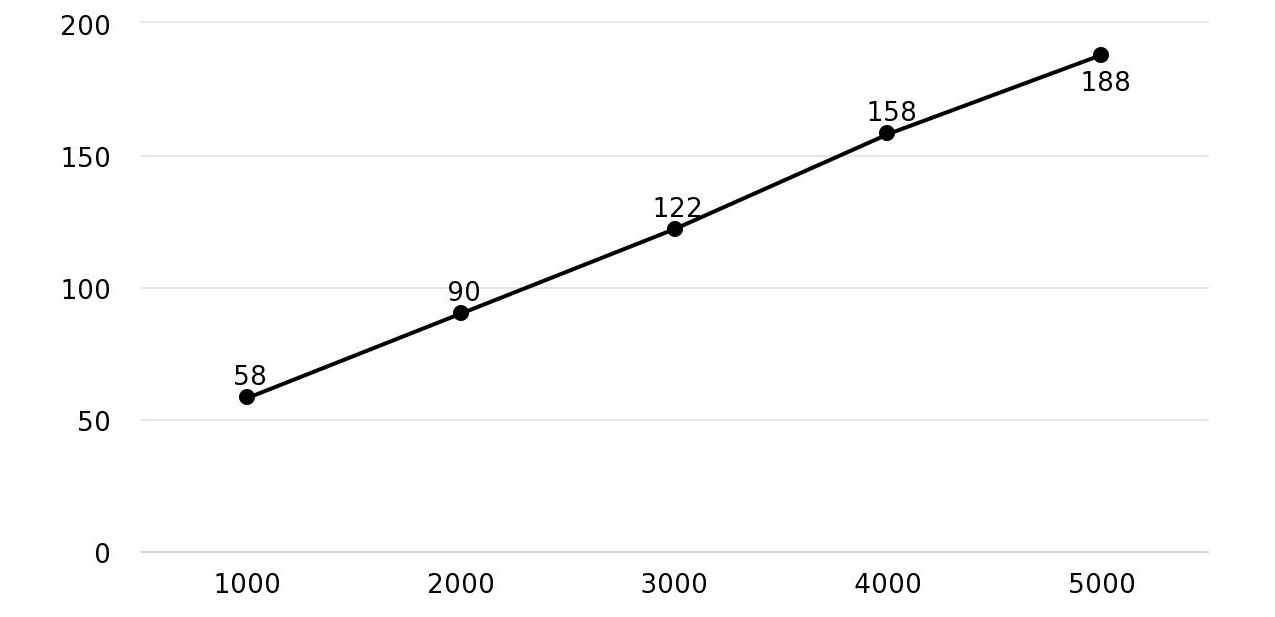
|  |  |
| --- | --- |
| n | time |
| 1000 | 58 |
| 2000 | 90 |
| 3000 | 122 |
| 4000 | 158 |
| 5000 | 188 |

Linear Search (Time Complexity: O(n))



Graph

Table

// Linear Search Program

#include <iostream>

#include <cstdlib>

#include <time.h>

using namespace std;

int main() {

clock\_t start, end;

double time;

int n;

cout<<"Enter the number of elements: "; cin>>n;

cout<<"\nArray Elements:-\n";

int arr[n];

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

arr[i] = (rand() % (n\*10)) + 1;

cout<<arr[i]<<endl;

}

int element;

cout<<"Enter the element to be searched: ";

cin>>element;

start = clock();

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

if(arr[i] == element){

end = clock();

cout<<"Element "<<element<<" found at position "<<i+1<<endl;

time = ((double)(end - start))/CLOCKS\_PER\_SEC \* 1000;

cout<<"LINEAR SEARCH TIME = "<<time;

}

}

return 0;

