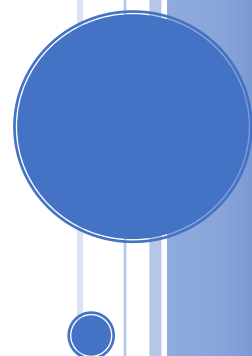




KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

REPORT

PERSONAL HOMEWORK 1



**UNIVERSITY OF SCIENCE
VIETNAM NATIONAL UNIVERSITY
HO CHI MINH CITY
INFORMATION TECHNOLOGY**

Lecturers:

- **Châu Thành Đức**
- **Phan Thị Phương Uyên**
- **Ngô Đình Hy**
- **Phan Thị Phương**

Student:

- **Nguyễn Đức Huy -19127422**

Progress:

- **Finish all the task 1 and 2 of Personal Homework 1**

CONTENT

1	Research	1
1.1	Explain Search Algorithms	1
1.2	Real Life Problem	2
2	Programming	3
2.1	Exercise 1: Winning Lottery Ticket	3
2.2	Exercise 2: Harvesting Orchard	4
3	Reference:.....	4

PERSONAL HOMEWORK 1:

SEARCH ALGORITHM

1 Research

1.1 Explain Search Algorithms

3 search algorithms that I know is absolutely **Sequential Search**, **Binary Search** and **Sentinel Search**.

Sequential Search:

A sequential search is when you look and check each piece of data of the list, one by one, and don't stop until you find what you are looking for target value. You can use a sequential search on any data, whether it is sorted or unsorted (imagine that is very low for you to search a list of data already ordered and if you looking for the max element of the list you, you must looking for all elements of the list, it takes a lot of time and comparison). However, sequential search is the only option for you to use when you need to search through data that is unsorted.

Given a list with n elements input, the best case is when the value is equal to first element of the list then you stop looking for any element and stop progressing, in which case is just use 1 comparison.

The worst case is when the value is not in the list or the value is in the last of the list, you must looking each element of the list, in which case it n comparison needed.

Binary Search:

An algorithm that tells us how to efficiently find a specific value in an ordered (sorted) list. It is called 'binary' search because each time you look at a value in the list you divide the list into 2 parts, one is discarded and the other is kept. The word "binary" here just means something that has two parts, such as a binary star system (made of two stars); binary search shouldn't be confused with binary numbers.

At each time you divide list of two part and use comparison that the key is looking for which part and then you will find this key in suitable part until you found or you not found the key in the list.

This Algorithms is very fast than sequential search because is divide 2 part at each time, and half elements of the list is gone you don't need check half of the list because elements of the list is ordered

The best case is when we luckily the key you want to find is equal to the middle element of the list and then you done

The worst case is the key you looking for isn't in the list or is in the end of the list you must take **logn** time to divide and conquer

Sentinel Search:

Sentinel Search is similar to Sequential search. It checks each element in the list until it find the element you looking for, but the different point is you set the flag value equal to the key you looking for in the end of the list and you don't need use comparison to break out the loop, because it must be found even the element you looking for isn't in the list, then after loop you just check condition if the position return result isn't equal to last element that you set above and that position is the right key you looking for, if not the key you looking for isn't in the list

In computer programming, a sentinel value (also referred to as a flag value, trip value, rogue value, signal value, or dummy data) is a special value in the context of an algorithm which uses its presence as a condition of termination, typically in a loop or recursive algorithm. The sentinel value is a form of in-band data that makes it possible to detect the end of the data when no out-of-band data (such as an explicit size indication) is provided. The value should be selected in such a way that it is guaranteed to be distinct from all legal data values, since otherwise the presence of such values would prematurely signal the end of the data.

1.2 Real Life Problem

Algorithms are really useful, in programming and real life. Here, I'll show you an example of how I used one of those algorithms to make my real task easier.

The real task is if you have a dictionary, and you have the one word you looking for meaning of this word but you know the word of dictionary is variety and large. You absolutely not looking for each page to find your meaning word. If you looking for each page you must take a lot of time to do this task.

The solution is very simple, you should use Binary method to solve this problem. And you know, I think most of dictionary is already ordered by letter in alphabet (A,B,C,D,E...,Z) and it is very useful to use binary method. If the word you looking for behind or front the middle alphabet you just divide a sub list alphabet (I mean a sub list alphabet you dive previous) then you divide and conquer until you found this word easily with few step. And you know the interesting is not the dictionary ordered traditionally you think. In second or so on letter of any word are already ordered. For example, if you want to looking for meaning of '**Interesting**' word, if you find the first page of letter I, then a lot page for letter I, what

