Chapter 5
Pointers & Arrays
포인터 + 배열

Part 2
5.5~5.6

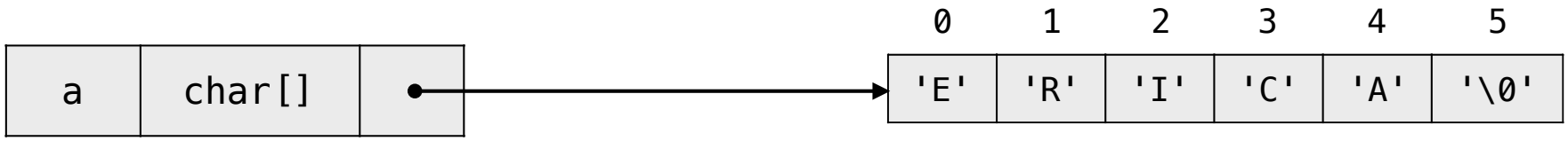
CSE2018 시스템프로그래밍기초
2016년 2학기

한양대학교 ERICA
컴퓨터공학과
도경구

1. Pointers & Addresses
2. Pointers & Function Arguments
3. Pointers & Arrays
4. Address Arithmetic
5. Character Pointers & Functions
6. Pointer Arrays; Pointers to Pointers
7. Multi-dimensional Arrays
8. Initialization of Pointer Arrays
9. Pointers vs. Multidimensional Arrays
10. Command-line Arguments
11. Pointers to Functions
12. Complicated Declarations

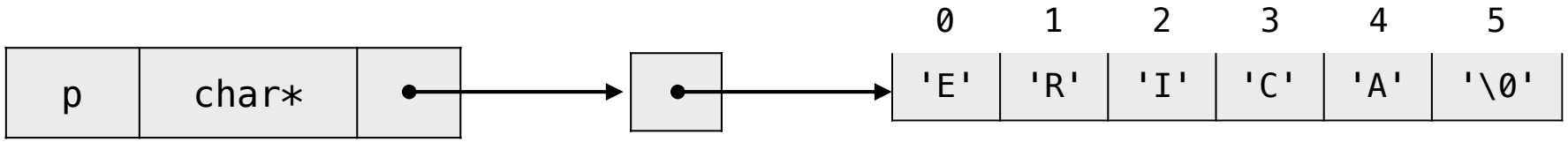
Character Pointers

```
char a[] = "ERICA";
```



문자열 내부 변경 가능

```
char *p = "ERICA";
```

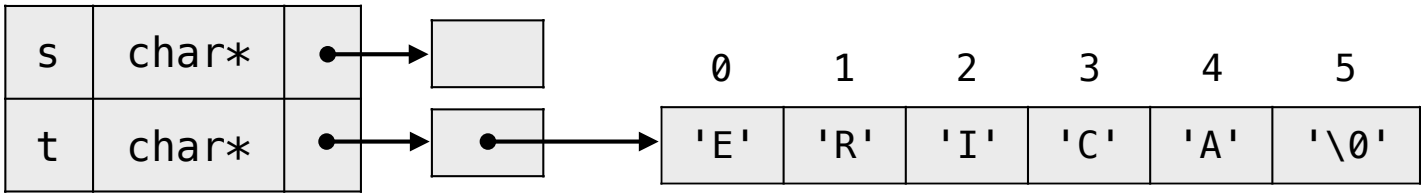


문자열 내부 변경 불가능

String Copy

문자열 t를 s에 복사?

```
char *s;
char *t = "ERICA";
```



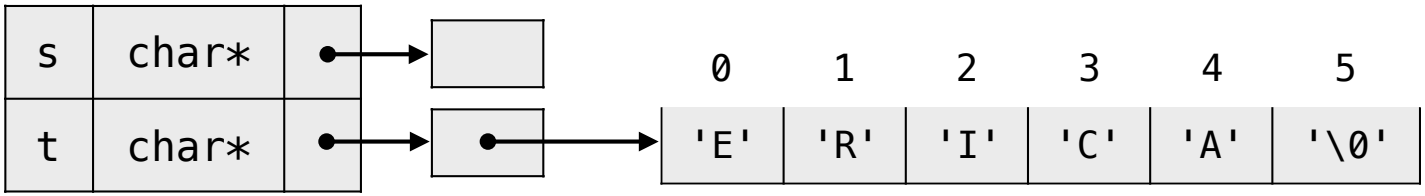

```
s = t
```

?

