**DAA Assignment 6**

**Quick Sort Using Deterministic and Randomized Variants**

**Neeti Kurulkar**

**#include <iostream>**

**#include <vector>**

**#include <ctime>**

**#include <cstdlib> // For rand()**

**using namespace std;**

**// Global counters for comparisons**

**long long deterministicComparisons = 0;**

**long long randomizedComparisons = 0;**

**// Swap two elements**

**void swap(int &a, int &b) {**

**int temp = a;**

**a = b;**

**b = temp;**

**}**

**// Partition function for deterministic QuickSort (pivot = last element)**

**int deterministicPartition(vector<int>& arr, int low, int high) {**

**int pivot = arr[high];**

**int i = low - 1;**

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

**deterministicComparisons++;**

**if(arr[j] <= pivot) {**

**i++;**

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

**}**

**}**

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

**return i+1;**

**}**

**// Deterministic QuickSort**

**void deterministicQuickSort(vector<int>& arr, int low, int high) {**

**if(low < high) {**

**int pi = deterministicPartition(arr, low, high);**

**deterministicQuickSort(arr, low, pi - 1);**

**deterministicQuickSort(arr, pi + 1, high);**

**}**

**}**

**// Partition function for randomized QuickSort**

**int randomizedPartition(vector<int>& arr, int low, int high) {**

**int pivotIndex = low + rand() % (high - low + 1); // Random pivot**

**swap(arr[pivotIndex], arr[high]); // Move pivot to end**

**int pivot = arr[high];**

**int i = low - 1;**

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

**randomizedComparisons++;**

**if(arr[j] <= pivot) {**

**i++;**

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

**}**

**}**

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

**return i+1;**

**}**

**// Randomized QuickSort**

**void randomizedQuickSort(vector<int>& arr, int low, int high) {**

**if(low < high) {**

**int pi = randomizedPartition(arr, low, high);**

**randomizedQuickSort(arr, low, pi - 1);**

**randomizedQuickSort(arr, pi + 1, high);**

**}**

**}**

**// Function to print array**

**void printArray(const vector<int>& arr) {**

**for(int x : arr)**

**cout << x << " ";**

**cout << endl;**

**}**

**int main() {**

**int n;**

**cout << "Enter the number of elements: ";**

**cin >> n;**

**vector<int> arr(n);**

**cout << "Enter the elements:\n";**

**for(int i = 0; i < n; i++) cin >> arr[i];**

**// Copy array for deterministic and randomized sorting**

**vector<int> arrDeterministic = arr;**

**vector<int> arrRandomized = arr;**

**srand(time(0)); // Seed random number generator**

**// Deterministic QuickSort**

**clock\_t start = clock();**

**deterministicQuickSort(arrDeterministic, 0, n-1);**

**clock\_t end = clock();**

**double detTime = double(end - start) / CLOCKS\_PER\_SEC;**

**// Randomized QuickSort**

**start = clock();**

**randomizedQuickSort(arrRandomized, 0, n-1);**

**end = clock();**

**double randTime = double(end - start) / CLOCKS\_PER\_SEC;**

**cout << "\nSorted array (Deterministic QuickSort):\n";**

**printArray(arrDeterministic);**

**cout << "Comparisons: " << deterministicComparisons << ", Time: " << detTime << " seconds\n";**

**cout << "\nSorted array (Randomized QuickSort):\n";**

**printArray(arrRandomized);**

**cout << "Comparisons: " << randomizedComparisons << ", Time: " << randTime << " seconds\n";**

**return 0;**

**}**

**Output:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**