퀵 정렬(Quick Sort)

- C. A. R. Hoare에 의해 만들어짐
- 평균적으로 수행속도가 빠른 방법으로 널리 사용됨
- 기본 정렬 방법 : 분할정복(divide and conquer)

  어떤 제어 값을 중심으로 두 개의 데이터 집합으로 분할한다.

  제어값 pivot을 중심으로

  list[0] 에서 list[j-1]은 pivot보다 작은 값(group1)을

  list[j+1]에서 list[n-1]은 pivot보다 큰 값(group2)을 가지도록 분할한다.

  이때 j위치가 pivot의 위치이므로 list[j]와 pivot을 교환한다.
- 나누어진 group1과 group2의 데이터에 대하여 다시 재귀적으로 quick_sort함수를 call하여 처리한다.

퀵 정렬(Quick Sort)의 이해

```
void quick_sort(int a[], int left, int right) {
        int pivot, I, j, tmp;
        if (left < right)  {
             i = left;   j = right+1;  pivot = a[left];
             while (i < j)  {
                  i 를 증가시켜가며 pivot보다 큰 값을 찾는다.
                  j 를 감소시켜가며 pivot보다 작은 값을 찾는다.
                  if (i < j)
                        i와 j 위치의 값을 교환하여 작은 값은 앞으로 모으고 큰 값은 뒤로 모은다.
             }

             a[j]와 a[left]의 값 교환
             quick_sort(a, left, j-1);
             quick_sort(a, j+1, right);
        }
}
```

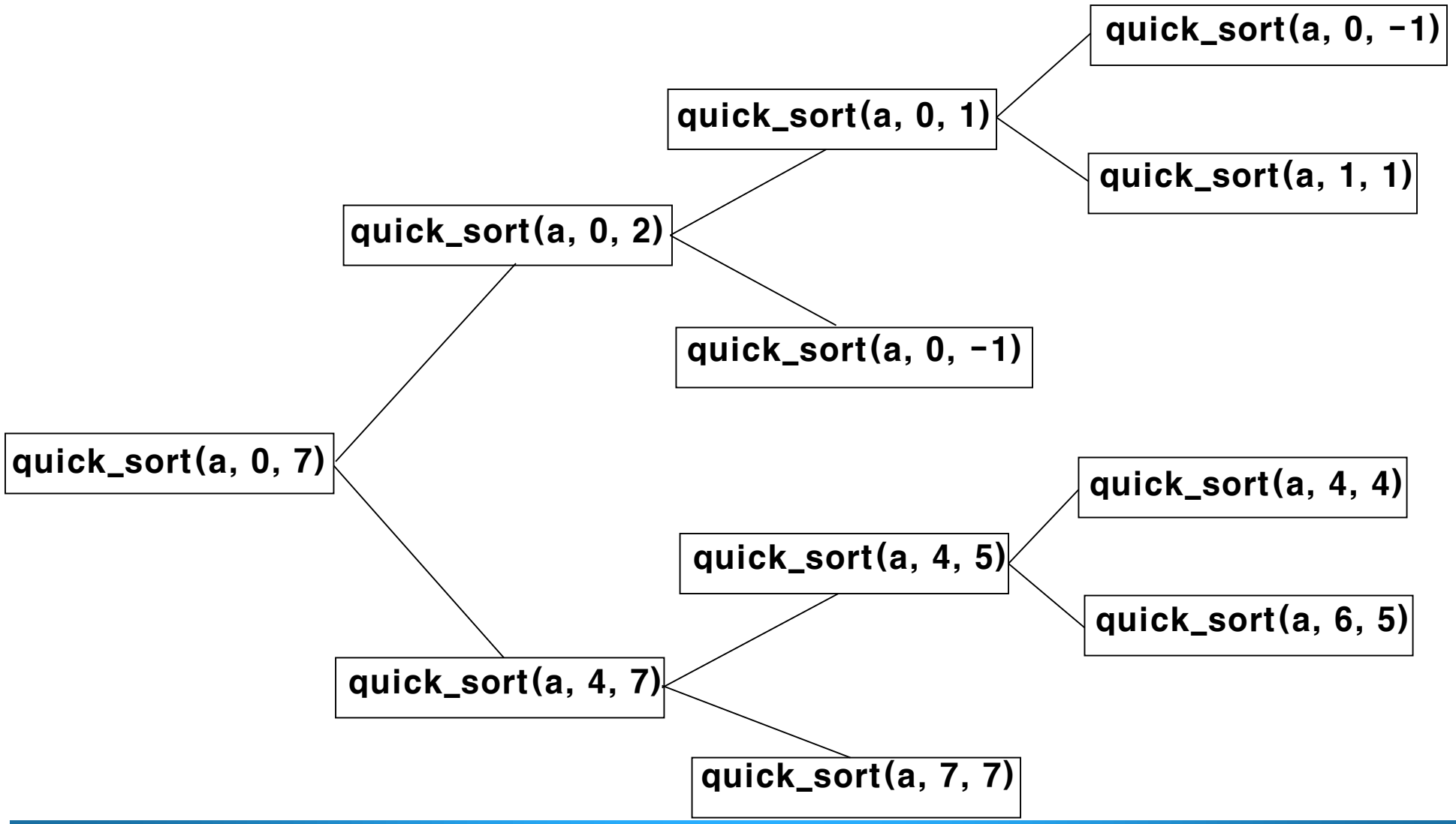
left	right	0	1	2	3	4	5	6	7	i	j
0	7	11	16	5	10	25	7	30	17	0	8
			7				16			1	5
		10	7	5	11	25	16	30	17	4	3