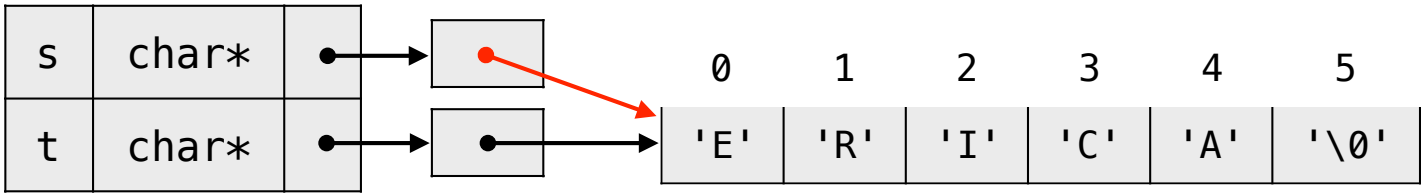
String Copy

문자열 t를 s에 복사?

```
char *s;
char *t = "ERICA";
```




```
s = t
```



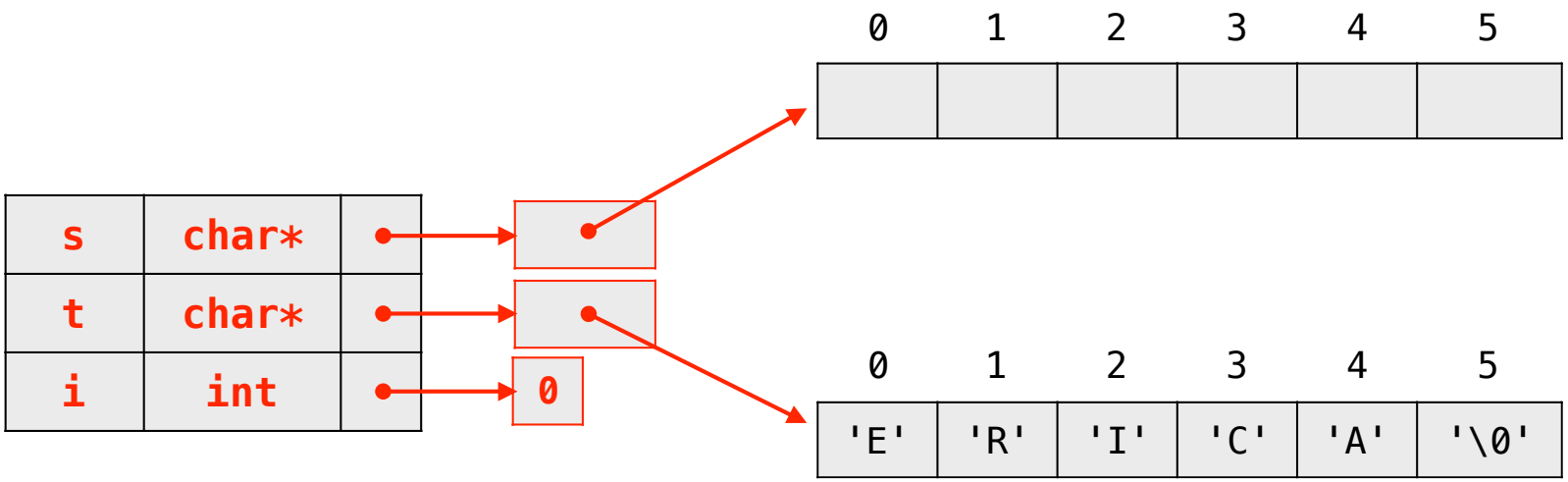
실제로 문자열을 복사한 건 아님!
포인터만 복사한 것임

String Copy
strcpy(s,t)

문자열 t를 s에 복사

array script version

```
void strcpy(char *s, char *t) {
    int i;

    i = 0;
    while ((s[i] = t[i]) != '\0')
        i++;
}
```

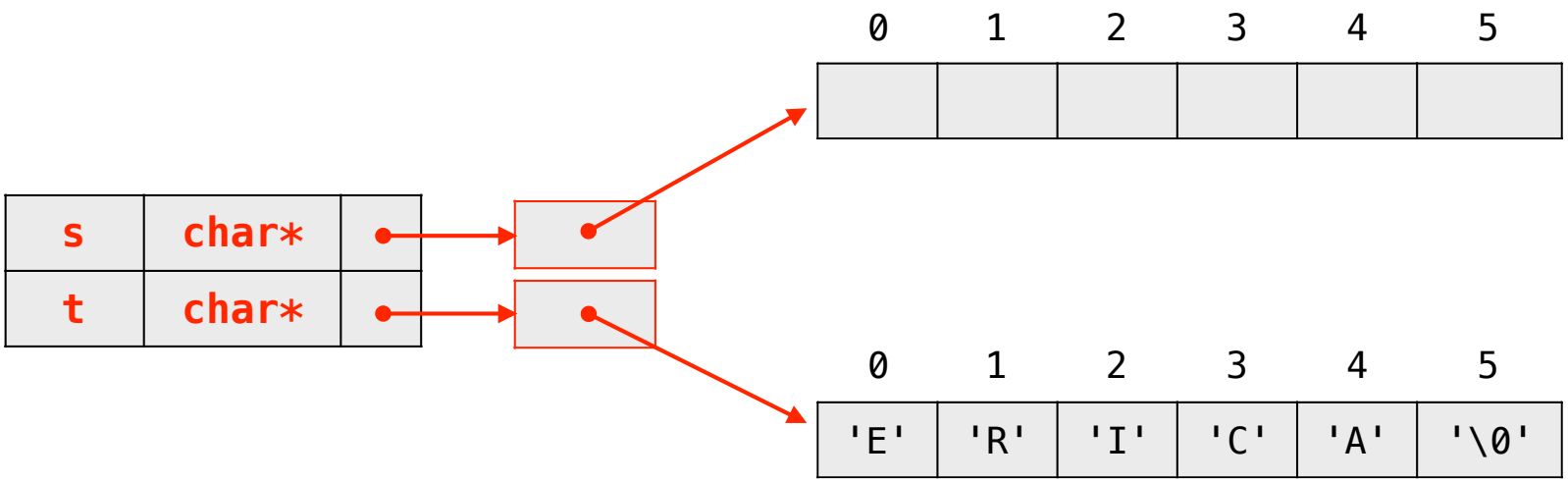


String Copy
strcpy(s,t)

