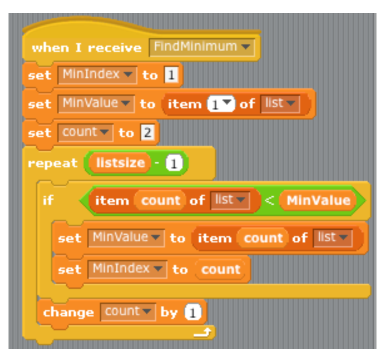
COMP 102: Excursions in Computer Science
Lecture 10: Searching

Instructor: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~jpineau/comp102

Quick recap of searching and sorting

- Recall our example last week about finding the minimum.
- I argued that this could be done much faster if the list was sorted first.
- Then I taught you how to sort lists.
- Now let's talk about searching.



Searching example

- **Given**: A list of names of students and their favourite colour.
- **Problem**: Find the favourite colour of the student named Alice, if she is in the class.

1. Bob 'black'
2. Mary 'red'
3. Carol 'yellow'
4. Allison 'blue'
5. Alice 'yellow'
6. Joe 'green'
7. Joseph 'purple'

Sequential Search

- Process each list entry from first to last.
  - Check if each entry processed is the entry for "Alice".
  - If we find the "Alice" entry,
    - Note Alice's favourite colour.
    - Stop searching.
- How many entries in the list are processed before Alice is found?

1. Bob 'black'
2. Mary 'red'
3. Carol 'yellow'
4. Allison 'blue'
5. Alice 'yellow'
6. Joe 'green'
7. Joseph 'purple'

Sequential search

- How many entries in the list are processed before Alice is found?

1. Bob 'black'
2. Mary 'red'
3. Carol 'yellow'
4. Allison 'blue'
5. George 'green'
6. Billy 'white'
7. Walter 'yellow'
8. Geoffrey 'pink'
9. Alice 'yellow'
10. Joe 'green'
11. Joseph 'purple'

Sequential Search on a Sorted List

- Can you speed-up the search if the list is sorted?

Yes! If you are looking for Alice.

What if you are looking for Joe's favourite colour?

Sorting won't help. Or can it?

1. Alice 'yellow'
2. Allison 'blue'
3. Bob 'black'
4. Carol 'yellow'
5. Joe 'green'
6. Joseph 'purple'
7. Mary 'red'

Binary Search

- Search algorithm for sorted lists.
- How do you find a word in the dictionary?

  E.g. “Joe”

Binary Search on a Dictionary

- Look at the **middle** page of the dictionary.
  - Read the words on this page.
- If the word you are looking for comes **after** these words:
  - Search among the pages of the dictionary that come after this page.
- If the word you are looking for comes **before** these words:
  - Search among the pages of the dictionary that come before.
- If the word you are looking for is **on** this page,
  - Stop searching!

Binary Search for “Joe”

1. Alice ‘yellow’
2. Allison ‘blue’
3. Bob ‘black’
4. Carol ‘yellow’ ← First, try the middle.
5. Joe ‘green’ ← Third try, got it!
6. Joseph ‘purple’ ← Second, try the middle of the second half.
7. Mary ‘red’

Comparing Search Algorithms

- Sequential search: 5 items examined to find “Joe”.
- Binary search: 3 items examined to find “Joe”.

- Which would choose?

Binary Search for "Walter"

1. Alice 'yellow'
2. Allison 'blue'
3. Billy 'white'
4. Bob 'black'
5. Carol 'yellow'
6. Geoffrey 'pink' ← 1st try
7. George 'green'
8. Joe 'green'
9. Joseph 'purple' ← 2nd try
10. Mary 'red' ← 3rd try
11. Walter 'yellow' ← 4th try

Comparing Search Algorithms

- Searching for "Joe":
  - Sequential search: 5 items examined.
  - Binary search: 3 items examined.
- Searching for "Walter"
  - Sequential search: 11 items examined.
  - Binary search: 4 items examined.
- Which would choose?

Worst-Case Analysis

- Binary search seems faster than sequential search for sorted lists.
- Let's think about the maximum possible number of items we need to check.
  - With sequential search: ***N* elements**

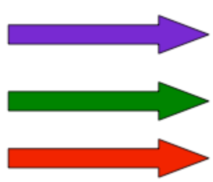
    where ***N*** = numbers of items in the sorted list.