j=3 : pivot의 위치

퀵 정렬 함수

```
void quick_sort(int a[], int left, int right) {
        int pivot, l, j, tmp;

        if (left < right)  {
            i = left;   j = right+1;  pivot = a[left];
            while (i < j)  {
                do  i++;  while (a[i] < pivot) && (i < right);
                do  j--;  while (a[j] > pivot) && (j > left);
                if (i < j) {
                      tmp = a[i]; a[i] = a[j]; a[j] = tmp;
                }
            }
            if (j != left) {
                 tmp = a[j];
                 a[j] = a[left];
                a[left] = tmp;
            }
            quick_sort(a, left, j-1);
            quick_sort(a, j+1, right);
        }
}
```

퀵 정렬(Quick Sort)의 재귀호출



퀵 정렬 정리 및 실습

1. 다음의 배열 list의 데이터를 퀵정렬하기 위해 quick_sort(list, 0, 10)을 수행할 때 다음 while문을 빠져나온 후 변경된 list의 값과 j의 값을 무엇인가요?

0	1	2	3	4	5	6	7	8	9	10
235	107	209	301	101	409	311	205	507	201	333

```
void quick_sort(int a[], int left, int right) {
    int pivot, l, j, tmp;
    if (left < right)  {
        i = left;   j = right+1;  pivot = a[left];
        while (i < j)  {
            do  i++;  while (a[i] < pivot) && (i < right);
            do  j--;  while (a[j] > pivot) && (j > left);
            if (i < j) {
                tmp = a[i]; a[i] = a[j]; a[j] = tmp;
            }
        }
    if (j != left) {
        tmp = a[j];
        a[j] = a[left];
        a[left] = tmp;
    }
    quick_sort(a, left, j-1);
    quick_sort(a, j+1, right);
    }
}
```

퀵 정렬 정리 및 실습

2. 다음의 main()을 이용하여 quick_sort()함수를 실행해 보고 내림차순으로 변경하기 위해 수정해야하는 부분을 적고 프로그램으로 확인해 보시오

235	107	209	301	101	409	311	205	507	201	333

```
void quick_sort(int a[], int left, int right);

main()
{
   int list[]={12, 17, 25, 33, 48, 52, 86, 37, 100, 70};
   int i, n;

   n=sizeof(list)/sizeof(int);
   quick_sort(list, 0, n-1);
   printf("정렬된 데이터 리스트: \n");
   for (i=0; i < n; i++)
            printf("%d ", list[i]);
}
```

```
while (i < j)  {
    do  i++;  while (a[i] < pivot) && (i < right);
    do  j--;  while (a[j] > pivot) && (j > left);
```

퀵 정렬 활용 프로그래밍 연습

사원번호(eid, long int), 이름(name, 19자이하의 문자열), 영어성적(escore, int),
소속부서(dcode, 4자의 문자열)로 구성되어 있는 파일이 준비되어 있다.

1) 파일로부터 데이터를 읽어 사원번호를 기준으로 오름차순으로 정렬하여 출력 파일에 저장한다.
2) dcode를 읽어 그 부서에 해당하는 사원정보를 출력한다.
3) 영어 성적이 큰 값부터 사원정보를 출력한다.

