SAMUEL GINN
COLLEGE OF ENGINEERING

Problem Solving, Patterns, and Search

Search problems

Entire companies and economies are based on search problems.

Most of us solve a ***search problem*** of some kind or another every day.

Do I have a specific song on my playlist?
What's the shortest driving route from my house to Las Vegas?
Where is the cheapest place in town to buy gas today?
What's a good movie to stream tonight?
Where are my keys?

Search algorithms

*"In computer science, a **search algorithm** is an algorithm for finding an item with specified properties among a collection of items."*

Search algorithm. (n.d.) In Wikipedia.

All search algorithms find **a solution** to the problem by examining **candidate solutions** in the **search space** and evaluating them based on some **matching criteria**.

A solution:	Could be only one solution or multiple solutions.
Search space:	The collection of all things that could possibly be solutions.
Candidate solution:	One item in the search space.
Matching criteria:	A property or condition that qualifies a candidate solution as a solution to the problem.

A central question for this course will be how to structure the search space to make this process efficient.

List search

Search space: A list of items that can be compared to a target item.

An abstract description of how the search space is structured. [more on this later …]

The only constraint placed on the data. [more on this later …]

Goal: Locate an item with a specific value or discover that it is not there.

In the case of duplicates, use the location of the first one found.

Approach: (1) Develop a strategy (2) Develop an algorithm that implements that strategy (3) Develop a program that implements that algorithm.

Strategy: Linear scan -> Linear Search

Starting at one end of the list, examine each element one by one and compare it to the target value. Stop when the target value is found or when all elements have been examined.

Algorithm:

1. Start at the first position in the list.
2. Compare the current element with the target value. Go to next element if no match.
3. Repeat step 2 until the target is found or until all elements of the list have been examined.

A solution

Strategy: **Linear Search**

Starting at one end of the list, examine each element one by one and compare it to the target value. Stop when the target value is found or when all elements have been examined.

Data structure: How can we represent the list of elements?

Let's choose an **array**, and let's assume that the data are all integer values.

```
public int search(int[] a, int target) {




}
```

Algorithm:

1. Start at the first position in the list.
2. Compare the current element with the target value. Go to next element if no match.
3. Repeat step 2 until the target is found or until all elements of the list have been examined.

A solution

Strategy: **Linear Search**

Starting at one end of the list, examine each element one by one and compare it to the target value. Stop when the target value is found or when all elements have been examined.

Data structure: How can we represent the list of elements?

Let's choose an **array**, and let's assume that the data are all integer values.

```
public int search(int[] a, int target) {
  int i = 0;
  while ((i < a.length) && (a[i] != target))
     i++;
  if (i < a.length)
     return i;
  else
     return -1;
}
```

Algorithm:

1. Start at the first position in the list.
2. Compare the current element with the target value. Go to next element if no match.
3. Repeat step 2 until the target is found or until all elements of the list have been examined.

A solution

Strategy: **Linear Search**

Starting at one end of the list, examine each element one by one and compare it to the target value. Stop when the target value is found or when all elements have been examined.

Data structure: How can we represent the list of elements?

Let's choose an **array**, and let's assume that the data are all integer values.

```
public int search(int[] a, int target) {
    for (int i = 0; i < a.length; i++) {
       if (a[i] == target)
          return i;
    }
    return -1;
}
```

Algorithm:

1. Start at the first position in the list.
2. Compare the current element with the target value. Go to next element if no match.
3. Repeat step 2 until the target is found or until all elements of the list have been examined.

Aspects of a solution

Correctness

Verification: Did we solve the problem correctly?

Validation: Did we solve the correct problem?

Generality

Reusability: Can we apply our solution in different contexts?

Efficiency

Running time: Does our solution run fast enough?

Memory: Does our solution use an appropriate amount of memory?