    " If there are 7 elements in the list, then in the worst-case, sequential search looks at 7 elements before finding the answer. "
  - With binary search: ???

Worst-case Complexity of Binary Search

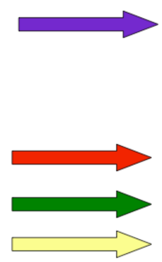
- Here, at most **3** elements of the list need to be analyzed.

1. Alice 'yellow'
2. Allison 'blue'
3. Bob 'black'
4. Carol 'yellow'
5. Joe 'green'
6. Joseph 'purple'
7. Mary 'red'



Worst-case Complexity of Binary Search

- Here, at most **4** elements of the list need to be analyzed.



1. Alice 'yellow'
2. Allison 'blue'
3. Billy 'white'
4. Bob 'black'
5. Carol 'yellow'
6. Geoffrey 'pink'
7. George 'green'
8. Joe 'green'
9. Joseph 'purple'
10. Mary 'red'
11. Walter 'yellow'

Why should you care?

- If your database has 8,388,607 names (e.g. the telephone book), using **sequential search** may examine all 8,388,607 names.
- To search a list of 8,388,607 names using **binary search** examines how many names at most?

23

How do we get this?

If you have 1 names in the list, need at most 1 check.

If you have 2 names in the list, need at most 2 checks.

If you have 4 names in the list, need at most 3 checks.

If you have 8 names in the list, need at most 4 checks.

If you have 16 names in the list, need at most 5 checks.

…..

If you have N names in the list, need at most $log_2(N)+1$ checks.

But!

- Binary search only works on **sorted lists**.
- In the worst-case, if you sort using Bubble sort, you will need:

$n*(n-1)$ comparisons for Bubble sort + $log_2(n)$ comparisons for Binary search

- In the worst-case, if you sort using Merge sort, you will need:

$n*log_2(n)$ comparisons for Merge sort + $log_2(n)$ comparisons for Binary search

- So why not keep things simple and use:

n comparisons for Sequential search (no sorting necessary) ?

Binary search vs Sequential search

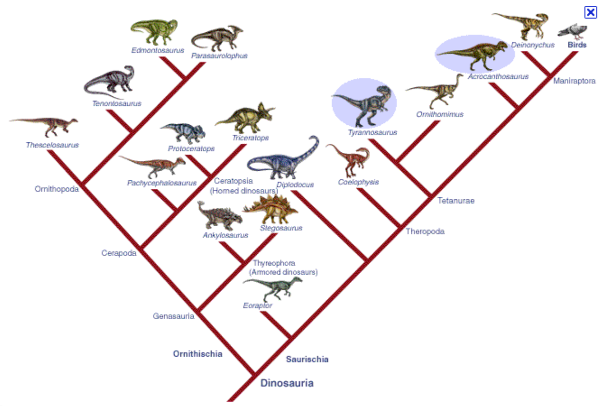
- In general, you need to sort only once, and then you can search the sorted list as many times as we want.
- If you don't need to do multiple searches, then it is better to just do sequential search, without any pre-sorting.

Quick Recap

- Searching is as useful as (if not more than) sorting.
- So far we have seen searching on arrays (sorted or not).
- This is interesting, but the fun is only beginning!
- In many problems, data is not stored in an array.

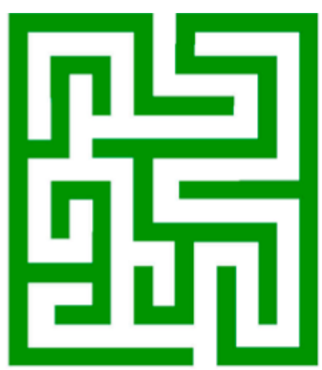
Searching data organized in a tree

- You excavated a fossil, and are trying to identify its species.
  - In what order do you consider the nodes in the tree?



