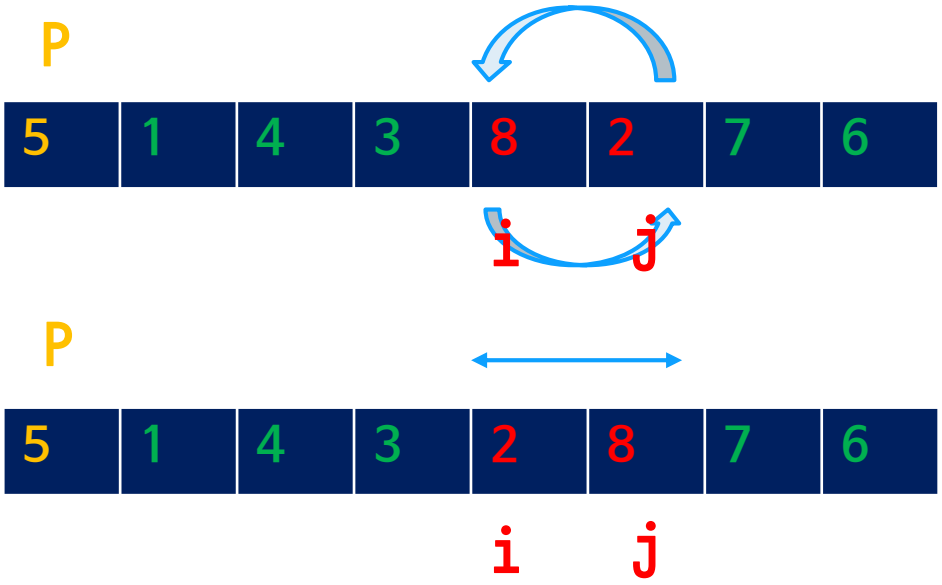
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 6

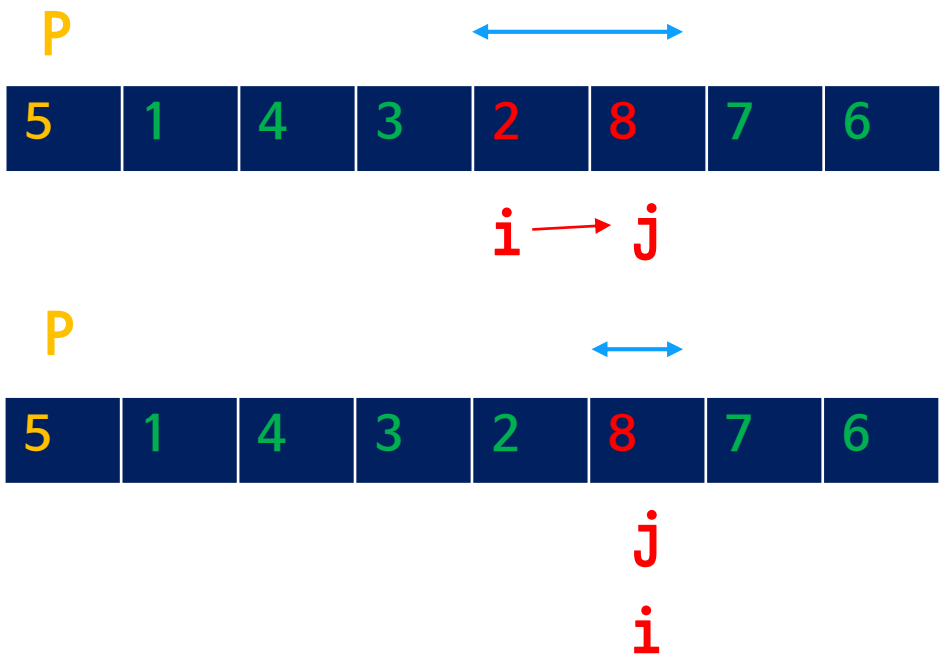
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 7