문자열 t를 s에 복사

pointer version I

```
void strcpy(char *s, char *t) {
    while ((*s = *t) != '\0')
        s++;
        t++;
}
```



String Copy
strcpy(s,t)

문자열 t를 s에 복사

pointer version 1

```
void strcpy(char *s, char *t) {
    while ((*s = *t) != '\0')
        s++;
        t++;
}
```

pointer version 2

```
void strcpy(char *s, char *t) {
    while ((*s++ = *t++) != '\0')
        ;
}
```

pointer version 3

```
void strcpy(char *s, char *t) {
    while ((*s++ = *t++))
        ;
}
```

strcpy is in the
standard library
<string.h>

String Compare
strcmp(s,t)

문자열 s와 t를 비교하여
작으면 음수, 같으면 0, 크면 양수를 내줌

array script version

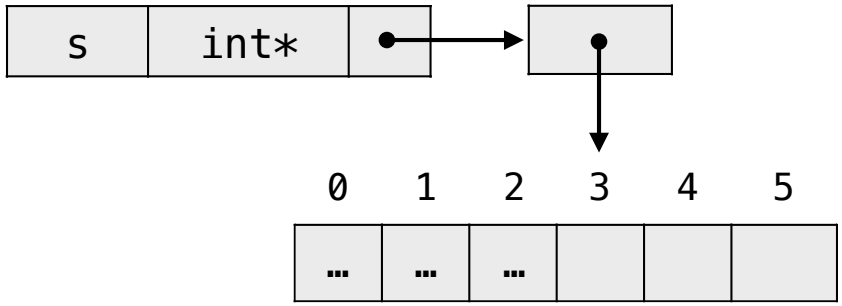
```
int strcmp(char *s, char *t) {
    int i;

    for (i = 0; s[i] == t[i]; i++)
        if (s[i] == '\0')
            return 0;
    return s[i] - t[i];
}
```

pointer version

```
int strcmp(char *s, char *t) {
    for ( ; *s == *t; s++, t++)
        if (*s == '\0')
            return 0;
    return *s - *t;
}
```

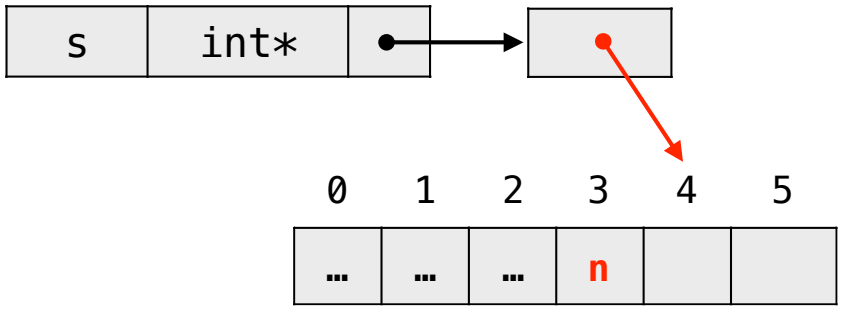
strcmp is in the
standard library
<string.h>

