**Pseudo median**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void swap(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void bubble\_sort(int temp[], int n){

int swap=0;

int i=0;

int j=0;

for(i=0;i<n-1;i++){

for (j = 0 ; j < n - i - 1; j++){

if (temp[j] > temp[j+1]){

swap = temp[j];

temp[j] = temp[j+1];

temp[j+1] = swap;

}

}

}

}

int pseudo(int arr[], int low, int high)

{

int size = high-low+1;

int m,d;

int a[size];

d=0;

for(m=low;m<=high;m++){

a[d] = arr[m];

d++;

}

if(size>0){

if(size<=5){

int i;

bubble\_sort(a,size);

printf("After sorting size <= 5 array is : \n");

for(i=0;i<size;i++){

printf("%d\n",a[i]);

}

int median = a[(size/2)];

printf("Size <= 5 median(IF) : %d\n",median);

return median;

}

else if(size>5){

int i,j,k,x,c,y;

int temp[5];

int new\_size = size - size%5;

int rem\_size = size%5;

printf("rem\_size : %d\n",rem\_size);

if(rem\_size==0){y=0;}

else{y=1;}

int median[(new\_size/5)+y];

if(y!=0){

k=0;

int rem\_arr[rem\_size];

for(i=new\_size;i<=(size-1);i++){

rem\_arr[k] = a[i];

k++;

}

bubble\_sort(rem\_arr,rem\_size);

printf("rem\_arr after sorting : \n");

for(i=0;i<rem\_size;i++){

printf("%d\n",rem\_arr[i]);

}

median[(new\_size/5)] = rem\_arr[(rem\_size/2)];

}

c=0;

for(i=0;i<new\_size;i=i+5){

k=0;

for(j=i;j<i+5;j++){

temp[k] = a[j];

k++;

}

printf("temp array - %d before sorting : \n",c);

for(x=0;x<5;x++){

printf("%d\n",temp[x]);

}

bubble\_sort(temp,5);

median[c] = temp[2];

printf("temp array - %d after sorting: \n",c);

for(x=0;x<5;x++){

printf("%d\n",temp[x]);

}

c++;

}

printf("Median array before sorting : \n");

for(i=0;i<(new\_size/5)+y;i++){

printf("%d\n",median[i]);

}

pseudo(median,0,(new\_size/5)+y-1);

}

}

}

int partition (int arr[], int low, int high){

int pivot=0;

pivot = pseudo(arr,low,high);

int index\_pivot;

for(int i=low;i<=high;i++){

if(arr[i]==pivot){

index\_pivot = i;

}

}

swap(&arr[index\_pivot],&arr[high]);

printf("Median is : %d\n",pivot);

int i = (low-1);

for (int j=low;j<=(high-1);j++){

if (arr[j] <= pivot){

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i+1], &arr[high]);

return (i+1);

}

void quick\_sort(int arr[], int low, int high)

{

if (low < high){

int pi = partition(arr, low, high);

quick\_sort(arr, low, pi - 1);

quick\_sort(arr, pi + 1, high);

}

}

void main(){

int n,i;

printf("Enter the size of array : ");

scanf("%d",&n);

int \*a;

a = (int \*)malloc(n\*sizeof(int));

srand(time(0));

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

a[i] = (rand()%200)+1;

}

printf("Initial Array : \n");

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

printf("%d\n",a[i]);

}

quick\_sort(a,0,n-1);

printf("Final Array after sorting : \n");

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

printf("%d\n",a[i]);

}

}