

DSA Lab-04 Tasks

Name: Syed Muhammad Raza Ali

Enrolment: 02-134231-028

Course: DSA Lab

Faculty: Miss Rabia

|  |
| --- |
| **Lab 4: Algorithm Analysis** |

Objective(s): Upon completion of this lab session, students will be able to:

|  |
| --- |
| Study the experimental ways to analyze the time complexity of different algorithms. |

Exercise 1: Library Books sorting

You've been assigned to evaluate sorting algorithms for organizing a cluttered stack of computer science books in the library based on their titles alphabetically. Your task is to compare insertion sort and selection sort in terms of their efficiency in arranging the books. Write a C++ program to implement these algorithms and assess their performance. Recommend the most suitable algorithm for streamlining book organization in the library, ensuring a tidy and efficient arrangement for patrons to locate books easily.

Code:

Using insertion sort

#include <iostream>

#include <string>

#include<math.h>

#include<chrono>

using namespace std;

int main() {

string arr[10] = { "IICT","CP","DSA","CALCULUS","MVC","OOP","PHYSICS","DLD","COAL","STATS" };

clock\_t c\_start, c\_end;

c\_start = clock();

string key;

int j;

for (int i = 0;i < 10;i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

cout << "[";

for (int i = 0;i < 10;i++) {

cout << arr[i] << " ";

}

cout << "]";

c\_end = clock();

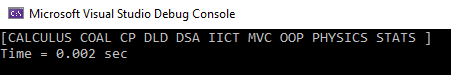
double time;

time = (double)(c\_end - c\_start) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime = " << time << " sec " << endl;

return 0;

}



Using selection sort

#include <iostream>

#include <string>

#include<math.h>

#include<chrono>

using namespace std;

int main() {

string arr[10] = { "IICT","CP","DSA","CALCULUS","MVC","OOP","PHYSICS","DLD","COAL","STATS" };

clock\_t c\_start, c\_end;

c\_start = clock();

string temp;

for (int i = 0; i < 10; i++) {

for (int j = i + 1; j < 10; j++) {

if (arr[j] > arr[i]) {

temp = arr[j];

arr[j] = arr[i];

arr[i] = temp;

}

}

}

cout << "[";

for (int i = 0;i < 10;i++) {

cout << arr[i] << " ";

}

cout << "]";

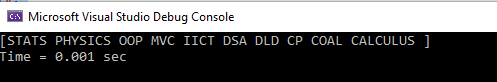
c\_end = clock();

double time;

time = (double)(c\_end - c\_start) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime = " << time << " sec " << endl;

return 0;}



Conclusion: Selection sort is taking less time so it is more efficient

Exercise 2: Ball in the box

At a toy manufacturing company, you need to develop a search algorithm to find a specific colored ball in a large box having 10 different colored balls. Write a C++ program to implement and compare linear and binary search, where you aim to determine the most efficient method, analyze their time complexity and runtime performance, and recommend the best approach for quickly locating the desired ball, optimizing customer service.

Code:

Using Linear search

#include <iostream>

#include <string>

#include<math.h>

#include<chrono>

using namespace std;

int main() {

string arr[10] = {"red","green","blue","orange","yellow","pink","white","black","purple","brown"};

cout << "Enter the color you want to search : ";

string colorName;

cin >> colorName;

clock\_t c\_start, c\_end;

c\_start = clock();

string temp;

bool flag;

for (int i = 0;i < 10;i++) {

if (arr[i] == colorName) {

cout << "Color Found at index no : " << i;

flag = true;

}

}

if (!flag) {

cout << "Color doesnt exist";

}

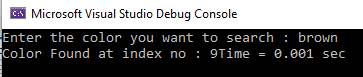
c\_end = clock();

double time;

time = (double)(c\_end - c\_start) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime = " << time << " sec " << endl;

return 0;

}

Using Binary Search

#include <iostream>

#include <string>

#include<math.h>

#include<chrono>

using namespace std;

int main() {

string arr[10] = {"red","green","blue","orange","yellow","pink","white","black","purple","brown"};

string temp;

for (int i = 0; i < 10; i++) {

for (int j = i + 1; j < 10; j++) {

if (arr[j] < arr[i]) {

temp = arr[j];

arr[j] = arr[i];

arr[i] = temp;

