**RANDOMIZED QUICK SORT**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

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

{

  int temp = \*a;

  \*a = \*b;

  \*b = temp;

}

int k = 0;

void permute(int \*\*arr1, int \*arr, int start, int end)

{

  int i;

  if (start == end)

  {

    for (i = 0; i <= end; i++)

    {

      arr1[k][i] = arr[i];

    }

    k++;

  }

  else

  {

    for (int i = start; i <= end; i++)

    {

      swap(&arr[start], &arr[i]);

      permute(arr1, arr, start + 1, end);

      swap(&arr[start], &arr[i]);

    }

  }

}

int partition(int \*arr, int low, int high, int \*countComp, FILE \*f)

{

  int pivot = low + (rand() % (high - low + 1));

  fprintf(f, "Pivot: %d\n", arr[pivot]);

  swap(&arr[high], &arr[pivot]);

  int i = low - 1;

  int j;

  for (j = low; j < high; j++)

  {

    (\*countComp)++;

    if (arr[j] < arr[high])

    {

      i++;

      swap(&arr[i], &arr[j]);

    }

  }

  swap(&arr[i + 1], &arr[high]);

  for (j = low; j <= high; j++)

  {

    fprintf(f, "%d ", arr[j]);

  }

  fprintf(f, "\n");

  return i + 1;

}

void quickSort(int \*arr, int low, int high, int \*countComp, FILE \*f)

{

  if (low < high)

  {

    int pivot = partition(arr, low, high, countComp, f);

    quickSort(arr, low, pivot - 1, countComp, f);

    quickSort(arr, pivot + 1, high, countComp, f);

  }

}

int main()

{

  srand(time(NULL));

  int n, i, j;

  printf("Enter the number of elements: ");

  scanf("%d", &n);

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

  for (i = 0; i < n; i++)

  {

    int num = 10 + (rand() % 90);

    a[i] = num;

  }

  FILE \*f = fopen("output.txt", "w");

  if (f == NULL)

  {

    printf("Error opening file\n");

    exit(1);

  }

  fprintf(f, "The array is: ");

  for (i = 0; i < n; i++)

  {

    fprintf(f, "%d ", a[i]);

  }

  fprintf(f, "\n\n");

  int fact = 1;

  for (i = 1; i <= n; i++)

  {

    fact \*= i;

  }

  int \*\*arr = (int \*\*)malloc(fact \* sizeof(int \*));

  for (i = 0; i < fact; i++)

  {

    arr[i] = (int \*)malloc(n \* sizeof(int));

  }

  permute(arr, a, 0, n - 1);

  int \*countComp = (int \*)malloc(fact \* sizeof(int));

  for (i = 0; i < fact; i++)

  {

    fprintf(f, "Permutation: ");

    for (j = 0; j < n; j++)

    {

      fprintf(f, "%d ", arr[i][j]);

    }

    fprintf(f, "\n");

    countComp[i] = 0;

    quickSort(arr[i], 0, n - 1, &countComp[i], f);

    fprintf(f, "Sorted permutation: ");

    for (j = 0; j < n; j++)

    {

      fprintf(f, "%d ", arr[i][j]);

    }

    fprintf(f, "\n");

    fprintf(f, "Number of comparisons: %d\n\n", countComp[i]);

  }

  float totalComp = 0.0;

  for (i = 0; i < fact; i++)

  {

    totalComp += countComp[i];

  }

  printf("Average comparison: %.2f\n", totalComp / fact);

  fclose(f);

  printf("Output written to output.txt\n");

  return 0;

}