필요한 자료구조

```
typedef struct employee {
    long eid;
    char name[20];
    int escore;
    char dname[5];
} edatatype;
```

파일에 저장된 데이터의 예

```
2007001    Lee1    760    S110
2005121    Park    820    R250
2006111    Kim1    650    S110
2008005    Jung    880    M210
2003543    Um      710    S110
2005770    Lee2    680    M210
2004133    Kim2    890    R250
. . .
```

퀵 정렬 활용 프로그래밍 연습

```
main(int argc, char *argv[])
{
    edatatype edb[50];
    FILE *infile, *out;
    int i=0, j;
    char dcode[5];

    if ((infile = fopen(argv[1], "r")) == NULL) {
       printf("입력 파일을 열 수 없습니다. \n");
       exit(1);
    }
    if ((out = fopen(argv[2], "w")) == NULL) {
       printf("결과 파일을 열 수 없습니다. \n");
       exit(1);
    }

    while (fscanf(infile, "%ld %s %d %s",
            &(edb[i].eid), edb[i].name, &(edb[i].escore), edb[i].dname) != EOF)   i++;
```

퀵 정렬 활용 프로그래밍 연습

1) 파일로부터 데이터를 읽어 사원번호를 기준으로 오름차순으로 정렬하여 출력 파일에 저장한다.

```
void quick_sorta(edatatype a[], int left, int right)
{
   long pivot;  int i, j;  edatatype tmp;

   if (left < right)  {
            i = left; j = right + 1;
            pivot = a[left].eid;
            while (i < j) {
               do  i++;  while ((a[i].eid < pivot) && (i < right));
               do  j--;  while ((a[j].eid > pivot) && (j > left));
               if (i < j) {  tmp = a[i];    a[i] = a[j];   a[j] = tmp; }
            }
            if (j != left) { tmp = a[j];   a[j] = a[left];  a[left] = tmp;  }
            quick_sorta(a, left, j-1);
            quick_sorta(a, j+1, right);
   }
}
```

```
quick_sorta(edb, 0, i-1);
```

퀵 정렬 활용 프로그래밍 연습

2) dcode를 읽어 그 부서에 해당하는 사원정보를 출력한다.

```
printf("Sales Department      : S110\n");
printf("Research Department   : R250\n");
printf("Managemnet Department : M210\n");

printf("\nEnter the Department code : ");
scanf("%s", dcode);

for (j=0; j < i ; j++)
   if (strcmp(dcode, edb[j].dname) == 0)
       printf("%ld\t%s\t%d\n", (edb[j].eid), edb[j].name, (edb[j].escore));
```

퀵 정렬 활용 프로그래밍 연습

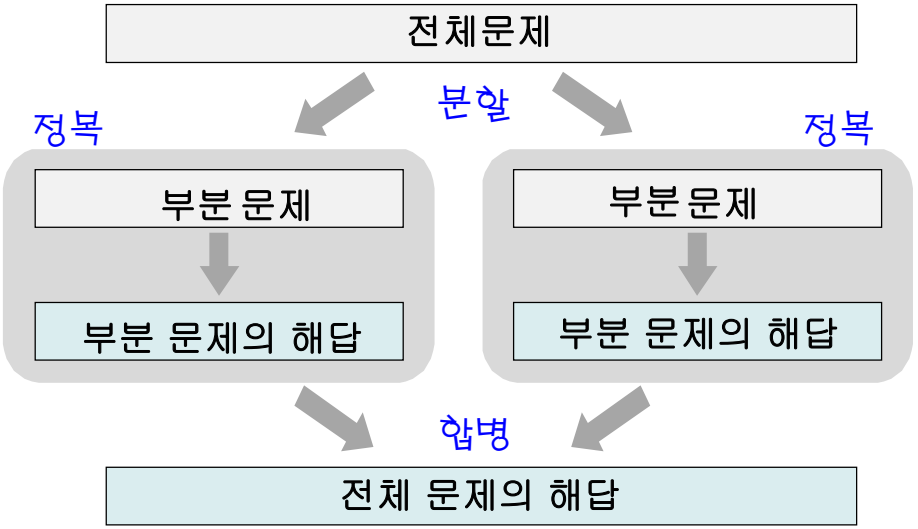
3) 영어 성적이 큰 값부터 사원정보를 출력한다.(내림차순으로 정렬)

```
void quick_sortd(edatatype a[], int left, int right)
{
   int pivot;  int i, j;  edatatype tmp;

   if (left < right)  {
            i = left; j = right + 1;
            pivot = a[left].escore;
            while (i < j) {
               do  i++;  while ((a[i].escore > pivot) && (i < right));
               do  j--;  while ((a[j].escore < pivot) && (j > left));
               if (i < j) {  tmp = a[i];    a[i] = a[j];   a[j] = tmp; }
            }
            if (j != left) { tmp = a[j];   a[j] = a[left];  a[left] = tmp;  }
            quick_sortd(a, left, j-1);
            quick_sortd(a, j+1, right);
   }
}
```

```
quick_sortd(edb, 0, i-1);
```