Stack push & pop



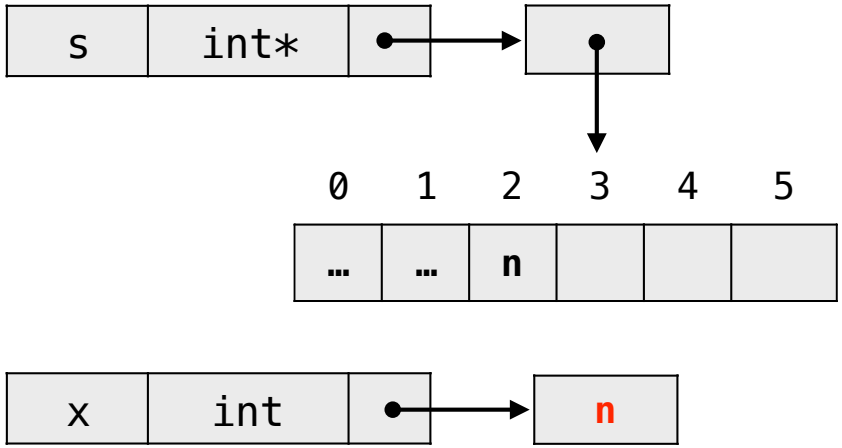
push n onto stack

```
*s++ = n;
```



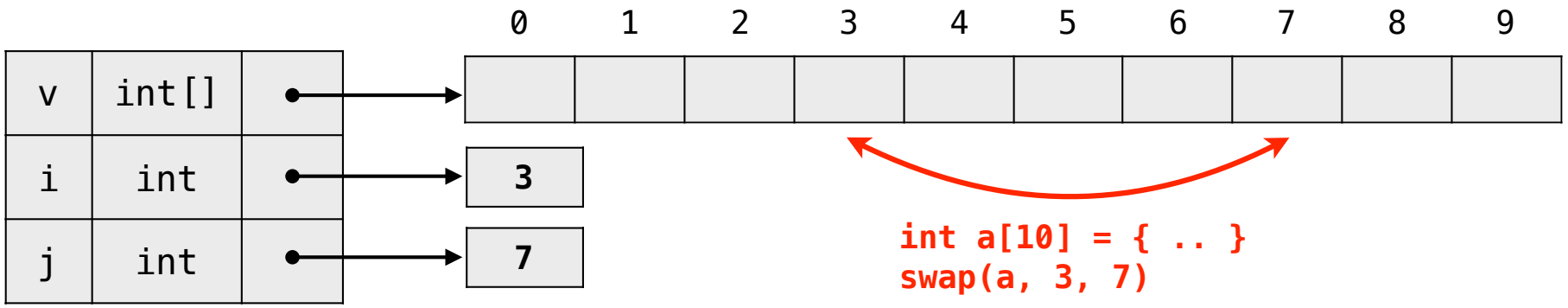
pop top of stack into x

```
x = *--s;
```



Swap Two Elements in Array

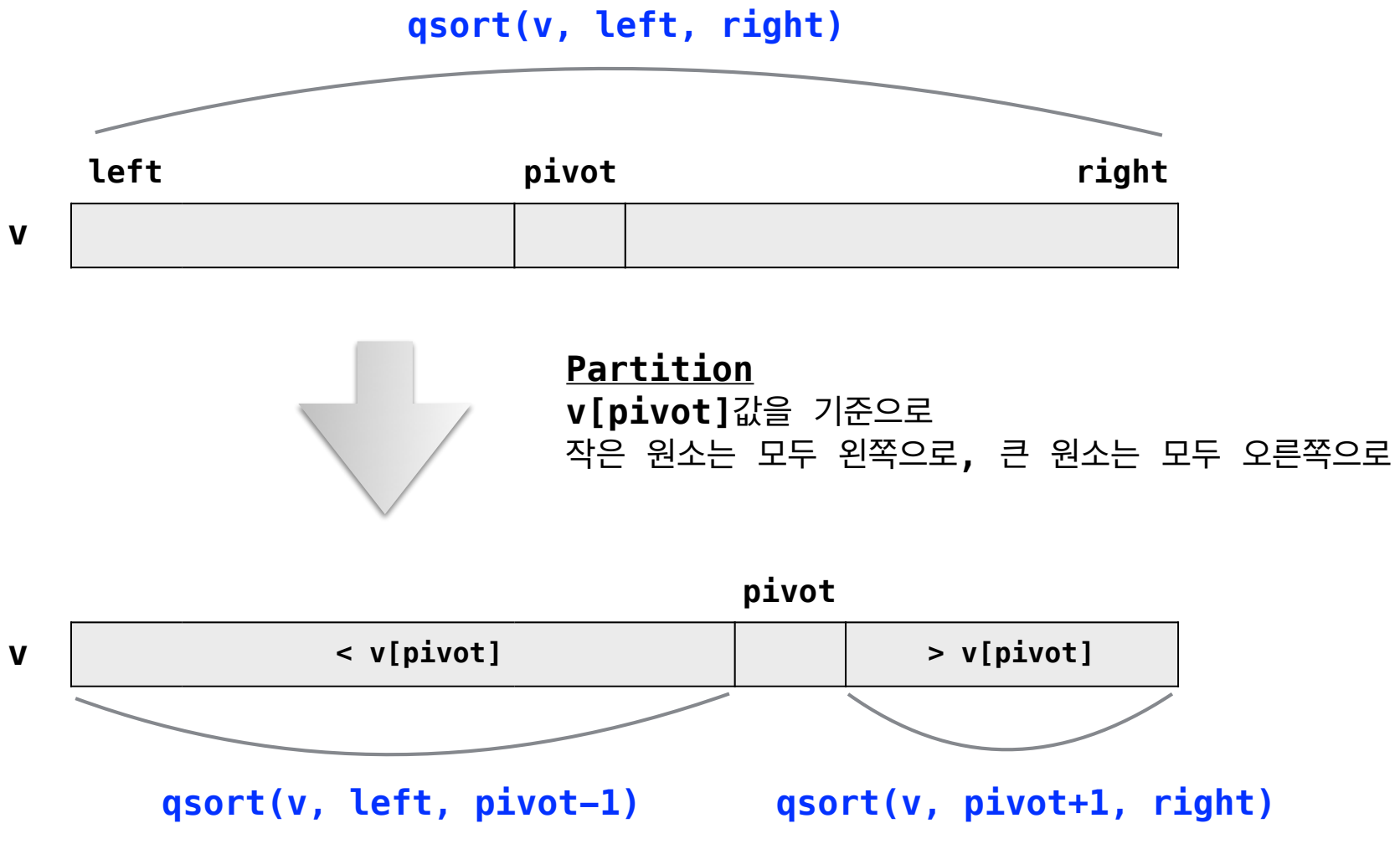
```
/* swap: interchange v[i] and v[j] */
void swap(int v[], int i, int j) {
    int temp;

    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}
```



