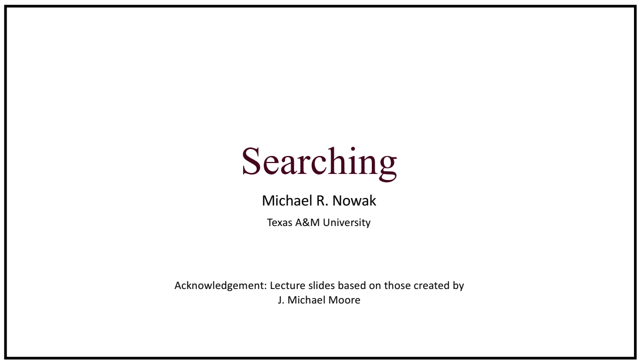
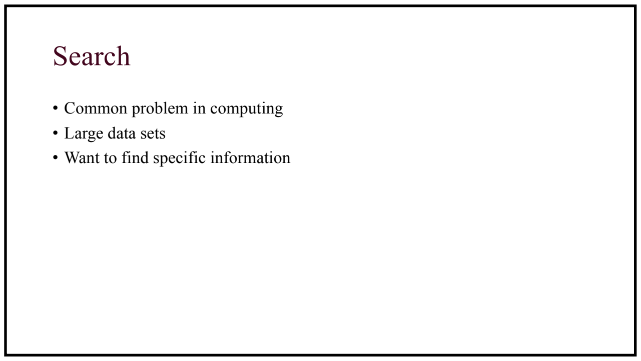


Searching

Michael R. Nowak

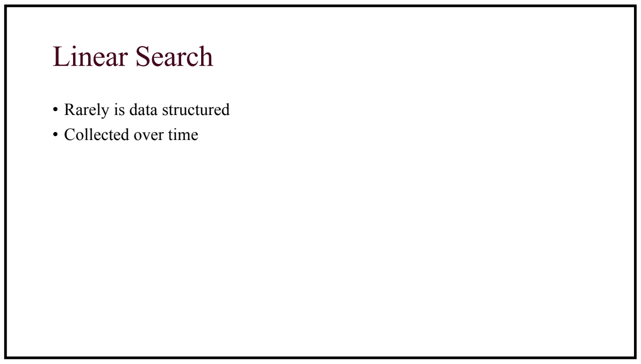
Texas A&M University

Acknowledgement: Lecture slides based on those created by
J. Michael Moore



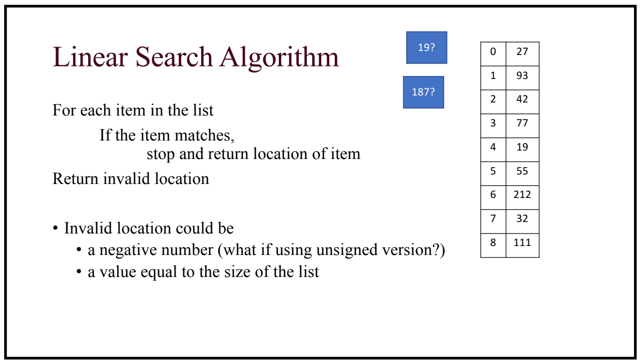
Search

- Common problem in computing
- Large data sets
- Want to find specific information



Linear Search

- Rarely is data structured
- Collected over time



Linear Search Algorithm

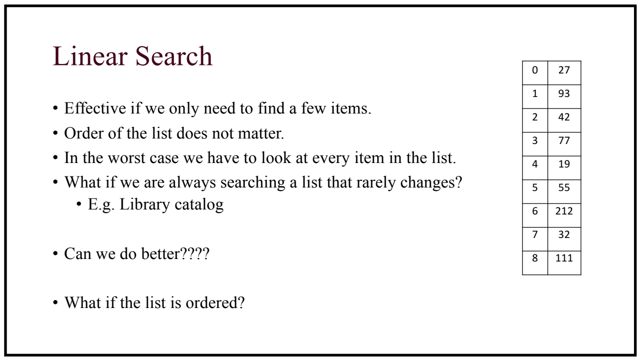
19?

187?

```
For each item in the list
    If the item matches,
        stop and return location of item
Return invalid location
```

- Invalid location could be
  - a negative number (what if using unsigned version?)
  - a value equal to the size of the list

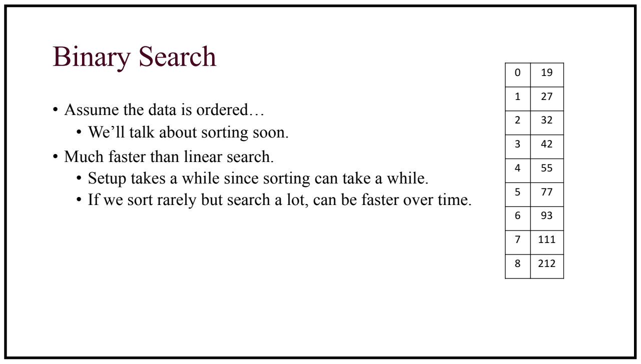
0	27
1	93
2	42
3	77
4	19
5	55
6	212
7	32
8	111



Linear Search

- Effective if we only need to find a few items.
- Order of the list does not matter.
- In the worst case we have to look at every item in the list.
- What if we are always searching a list that rarely changes?
  - E.g. Library catalog
- Can we do better????
- What if the list is ordered?

0	27
1	93
2	42
3	77
4	19
5	55
6	212
7	32
8	111



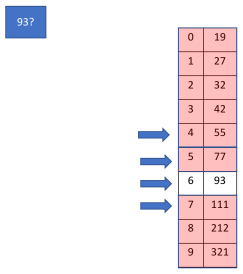
Binary Search

- Assume the data is ordered…
  - We'll talk about sorting soon.
- Much faster than linear search.
  - Setup takes a while since sorting can take a while.
  - If we sort rarely but search a lot, can be faster over time.

0	19
1	27
2	32
3	42
4	55
5	77
6	93
7	111
8	212

Binary Search

- Look at middle element
  - Matches? We are done!
  - Less than? Look before
  - Greater than? Look after
- Divide and Conquer Algorithm



0	19
1	27
2	32
3	42
4	55
5	77
6	93
7	111
8	212
9	321

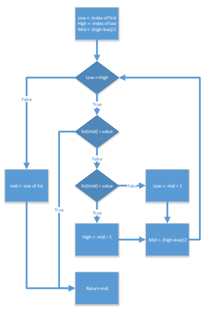
Binary Search Algorithm

```
1. low <- index of first
2. high <- index of last
3. while low <= high
   a. mid = (high + low)/2;
   b. if list[mid] = value
            return mid
   c. else if list[mid] < value (not at mid or below)
            low <- mid + 1 (low to next higher)
   d. else (no check but list[mid] > value – not at mid or above)
            high <- mid – 1 (high to next lower)
4. return invalid position (size of list)
```

0	19
1	27
2	32
3	42
4	55
5	77
6	93
7	111
8	212

Binary Search Algorithm

- With a flowchart



Sort???

- Use built in sort

```
#include <algorithm>
/* ... */
vector<int> vec;
/* ... */
sort(vec.begin(), vec.end());
/* ... */
```

- Or write your own… (Coming soon)

Linear vs. Binary Search

- Value more clear when you have lots of values
- Suppose we have integers ordered from smallest to largest.

Number of Integers	Linear Search Worst case number of elements examined	Binary Search Worst case number of elements examined
1,000,000,000	1,000,000,000	30
1,000,000,000,000,000,000,000 ($1X10^{21}$)	$1X10^{21}$	70
$1X10^{50}$	$1X10^{50}$	166