합병 정렬(Merge Sort)의 이해

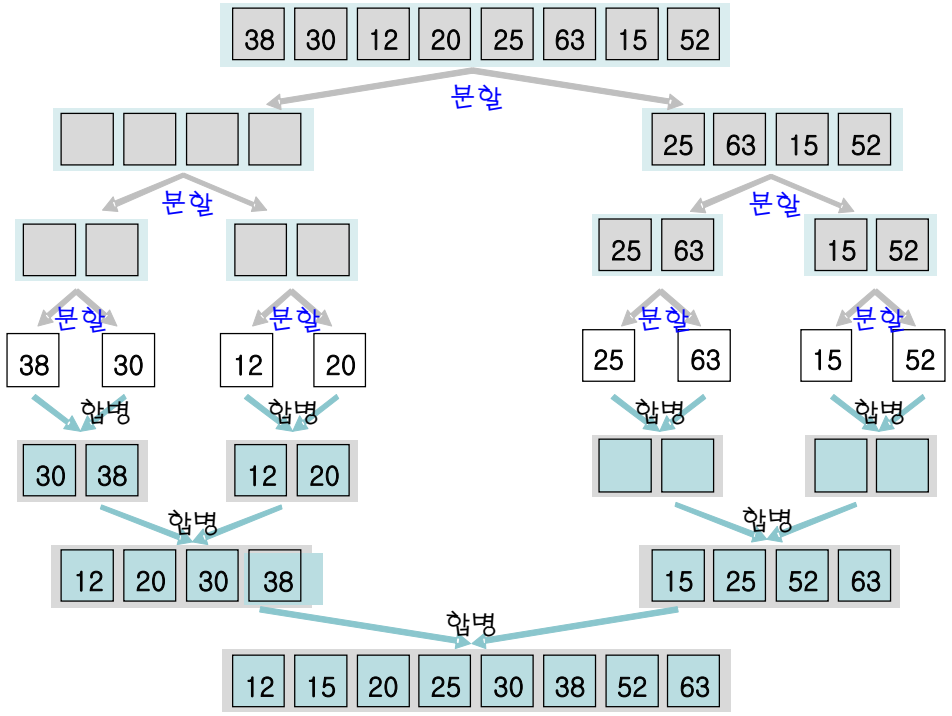
- John von Neumann이 제안한 분할정복(divide and conquer) 방법
- 문제를 2개의 문제로 나누고 각 각 해결한 후
  두 개의 결과를 합해 원래의 문제를 해결하는 방법으로
  재귀(recursive) 기법을 이용하여 해결한다.




```
void merge_sort(int a[], int left, int right)

    if (left < right)  {
        mid = (left+right)/2
        merge_sort(a, left, mid)
        merge_sort(a, mid+1, right)
        merge(a, left, mid, right)
    }
```

합병 정렬(Merge Sort)의 이해

분할정복(divide and conquer) 방법



[합병정렬의 분할-합병 그래프]

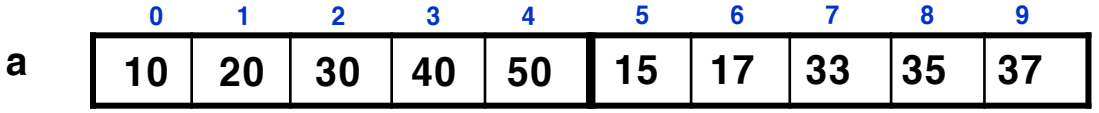
```
void merge_sort(int a[], int left, int right) {

        if (left < right)  {
            mid = (left+right)/2
            merge_sort(a, left, mid)
            merge_sort(a, mid+1, right)
            merge(a, left, mid, right)
        }
}
```