Quicksort

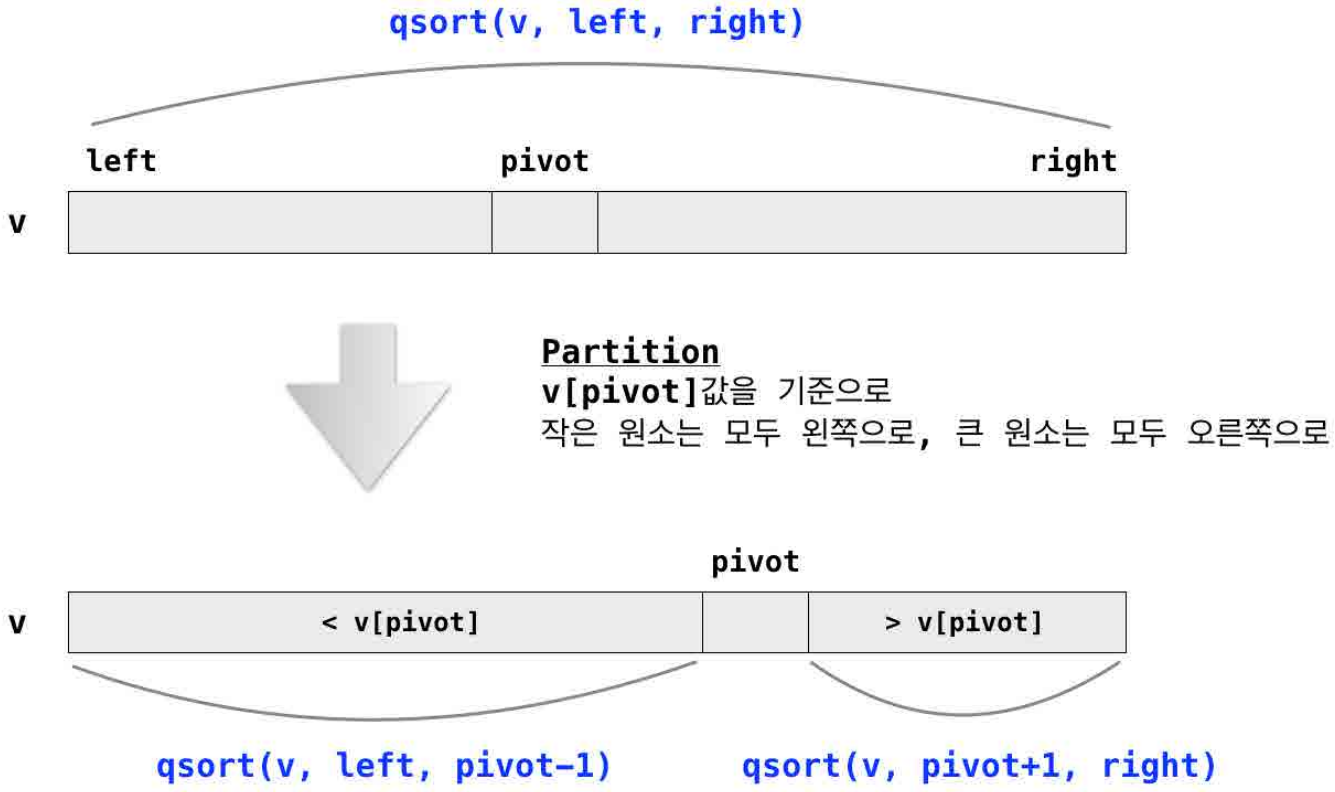
C. A. R. Hoare
(1962)