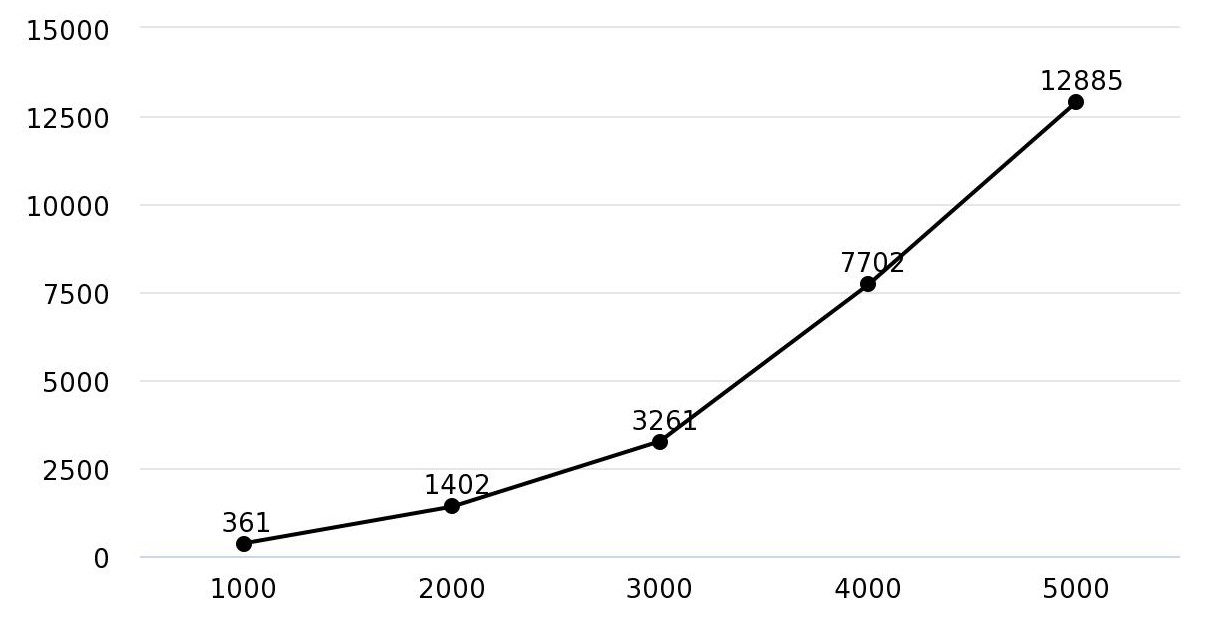
}

|  |  |
| --- | --- |
| n | time |
| 1000 | 361 |
| 2000 | 1402 |
| 3000 | 3261 |
| 4000 | 7702 |
| 5000 | 12885 |

Selection Sort (Time Complexity: O(n2))

Graph

Table

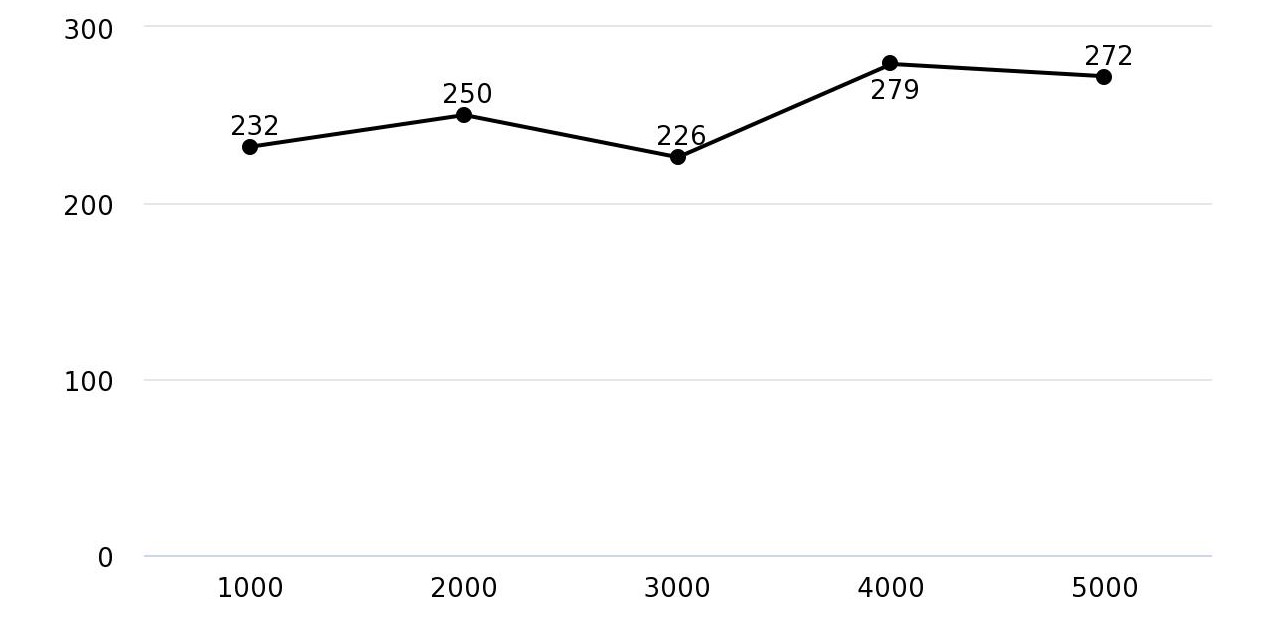


|  |  |
| --- | --- |
| n | time |
| 1000 | 232 |
| 2000 | 250 |
| 3000 | 226 |
| 4000 | 279 |
| 5000 | 272 |

Binary Search (Time Complexity: O(log n))

Graph

Table



// Binary Search Program

#include <iostream>

#include <cstdlib>

using namespace std;

clock\_t start1, end1, start2, end2;

double time1, time2;

void SelectionSort(int \*arr, int n){

int i, j, temp;

start1 = clock();

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

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

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

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

end1 = clock();

time1 = ((double)(end1 - start1))/CLOCKS\_PER\_SEC \* 1000;

cout<<"SORTING TIME = "<<time1;

}

int main() {

int n;

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

cin>>n;

cout<<"\nArray Elements:-\n";

int arr[n];

int first, mid, last;

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

arr[i] = (rand() % (n\*10)) + 1;

cout<<arr[i]<<endl;

}

SelectionSort(arr,n);

int element;

cout<<"\nEnter the element to be searched: ";

cin>>element;

first = 0; last = n-1;

start2 = clock();

while(first <= last){

mid = (first+last)/2;

if(arr[mid] == element){

cout<<"Element "<<element<<" found at position "<<mid+1<<endl;

end2 = clock();

time2 = ((double)(end2 - start2))/CLOCKS\_PER\_SEC \* 1000;

cout<<"BINARY SEARCH TIME = "<<time2;

break;

}

else if(arr[mid] > element){

last = mid - 1;

}

else{

first = mid + 1;

}

}

return 0;

}