Quick Sort Program in C

13/5/2020

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Implementation in C

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

#define MAX 7

int intArray[MAX] = {4,6,3,2,1,9,7};

void printline(int count) {
   int i;

   for(i = 0;i < count-1;i++) {
      printf("=");
   }

   printf("=\n");
}

void display() {
   int i;
   printf("[");

   // navigate through all items
   for(i = 0;i < MAX;i++) {
      printf("%d ",intArray[i]);
   }

   printf("]\n");
}

void swap(int num1, int num2) {
   int temp = intArray[num1];
   intArray[num1] = intArray[num2];
   intArray[num2] = temp;
}

int partition(int left, int right, int pivot) {
   int leftPointer = left -1;
   int rightPointer = right;

   while(true) {
      while(intArray[++leftPointer] < pivot) {
         //do nothing
      }

      while(rightPointer > 0 && intArray[--rightPointer] > pivot) {
         //do nothing
      }

      if(leftPointer >= rightPointer) {
         break;
      } else {
         printf(" item swapped :%d,%d\n", intArray[leftPointer],intArray[rightPointer]);
         swap(leftPointer,rightPointer);
      }
   }

   printf(" pivot swapped :%d,%d\n", intArray[leftPointer],intArray[right]);
   swap(leftPointer,right);
   printf("Updated Array: ");
```

```
   return leftPointer;
}

void quickSort(int left, int right) {
   if(right-left <= 0) {
      return;
   } else {
      int pivot = intArray[right];
      int partitionPoint = partition(left, right, pivot);
      quickSort(left,partitionPoint-1);
      quickSort(partitionPoint+1,right);
   }
}

int main() {
   printf("Input Array: ");
   display();
   printline(50);
   quickSort(0,MAX-1);
   printf("Output Array: ");
   display();
   printline(50);
}
```

If we compile and run the above program, it will produce the following result –

Output

```
Input Array: [4 6 3 2 1 9 7 ]
==================================================
 pivot swapped :9,7
Updated Array: [4 6 3 2 1 7 9 ]
 pivot swapped :4,1
Updated Array: [1 6 3 2 4 7 9 ]
 item swapped :6,2
 pivot swapped :6,4
Updated Array: [1 2 3 4 6 7 9 ]
 pivot swapped :3,3
Updated Array: [1 2 3 4 6 7 9 ]
Output Array: [1 2 3 4 6 7 9 ]
==================================================
```