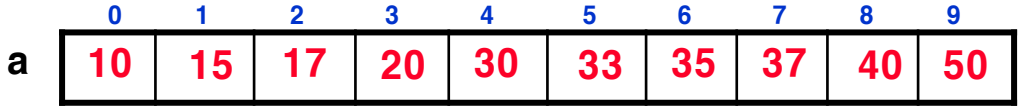
합병 정렬(Merge Sort)

- 하나의 리스트에 정렬된 두 그룹(g1. g2)의 데이터를 합병 merge(a, left, mid, right)

- g1의 현재 위치를 나타내는 인덱스를 i, g2의 현재위치를 나타내는 인덱스를 j,
  정렬된 리스트의 현재 위치를 나타내는 인덱스를 k라 할 때(초기치 i=left, j=mid+1, k=left)
  a[i] <= a[j] 이면 slist[k] = a[i], i++, k++
  a[i] > a[j] 이면 slist[k] = a[j], j++, k++

  어느 한 그룹을 다 처리하면 나머지 리스트의 데이터를 복사한다.
  a1의 데이터의 수 n1, a2의 데이터의 수를 n2라고 할 때
  while (j <= right) slist[k]=a[j], k++, j++
  while (i <= mid) slist[k]=a[i], k++, i++





합병 정렬(Merge Sort) 함수

```
void merge(int a[],  int left, int mid, int right)
{
    int i=left, j=mid+1, k=left, h;
    while (i <= mid  && j <= right) {
        if (a[i] <= a[j]) {
            slist[k] = a[i];
            i++; k++;
        }
        else {slist[k] = a[j], j++, k++
         }
    }
    if (i == mid+1)
        while (j <= right) slist[k++] = a[j++];
    else
        while (i <= mid) slist[k++] = a[i++];
    for (h=left ; h<=right ; h++ )
        a[h] = slist[h];
}
```

	0	1	2	3	4	5	6	7	8	9
a	10	20	30	40	50	15	17	33	35	37

	0	1	2	3	4	5	6	7	8	9
a	10	15	17	20	30	33	35	37	40	50

합병정렬 알고리즘의 활용

- 정렬된 두 데이터 집합을 하나의 정렬된 데이터집합으로 합병
- a1, a2, a의 현재 위치를 나타내는 인덱스를 i, j, k라 할 때(초기치 i=j=k=0)

  a1[i] < a2[j] 이면 a[k] = a1[i], i++, k++

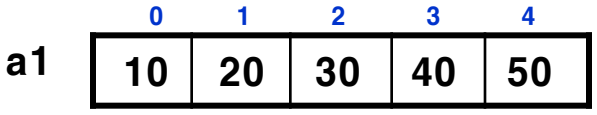
  a1[i] > a2[j] 이면 a[k] = a2[j], j++, k++

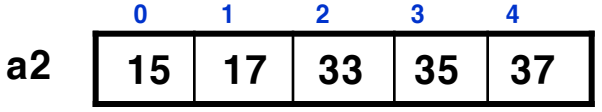
  어느 한 list를 다 처리하면 나머지 리스트의 데이터를 복사한다.

