**QUICK SORT**

* Consider first element as pivot
* Assign 0 to low, low+1 to i and high to j



* Keep incrementing i, whenever a[i] < = pivote
* Keep decrementing j, whenever pivote < a[j].
* If i<j, exchange a[i] and a[j] and repeat the process
* If i exceeds j, exchange a[low] and a[j]

Example:

low i j, high

**39** 33 11 98 36 69 65 10 88 78 ( 39 >33, incr i)

i

**39** 33 11 98 36 69 65 10 88 78 (39 > 11, incr i)

i

**39**  33 11 98 36 69 65 10 88 78 (39 <98, stop incrementing i, & compare 42 with a[j] i.e 78 )

i j

**39** 33 11 98 36 69 65 10 88 78 ( 39 <78, decr j)

i j

**39** 33 11 98 36 69 65 10 88 78 ( 39 < 88, decr j)

i j

**39** 33 11 98 36 69 65 10 88 78 ( 39 > 10, stop decr j)

Since i < j, exchange a[i] with a[j] and repeat the process.

i j

**39**  33 11 10 36 69 65 98 88 78 (39 >10, incr i)

i j

**39** 33 11 10 36 69 65 98 88 78 ( 39 > 36, incr i)

i j

**39**  33 11 10 36 69 65 98 88 78 ( 39 < 69, decr j)

i j

**39** 33 11 10 36 69 65 98 88 78 (39 < 65, decr j )

i,j

**39** 33 11 10 36 69 65 98 88 78 ( 39 <69, decr j)

j i

**39** 33 11 10 36 69 65 98 88 78 (39 < 36, is false )

Since i exceeds j, exchange a [ low] with a[j].

j i

**36** 33 11 10 **39** 69 65 98 88 78

int partition (int a [ ], int low, int high)

{

int i, j, temp, pivote;

pivote = a [low];

i = low +1;

j = high;

while(1){

while (i < high && pivote > = a [i]) i++;

while (pivote < a[j] ) j - -;

if (i < j) {

temp = a[i] ;

a[i] = a[j] ;

a[j] = temp;

}

else

{

temp = a[low];

a[low] = a[j];

a[j] = temp;

return j;

}

}

}

void quicksort(int a[ ], int low, int high)

{

int j;

if (low < high){

j = partition (a, low, high);

quicksort(a, low, j-1);

quicksort(a, j+1, high);

}

}

**RADIX SORT\\\\\\**

**Discuss an example**

**Function**

int separate(int item, int j)

{

return (item/(int)pow(10,j-1))%10;

}

void radix\_sort(int a[], int n)

{

int i, j, k, m, big, digit;

node \* p[10], temp;

big=largest(a, n);

m=log10(big)+1;

for(j=1; j<=m; j++)

{

for(i=0; i<=9; i++) p[i]=NULL;

for(i=0; i<n; i++)

{

digit=separate(a[i], j);

p[digit]=insert\_rear(a[i], p[digit]);

}

k=0;

for(i=0; i<=9; i++)

{

temp=p[i];

while(temp!=NULL)

{

a[k++]=temp->data;

temp=temp->link;

}

}

}

}

**MERGE SORT**

1. Divide the array into two equal parts.
2. Recursively sort the left part of the array.
3. Recursively sort the right part of the array.
4. Merge the sorted left and right part.

**Discuss examples**

**Simple merge function**

void simple\_merge (int a[ ], int low, int mid, int high )

{

int i, j, k, c [20];

k = low;

i = low;

j = mid +1;

while (i < = mid ) & & (j < = high)

{

if (a [i] < a [j])

c [k ++] = a[i ++];

else

c [k ++] = a [j ++];

}

while ( i < = mid ) c [ k ++ ] = a [ i ++];

while ( j < = high ) c [ k ++] = a [ j ++];

for (i = low ; i < = k -1 ; i ++) a [i] = c[i];

}

**Merge sort function:**

void merge\_sort ( int a [ ], int low, int high )

{

int mid ;

if (low < high )

{

mid = (low + high)/2 ;

merge\_sort (a, low, mid);

merge\_sort (a, mid +1,high);

simple\_merge (a, low, mid, high);

}

}