}

}

}

cout << "[";

for (int i = 0;i < 10;i++) {

cout << arr[i] << " ";

}

cout << "]";

cout << "\nEnter the color you want to search : ";

string colorName;

cin >> colorName;

clock\_t c\_start, c\_end;

c\_start = clock();

int beg = 0;

int end = 9;

int mid = (beg + end) / 2;

while (beg <= end && arr[mid] != colorName) {

if (colorName < arr[mid])

end = mid - 1;

else

beg = mid + 1;

mid = (beg + end) / 2;

}

bool flag = false;

if (arr[mid] == colorName)

flag = true;

if (flag)

cout << "Value found after divisions : " << mid;

else

cout << "Value doesn't exist";

c\_end = clock();

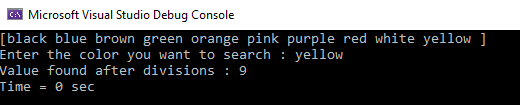
double time;

time = (double)(c\_end - c\_start) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime = " << time << " sec " << endl;

return 0;

}



Conclusion: Binary search is taking less time so it is more efficient

Exercise 3

Consider the following array. Write a program to determine the time taken to search following elements using linear search and binary

Item=44

Item=400

Item=450

{2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,100,102,104,106,108,110,112,114,116,118,120,122,124,126,128,130,132,134,136,138,140,142,144,146,148,150,152,154,156,158,160,162,164,166,168,169,170,172,174,176,178,180,182,184,186,188,190,192,194,196,198,200,202,204,206,208,210,212,214,216,218,220,222,224,226,228,230,232,234,236,23238,240,242,244,246,248,250,252,254,256,258,260,262,264,266,268,270,272,274,276,278,280,282,284,286,288,300,302,304,306,308,310,312,314,316,318,320,322,324,326,328,330,332,334,336,338,340,342,344,346,348,350,352,354,356,358,360,362,364,366,368,370,372,374,376,378,380,382,384,386,388,390,392,394,396,398,400}

Code:

Using Linear Search

#include <iostream>

#include <chrono>

using namespace std;

int main() {

int arr[193] = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 169, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 23238, 240, 242, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372, 374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400 };

bool flag = false;

clock\_t c\_start, c\_end;

c\_start = clock();

for (int i = 0; i < 193; i++) {

if (arr[i] == 44) {

cout << "44 found at index no : " << i << endl;

}

}

for (int i = 0; i < 193; i++) {

if (arr[i] == 400) {

cout << "400 found at index no : " << i << endl;

}

}

for (int i = 0; i < 193; i++) {

if (arr[i] == 450) {

flag = true;

cout << "450 found at index no : " << i << endl;

}

}

if (!flag)

cout << "450 doesn't exist in the array...";

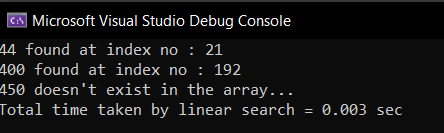
c\_end = clock();

double time;

time = (double)(c\_end - c\_start) / (double)CLOCKS\_PER\_SEC;

cout << "\nTotal time taken by linear search = " << time << " sec " << endl;

return 0;

}

Using Binary Search

#include <iostream>

#include <chrono>

using namespace std;

int main() {

int arr[193] = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 169, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 23238, 240, 242, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372, 374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400 };

int beg = 0;

int end = 192;

int mid = (beg + end) / 2;

clock\_t c\_start, c\_end;

c\_start = clock();

while (beg <= end && arr[mid] != 44) {

if (44 < arr[mid])

end = mid - 1;

else

beg = mid + 1;

mid = (beg + end) / 2;

}

if (arr[mid] == 44)

cout << "Element found at index no : " << mid;

else

cout << "Element doesn't exist...";

beg = 0;

end = 192;

mid = (beg + end) / 2;

while (beg <= end && arr[mid] != 400) {

if (400 < arr[mid])

end = mid - 1;

else

beg = mid + 1;

mid = (beg + end) / 2;