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right);
```



```c
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

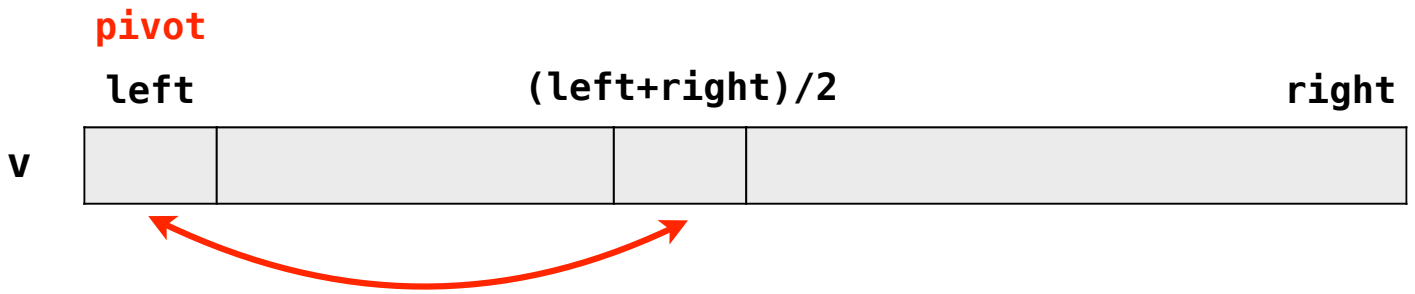
Recursion



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

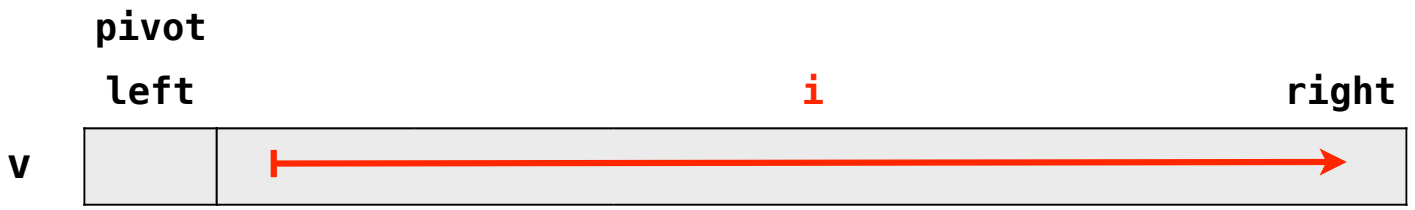
- 기준값은 (left+right)/2 위치의 값
- 기준값을 배열의 맨 앞 값과 교환 => 기준값은 left에 위치
- pivot은 기준값이 있어야 할 위치로 값을 옮기면서 오른쪽으로 이동



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

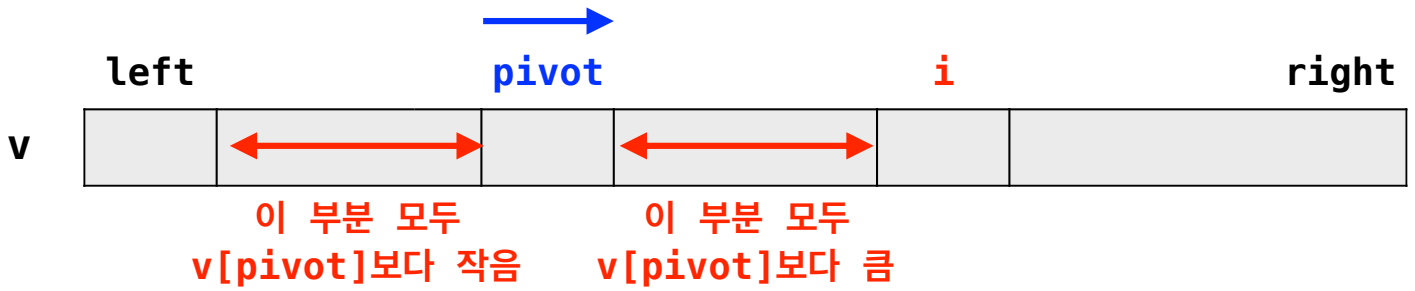
- for 루프의 임무는
- 기준값을 제외한 나머지 원소를 하나씩 검사하여
- 기준값보다 작은 값을 모두 pivot의 왼쪽으로 옮기는 것



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

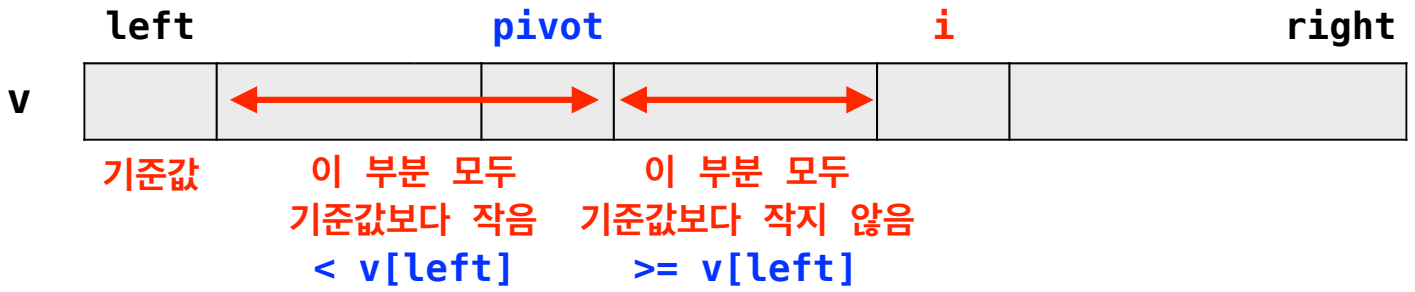
- pivot은 오른쪽으로 움직임
- 아래는 for 루프의 반복 중의 모습



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

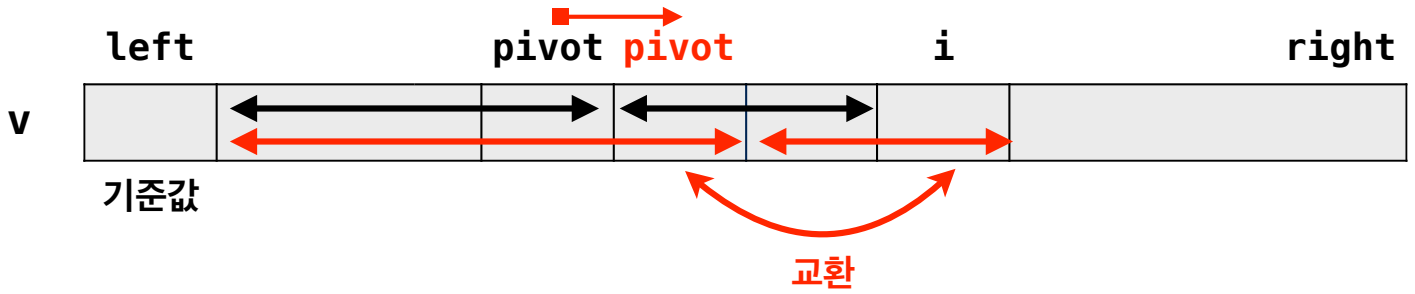
- pivot은 오른쪽으로 움직임
- 아래는 for 루프의 반복 중의 모습