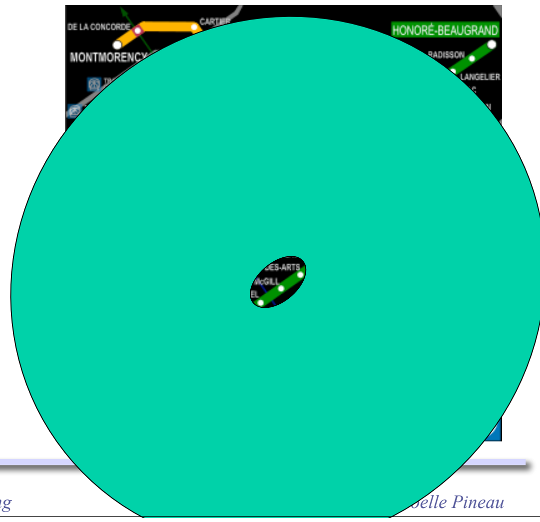
Searching through a maze

- Interesting questions:
  - How do we search through the maze?
  - How do we encode this problem for the computer?



Searching through the subway system

What is the shortest path from the Université de Montréal station to the McGill station?



How should we encode this problem?

Can't store the list of stations in a simple array.

This is an example of a **graph**.

Graphs

A **graph** is an abstract representation defined by a pair **(N, E)**, where

N is a collection of nodes (or objects)

E is a collection of pairs of nodes, called edges (representing the relations between the objects.)

In the Montreal metro system:

- What are the nodes?

  The metro stations.

- What are the edges?

  Rail link between neighbouring stations.

Paths

- A **path** is a sequence of adjacent nodes.
  E.g. "McGill" - "Place-des-Arts" - "St-Laurent" - "Berri-UQAM"

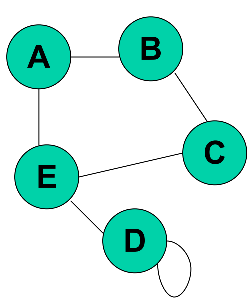
- The **path length** is the total number of nodes along a path.

- We can store a graph in memory using an **adjacency matrix**, which defines which nodes are next to each other.

Adjacency matrix

- Consider a 2-D matrix, showing the relation between any pair of nodes (1=neighbours, 0=not neighbours).

Example



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	0
C	0	1	0	0	1
D	0	0	0	1	1
E	1	0	1	1	0

Downtown Montreal map

- Nodes?
- Edges?
- Path?

Interesting questions on graphs

Question #1: What is the shortest path between two given (non-neighbour) nodes?

Question #2: What is the best path to visit <u>all</u> nodes with minimum overall travel time?

Question #3: What is the overall topology of the graph?

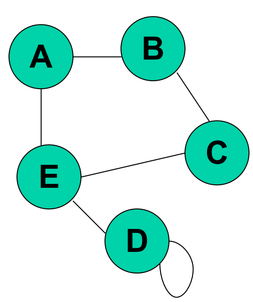
Many more interesting questions!

A few more definitions

- A **directed graph**, is a graph where there may be an edge from A to B, but not from B to A. So we say there is a **direction** to each edge.
- In **undirected graphs**, each connected pairs of nodes is **connected in both directions**.
- A **cycle** is a path in which the **first and last nodes are the same**.
- A **tree** is a graph that has **no cycle**.

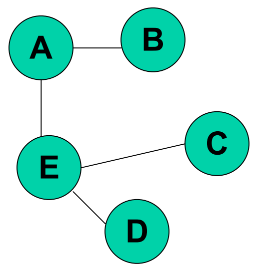
Example of a Cycle

- Nodes A-B-C-E form a cycle.
- Node D forms a cycle.



Example of a Tree

- The following graph is a tree.



Example of an Undirected Graph

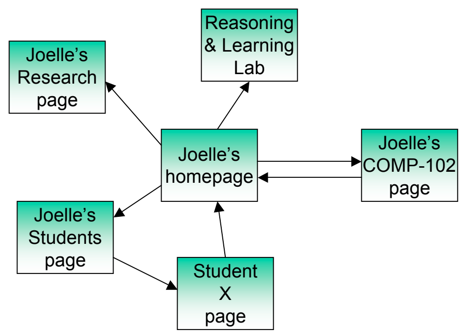
If A is a neighbour of B,

then B is a neighbour of A

(and similarly for all nodes.)

Example of a Directed Graph

- The internet!
  - Nodes are the web-pages.
  - Edges are the hyper-links, taking you from one page to another.



Take-home message

- Searching is one of the most useful algorithms.
- You should understand sequential search and binary, and be familiar with the pros/cons of each.
- Be able to recognize graphs, and define the key components (nodes, edges, paths, etc.)
- **Midterm**: Oct.18, 11:30am, in class