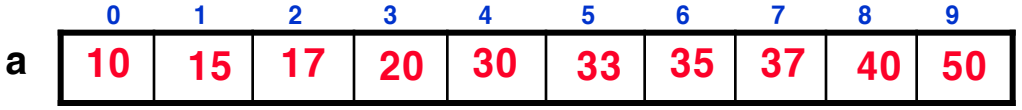
  a1의 데이터의 수 n1, a2의 데이터의 수를 n2라고 할 때

  while (j < n2) a[k]=a2[j], k++, j++

  while (I < n1) a[k]=a1[i], k++, i++

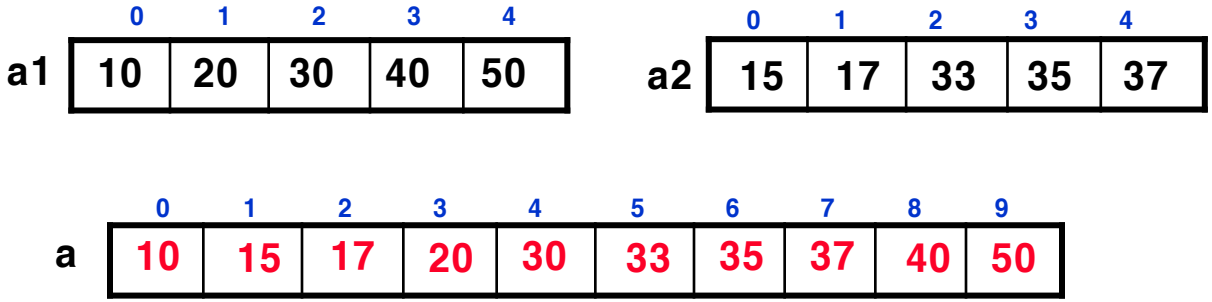






합병정렬 알고리즘의 활용

```
int merge2(int a1[], int a2[], int a[], int n1, int n2)
{
    int i=0, j=0, k=0;

    while (i < n1 && j < n2) {
        if (a1[i] < a2[j])
            a[k++] = a1[i++];
        else if (a1[i] > a2[j])
            a[k++] = a2[j++];
        else  {
            a[k++] = a1[i++];
            j++;
        }
    }
    if (i == n1)
        while (j < n2) a[k++] = a2[j++];
    else
        while (i < n1) a[k++] = a1[i++];

    return k;
}
```



합병정렬 알고리즘의 활용

4개의 정렬된 데이터 리스트를 준비하여 3번의 합병정렬함수를 call하여 정렬된 전체 리스트를 출력하는 프로그램을 작성한다.

```
dnum1 = merge(list1, list2, tlist1, n1, n2)
dnum2 = merge(list3, list4, tlist2, n3, n4)
dnum = merge(tlist1, tlist2, list, dnum1, dnum2)
```

합병정렬 알고리즘의 활용

1. 다음 main()함수를 기준으로 함수 merge()와 merge_sort()를 작성하여 프로그램을 완성하시오

```
#define MAX_SIZE 100
void print_list(int *list, int n, char *mesg);
void merge(int list[], int left, int mid, int right);
void merge_sort(int list[], int left, int right );

main()
{
    int list[]={32, 15, 20, 55, 40, 10, 27, 30, 70, 60, 50, 90, 66, 33};
    int n;

    n = sizeof(list)/sizeof(int);

    print_list(list, n, "데이터 리스트  : \n");

    merge_sort(list, 0, n-1);

    print_list(list, n,  " 정렬된 데이터 리스트 : \n");

}
```

```
void print_list(int *list, int n, char *mesg)
{
    int i;
    printf(mesg);
    for (i=0; i < n; i++) {
        printf("%d   ", list[i]);
        if ((i%7) == 6) printf("\n");
    }
    printf("\n\n");
}
```

합병정렬 정리

합병정렬을 위한 다음 함수에서 내림차순으로 정렬하려면 무엇을 변경해야하는가?

```
void merge(int a[],  int left, int mid, int right)
{
    int i=left, j=mid+1, k=left, h;
    while (i <= mid  && j <= right) {
        if (a[i] < a[j])
            slist[k++] = a[i++];
        else {
            slist[k++] = a[j++];
        }
    }
    if (i == mid+1)
        while (j <= right) slist[k++] = a[j++];
    else
        while (i <= mid) slist[k++] = a[i++];
    for (h=left ; h<=right ; h++ )
        a[h] = slist[h];
}
```

```
void merge_sort(int a[], int left, int right) {

    if (left < right)  {
        mid = (left+right)/2
        merge_sort(a, left, mid)
        merge_sort(a, mid+1, right)
        merge(a, left, mid, right)
    }
}
```

합병정렬 정리

2. merge2()는 합병정렬의 개념을 이용하여 정렬된 2개의 데이터 리스트를 합치는 함수이다.
   다음 merge2(a1, a2, a, 5, 7)을 입력으로 주면 whlie을 수행한후 i,j,k의 값은?

a1	10	15	22	25	33

a2	20	30	40	50	60	70	80

```
int merge2(int a1[], int a2[], int a[], int n1, int n2)
{
    int i=0, j=0, k=0;
    while (i < n1 && j < n2) {
      if (a1[i] < a2[j])
         a[k++] = a1[i++];
      else if (a1[i] > a2[j])
         a[k++] = a2[j++];
      else  {
         a[k++] = a1[i++];
         j++;
      }
   }
   printf( “i = %d, j = %d, k = %d\n” , i,  j, k);
   if (i == n1)
      while (j < n2) a[k++] = a2[j++];
   else
      while (i < n1) a[k++] = a1[i++];

   return k;
}
```