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

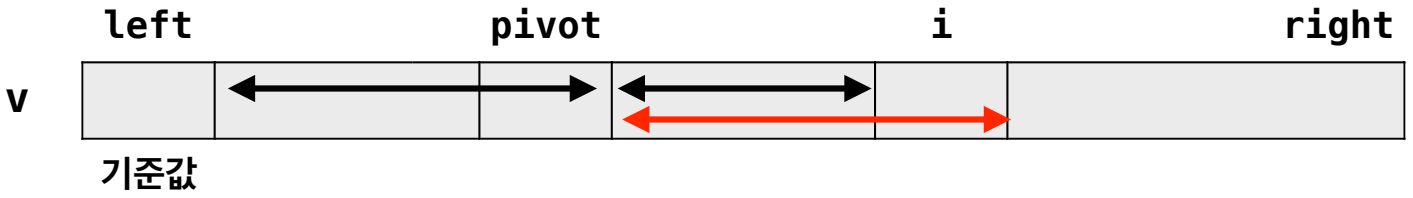
- v[i] < v[left]이면
  - pivot을 오른쪽으로 한칸 이동하고
  - v[pivot]과 v[i]를 교환



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

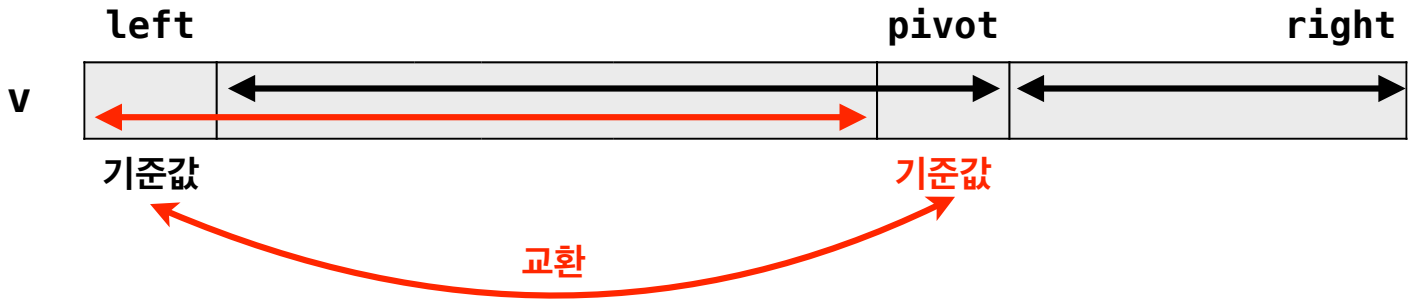
- v[i] < v[left]이 아니면
  - i가 오른쪽으로 한칸 이동



Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

- 기준값을 제 위치로 이동

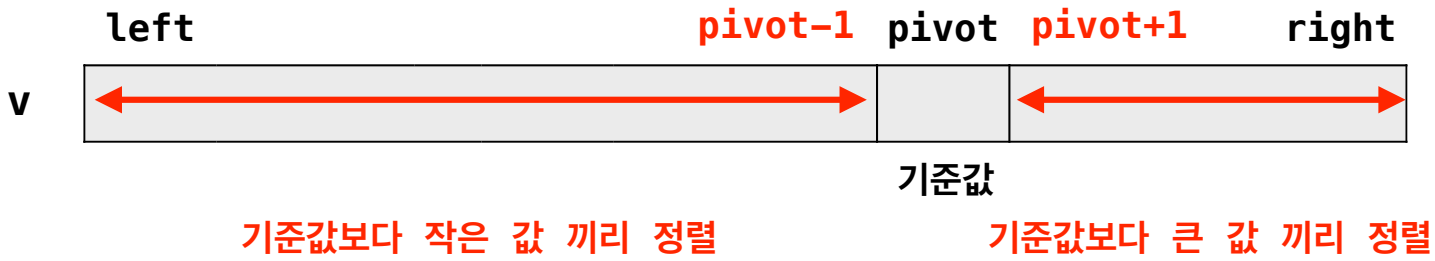


Partition 과정

```
/* qsort: sort v[left]…v[right] into increasing order */
void qsort(int v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```

Recursion

- 기준값을 기준으로 양쪽 배열을 재귀적으로 정렬!



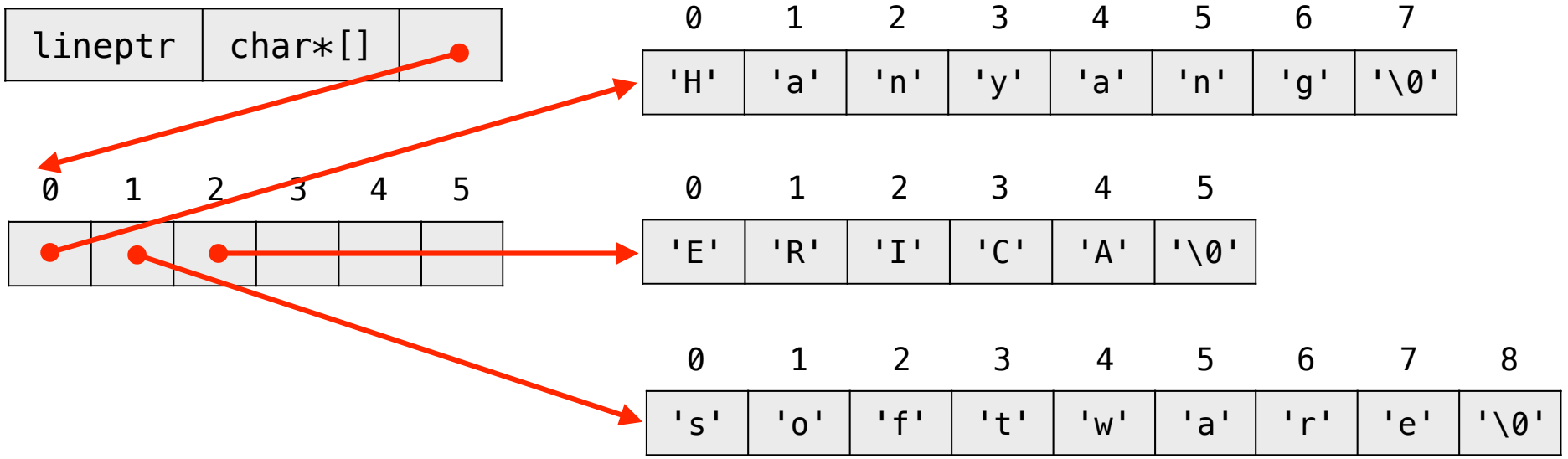
Pointer Arrays (Pointer to Pointers)

사례학습 : 문자열 배열의 정렬

- 문자열이 한줄씩 나열된 텍스트를 알파벳 순으로 정렬하기
- 알고리즘
  - 문자열 한줄씩 배열에 읽어들인다.
  - 문자열배열을 알파벳 순으로 정렬한다.
  - 정렬한 순서로 프린트 한다.

문자열 배열의 표현

```
char *lineptr[6];
```



사례학습 : 문자열 배열의 정렬

```
#include <stdio.h>
#include <string.h>

#define MAXLINES 5000

char *lineptr[MAXLINES];

int readlines(char *lineptr[], int nlines);
void writelines(char *lineptr[], int nlines);

void qsort(char *lineptr[], int left, int right);

main() {
    int nlines;

    if ((nlines = readlines(lineptr, MAXLINES)) >= 0) {
        qsort(lineptr, 0, nlines-1);
        writelines(lineptr, nlines);
        return 0;
    }
    else {
        printf("error: input too big to sort\n");
        return 1;
    }
}
```

사례학습 : 문자열 배열의 정렬

```
/* readline: read line into s, return length */
int readline(char s[], int lim) {
    int c, i;

    i = 0;
    while (--lim > 0 && (c = getchar()) != EOF && c != '\n')
        s[i++] = c;
    if (c == '\n')
        s[i++] = c;
    s[i] = '\0';
    return i;
}
```

사례학습 : 문자열 배열의 정렬

```
#define MAXLEN 1000

char *alloc(int);

int readlines(char *lineptr[], int maxlines) {
    int len, nlines;
    char *p, line[MAXLEN];

    nlines = 0;
    while ((len = readline(line, MAXLEN)) > 0)
        if (nlines >= maxlines || (p = alloc(len)) == NULL)
            return -1;
        else {
            line[len-1] = '\0';
            strcpy(p, line);
            lineptr[nlines++] = p;
        }
    return nlines;
}
```

사례학습 : 문자열 배열의 정렬

```
void writelines(char *lineptr[], int nlines) {
    int i;

    for (i = 0; i < nlines; i++)
        printf("%s\n", lineptr[i]);
}
```

```
void writelines(char *lineptr[], int nlines) {
    int i;

    while (nlines-- > 0)
        printf("%s\n", *lineptr++);
}
```

사례학습 : 문자열 배열의 정렬

```
/* swap: interchange v[i] and v[j] */
void swap(char *v[], int i, int j) {
    char *temp;

    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}
```

사례학습 : 문자열 배열의 정렬

```
/* qsort: sort v[left]...v[right] into increasing order */
void qsort(char *v[], int left, int right) {
    int i, last;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    pivot = left;
    for (i = left+1; i <= right; i++)
        if (strcmp(v[i], v[pivot]) < 0) /* (v[i] < v[left]) */
            swap(v, ++pivot, i);
    swap(v, left, pivot);
    qsort(v, left, pivot-1);
    qsort(v, pivot+1, right);
}
```