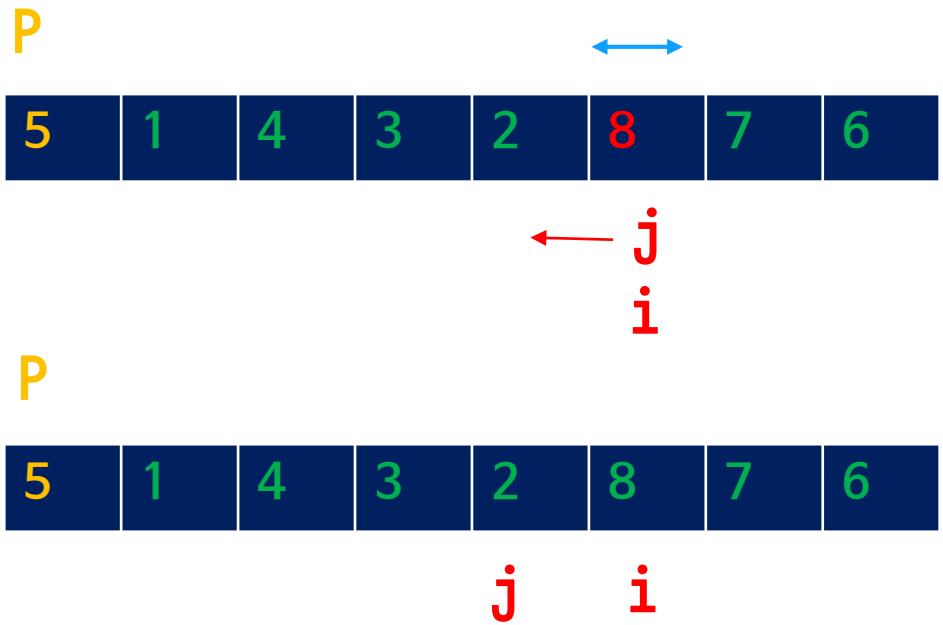
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 8

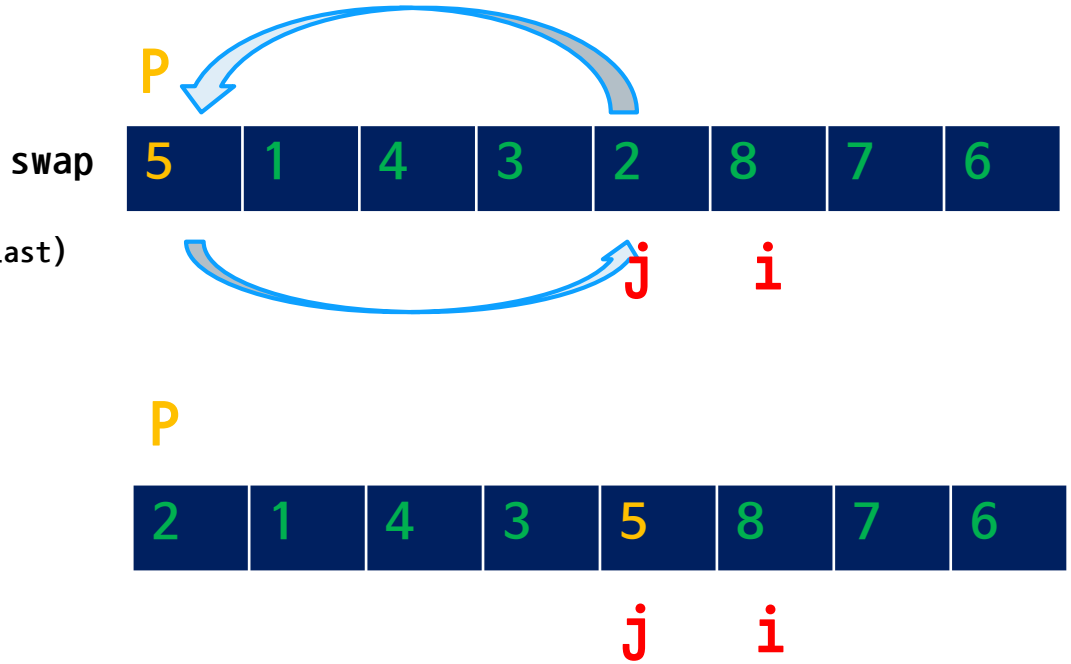
```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```



Lab 6 : quick sort 동작설명 9

```
void quicksort(int array[8],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(array[i]<=array[pivot]&&i<last)
            i++;
         while(array[j]>array[pivot])
            j--;
         if(i<j){
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
         }
      }

      temp=array[pivot];
      array[pivot]=array[j];
      array[j]=temp;
      quicksort(array,first,j-1);
      quicksort(array,j+1,last);

   }
}
```