}

if (arr[mid] == 400)

cout << "\nElement found at index no : " << mid;

else

cout << "Element doesn't exist...";

beg = 0;

end = 192;

mid = (beg + end) / 2;

while (beg <= end && arr[mid] != 450) {

if (450 < arr[mid])

end = mid - 1;

else

beg = mid + 1;

mid = (beg + end) / 2;

}

if (arr[mid] == 450)

cout << "\nElement found at index no : " << mid;

else

cout << "\nElement doesn't exist...";

c\_end = clock();

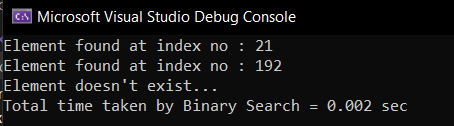
double time;

time = (double)(c\_end - c\_start) / (double)CLOCKS\_PER\_SEC;

cout << "\nTotal time taken by Binary Search = " << time << " sec " << endl;

return 0;

}



Exercise 4

Write programs for following two different algorithms for finding that given number is prime or not. Also determine the time taken by both algorithms

|  |  |
| --- | --- |
| Frist Algorithm | Second Algorithm |
| for i 🡨 2 to n-1  if i divides n  n is not prime number | for i 🡨 2 to √n  if i divides n  n is not prime number |

.

Calculate times taken by these programs for given values and conclude which algorithm is better than other

i. n = 11

ii. n = 101

iii. n = 1111

iv. n = 1000003

v. n =

Plot graphs for time of execution vs. “n”, for all the values of n given .Use excel for plotting graph.

Code:

#include <iostream>

#include <chrono>

using namespace std;

int main() {

int n1 = 11;

int n2 = 101;

int n3 = 1111;

int n4 = 1000003;

int n5 = 10000000019;

clock\_t c\_start1, c\_end1;

c\_start1 = clock();

bool isPrime1 = true;

for (int i = 2; i < n1; i++) {

if (n1 % i == 0)

isPrime1 = false;

}

if (isPrime1)

cout << n1 << " is a prime number";

c\_end1 = clock();

double time1 = (double)(c\_end1 - c\_start1) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime for 11 = " << time1 << " sec " << endl;

clock\_t c\_start2, c\_end2;

c\_start2 = clock();

bool isPrime2 = true;

for (int i = 2; i < n2; i++) {

if (n2 % i == 0)

isPrime2 = false;

}

if (isPrime2)

cout << n2 << " is a prime number";

c\_end2 = clock();

double time2 = (double)(c\_end2 - c\_start2) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime for 101 = " << time2 << " sec " << endl;

clock\_t c\_start3, c\_end3;

c\_start3 = clock();

bool isPrime3 = true;

for (int i = 3; i < n3; i++) {

if (n3 % i == 0)

isPrime3 = false;

}

if (isPrime3)

cout << n3 << " is a prime number";

else

cout << n3 << " is not a prime number";

c\_end3 = clock();

double time3 = (double)(c\_end3 - c\_start3) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime for 1111 = " << time3 << " sec " << endl;

clock\_t c\_start4, c\_end4;

c\_start4 = clock();

bool isPrime4 = true;

for (int i = 4; i < n4; i++) {

if (n4 % i == 0)

isPrime4 = false;

}

if (isPrime4)

cout << n4 << " is a prime number";

c\_end4 = clock();

double time4 = (double)(c\_end4 - c\_start4) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime for 1000003 = " << time4 << " sec " << endl;

clock\_t c\_start5, c\_end5;

c\_start5 = clock();

bool isPrime5 = true;

for (int i = 5; i < n5; i++) {

if (n5 % i == 0)

isPrime5 = false;

}

if (isPrime5)

cout << n5 << " is a prime number";

else

cout << n5 << " is not a prime number";

c\_end5 = clock();

double time5 = (double)(c\_end5 - c\_start5) / (double)CLOCKS\_PER\_SEC;

cout << "\nTime for 10000000019 = " << time5 << " sec " << endl;

return 0;

}

