**Qumber Zaidi, Ali Abbas, Harith llangachcharige**

**Insertion sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

void InsertionSort(int A[], int n)

{

int i, value, index;

for (i = 1; i < n; i++) // i=1 : 1 step , i<n: n time (n-1 + 1), i++ : n-1 time

{

value = A[i]; // n-1 time

index = i; // n-1 time

while (index > 0 && A[index - 1] > value) // best case: two steps , worst case i(index) +1 times

{

A[index] = A[index - 1]; // best case: 0 , worst case: i times

index = index - 1; // best case: 0 , worst case: i times

}

A[index] = value; //n-1 times

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a3;

double totalTime = 0;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 50; j++)

{

int size = n[i];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int k = 0; k < n[i]; k++)

{

a3[i] = rand() % n[i];

}

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

InsertionSort(a3, size);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

totalTime += time\_span3.count();

std::cout << "It took me " << time\_span3.count() << " seconds." << endl;

}

cout << "Average time for " << n[i] << " size: " << totalTime / 50 << endl;

totalTime = 0;

delete[] a3;

}

//closes the for loop(incomplete)

return 0;

}

**Selection sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

void swap(int\* xp, int\* yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void SelectionSort(int a[], int n) {

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

int large = 0;

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

if (a[large] < a[j])

large = j;

swap(a[large], a[n - i]);

}

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a3;

double totalTime = 0;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 50; j++)

{

int size = n[i];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int k = 0; k < n[i]; k++)

{

a3[i] = rand() % n[i];

}

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

SelectionSort(a3, size);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

totalTime += time\_span3.count();

std::cout << "It took me " << time\_span3.count() << " seconds." << endl;

}

cout << "Average time for " << n[i] << " size: " << totalTime / 50 << endl;

totalTime = 0;

delete[] a3;

}

//closes the for loop(incomplete)

return 0;

}

**BubbleSort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

void swap(int\* xp, int\* yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void BubbleSort(int a[], int n) {

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

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

if (a[j] > a[j + 1])

swap(a[j], a[j + 1]);

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a3;

double totalTime = 0;

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 50; j++)

//for loop for Task two 50 times (incomplete)

{

int size = n[i];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int k = 0; k < n[i]; k++)

{

a3[i] = rand() % n[i];

}

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

BubbleSort(a3, size);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

totalTime += time\_span3.count();

std::cout << "It took me " << time\_span3.count() << " seconds." << endl;

}

cout << "Average time for " << n[i] << "Size: " << totalTime / 50 << endl;

totalTime = 0;

delete[] a3;

}

//closes the for loop(incomplete)

return 0;

}

**Merge Sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

void Merge(int a[], int begin, int mid, int end) {

int i = begin;

int j = mid + 1;

int\* b = new int[end - begin + 1];

int k = 0;

while (i <= mid && j <= end) {

if (a[i] <= a[j]) {

b[k] = a[i];

i++;

k++;

}

else

{

b[k] = a[j];

j++;

k++;

}

}

while (i <= mid) {

b[k] = a[i];

k++;

i++;

}

while (j <= end) {

b[k] = a[j];

k++;

j++;

}

for (i = begin, k = 0; i <= end; i++, k++) {

a[i] = b[k];

}

}

void Mergesort(int a[], int begin, int end) {

if (begin < end) {

int mid = (begin + end) / 2;

Mergesort(a, begin, mid);

Mergesort(a, mid + 1, end);

Merge(a, begin, mid, end);

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a3;

double totalTime = 0;

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 50; j++)

//for loop for Task two 50 times (incomplete)

{

int size = n[i];

a3 = new int[size];

for (int k = 0; k < n[i]; k++)

{

a3[i] = rand() % n[i];

}

/\*for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}\*/

//cout << endl;

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

Mergesort(a3, 0, size - 1);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

totalTime += time\_span3.count();

std::cout << "It took me " << time\_span3.count() << " seconds." << endl;

/\* cout << endl;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}\*/

//cout << endl;

}

cout << "Average time for " << n[i] << "Size: " << totalTime / 50 << endl;

totalTime = 0;

delete[] a3;

}

//closes the for loop(incomplete)

return 0;

}

**Quick Sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

int partition(int A[], int begin, int end) {

int pivot = A[end];

int i = begin - 1;

for (int j = begin; j < end; j++)

if (A[j] < pivot) {

i++;

swap(A[i], A[j]);

}

swap(A[i + 1], A[end]);

return i + 1;

}

void QuickSort(int A[], int begin, int end) {

if (begin < end) {

int q = partition(A, begin, end);

QuickSort(A, begin, q - 1);

QuickSort(A, q + 1, end);

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a3;

double totalTime = 0;

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 50; j++)

{

int size = n[i];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int k = 0; k < n[i]; k++)

{

a3[i] = rand() % n[i];

}

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

QuickSort(a3, 0, size - 1);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

totalTime += time\_span3.count();

std::cout << "It took me " << time\_span3.count() << " seconds." << endl;

}

cout << "Average time for " << n[i] << " size: " << totalTime / 50 << endl;

totalTime = 0;

delete[] a3;

}

return 0;

}