|  |
| --- |
| **// Number Generator** #include <bits/stdc++.h> #include <iostream>   using namespace std;   int main() {  srand(static\_cast<unsigned int>(time(0)));  const int numCount = 10000;    ofstream outFile("number.txt");  if (outFile.is\_open())  {  for (int i = 0; i < numCount; i++)  {  int randomNum = rand() % 1000;  outFile << randomNum <<endl;  }  outFile.close();  cout << "Random numbers written to number.txt." << endl;  }  else  {  cerr << "Unable to open file for writing." << endl;  return 1;  }    cout<<numCount<<endl;    return 0; }  **// Quick Sort**  #include <iostream> #include <vector> #include <ctime> #include <fstream> #include <chrono>   using namespace std;   vector<int> a ;   void Interchange(vector<int>& a, int i, int j) {  int p = a[i];  a[i] = a[j];  a[j] = p; }   int partition(vector<int>& a, int m, int p) {  int v = a[m]; // Pivot element  int i = m ; // Start from the next element  int j = p;    while (i<=j)  {  while (a[i] <= v)  { i++; }    while (a[j] > v)  { j--; }    if (i < j)  { Interchange(a, i, j); }  }  a[m] = a[j];  a[j] = v;    return j; }   void QuickSort(int p, int q) {  if (p < q)  {  int j = partition(a, p, q);  QuickSort(p, j - 1);  QuickSort(j + 1, q);  } }   int main() {  ifstream inFile("number.txt");  if (inFile.is\_open())  {  int num;  while (inFile >> num)  {  a.push\_back(num);  }  inFile.close();  }  else  {  cerr << "Unable to open file for reading." << endl;  return 1;  }    int n = a.size();  cout << "Vector Size: " << n << endl;    // Start timing  auto start = chrono::high\_resolution\_clock::now();    // Execute the function to measure  QuickSort(0, n - 1);    // End timing  auto End = chrono::high\_resolution\_clock::now();    // Calculate the duration  chrono::duration<double> duration = End - start;    // Output the time taken in seconds  cout << "Time taken: " << duration.count() << " seconds" << endl;    return 0; }     **// Merge Sort** #include <iostream> #include <fstream> #include <vector> #include <cmath> #include <chrono>   using namespace std;   vector<int> a; vector<int> b;   void Merge(int low, int mid, int high) {  int h = low;  int i = low;  int j = mid+1;    while((h<=mid) && (j<=high))  {  if(a[h]<=a[j])  {  b[i] = a[h];  h = h+1;  }  else  {  b[i] = a[j];  j = j+1;  }  i = i+1;  }    if(h>mid)  {  for(int k=j; k<=high; k++)  {  b[i]= a[k];  i = i+1;  }  }  else  {  for(int k=h; k<=mid; k++)  {  b[i] = a[k];  i = i+1;  }  }  for(int k=low; k<= high; k++)  {  a[k] = b[k];  } } void MergeSort(int low , int high) {  if(low<high)  {  int mid = floor( (low+high)/2 );    MergeSort(low,mid);  MergeSort(mid+1, high);    Merge(low, mid, high);  } }     int main() {  ifstream inFile("number.txt");  if (inFile.is\_open())  {  int num;  while (inFile >> num)  {  a.push\_back(num);  }  inFile.close();  }  else  {  cerr << "Unable to open file for reading." << endl;  return 1;  }    int n = a.size();  b.resize(n);  cout << "Vector Size: " << n << endl;    // Start timing  auto start = chrono::high\_resolution\_clock::now();    // Execute the function to measure  MergeSort(0, n - 1);    // End timing  auto End = chrono::high\_resolution\_clock::now();    // Calculate the duration in nanoseconds  chrono::duration<double> duration = End - start;    // Output the time taken in seconds  cout << "Time taken: " << duration.count() << " seconds" << endl;    return 0; }  **QuickSort VS MergeSort Time Complexity**: |