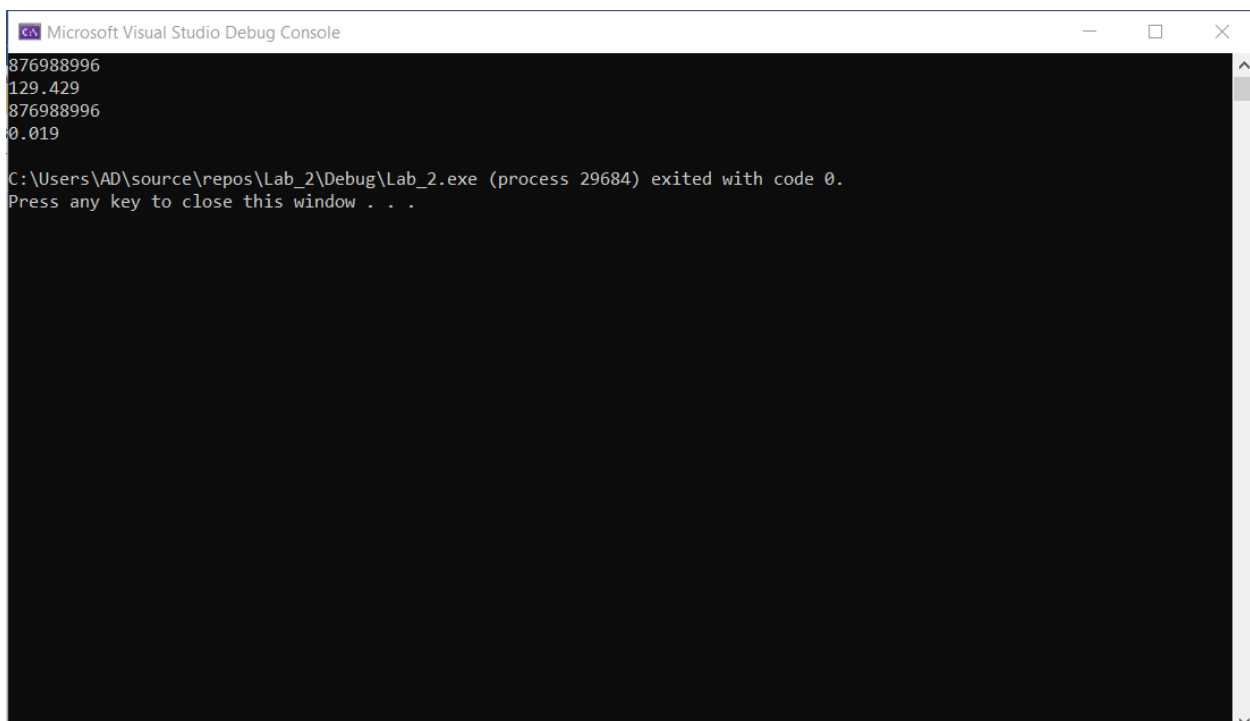
should you do, you know second letter is n and you just use this method for a second letter (maybe third fourth so on until you find meaning of this word).

2 Programming

2.1 Exercise 1: Winning Lottery Ticket

This problem is very simple to use sequential search. The solution is you just implement a function (bool function) that check number is a square number or not. Another function you need implement is that calculate sum of digit of number(function **sum_of_digit** return sum of digit from any unsigned number). And then you just use 1 loop from A to B, and check 2 condition from each loop for each number, if this number at runtime qualified all condition and greater than max qualified number previous and you just update max number digit again until break loop.

But as far as I concern this method above is not efficiently for large range. Can we optimize this method. To come up with my new idea is from the range A to B you should probably to find small range by square root A and B you find smaller than range A to B. And then from the new range you just square two of each element between new range A and B but it still qualified condition. For example, A=30 and B=90 you square root of A and B and round it, new range will 5 to 9 right now, but you must check after round again because $5^2 < 30$ (A). And instead of searching from 30 to 90 that is 61 elements checked you just check from 6 to 9 it just 4 elements checked (it fast than traditional method right).



```
Microsoft Visual Studio Debug Console
876988996
129.429
876988996
0.019

C:\Users\AD\source\repos\Lab_2\Debug\Lab_2.exe (process 29684) exited with code 0.
Press any key to close this window . . .
```

Imagine if number in the range is very large, the new method is very useful than traditional method. I check each method and you can see above.

The input I give is $A=1$ and $B=100000000000$. Two method return same result but the method use square root is fast than first method, it second method save a lot of time and I choose it for this problem.

2.2 Exercise 2: Harvesting Orchard

The problem is very simple. First you need implement function that return fruits by the day (function **find_fruits_by_day** return fruits by the day you havest), you should use a level plus method in mathematic for this function. And then you use binary search from day 1 to day N, if fruits of middle day less than fruits T you just divide day on the right half and so on until if found if not find it on the left half.

3 Reference:

https://en.wikipedia.org/wiki/Binary_search_algorithm

<https://medium.com/smelly-code/binging-binary-search-aa172b5b31c2>