ECSE-322

18 January 2008

Lecture 7

Searching and Sorting

Wed 1035 ✓ 0100

0835 ✓ 3

W. 0935 3

1635 ✓

M 1635-1825

W 1035-1125

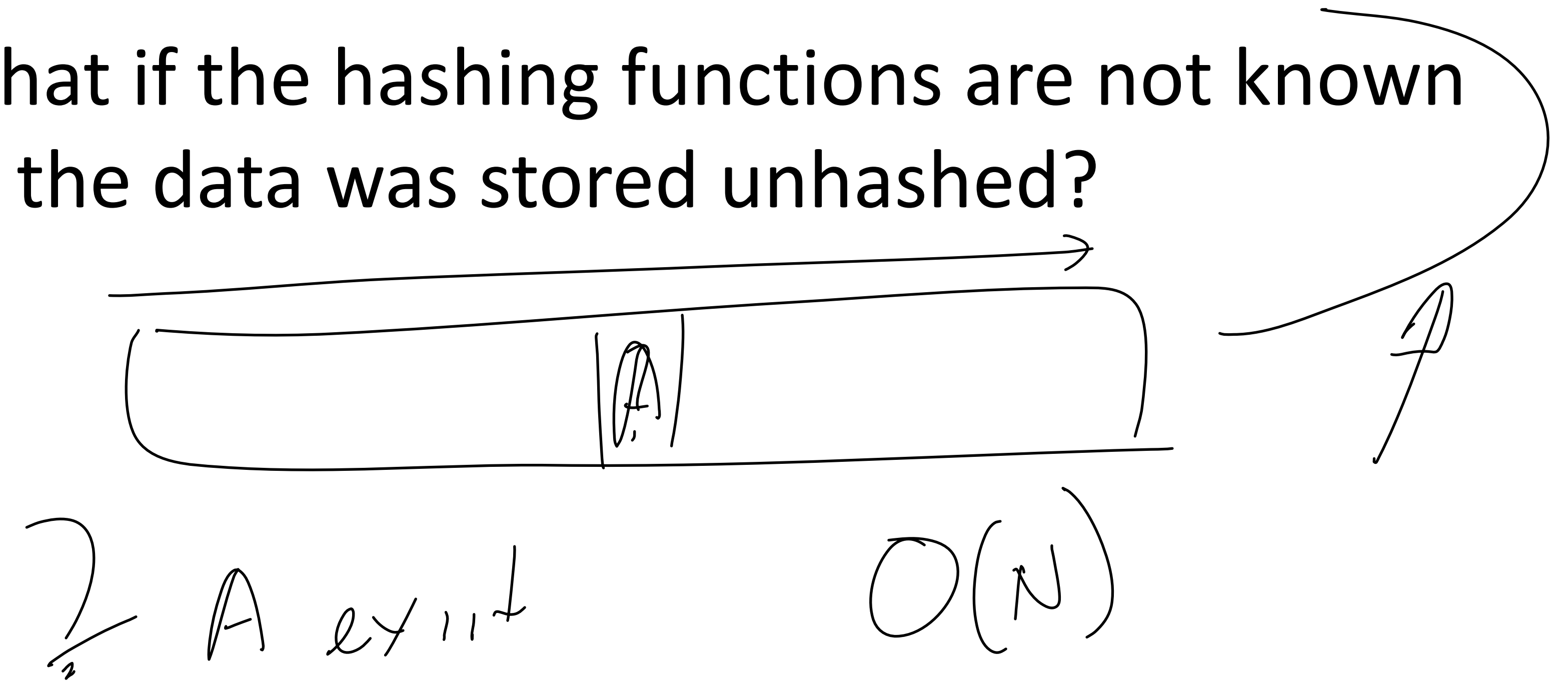
1635-1725

F 0835-1025

HASHING ?

Searching

- Data is only stored so that it can be found.
- In a hashing system, data is retrieved by following the hashing function.
- What if the hashing functions are not known or the data was stored unhashed?



Searching

- Find a data item given a list of data items..
- The data might be
  - Totally unordered ✓✓
  - Sequentially ordered ✓
- Problem
  - Determine where a key (or number), *l*, occurs in a linear array of items *r*.

Searching

- The number is known to be between

  r_l and r_u

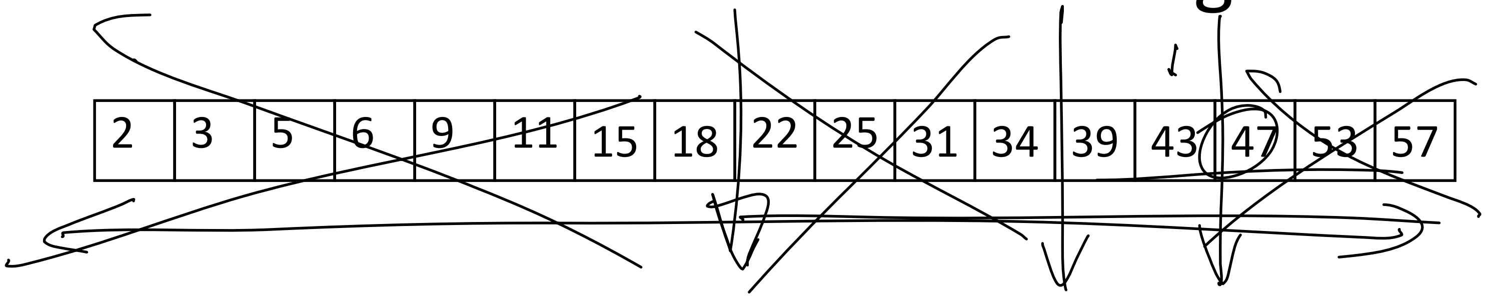
  – A value is to be returned such that
    - S=m if there exists m such that r_m = I, (l<=m<=u)
    - S=0 if item does not exist in range
- An obvious approach
  – Start at the lower bound and increment the location until the item is found or the upper bound is reached.

Searching

- On average - how much of the array will be searched to find a hit?
- This is *linear searching* and requires *O(N)* time.
- The system works well if the data is randomly ordered or we do not know of any structure.

Searching

- Assume data is stored in ascending order..



2	3	5	6	9	11	15	18	22	25	31	34	39	43	47	53	57

Question:

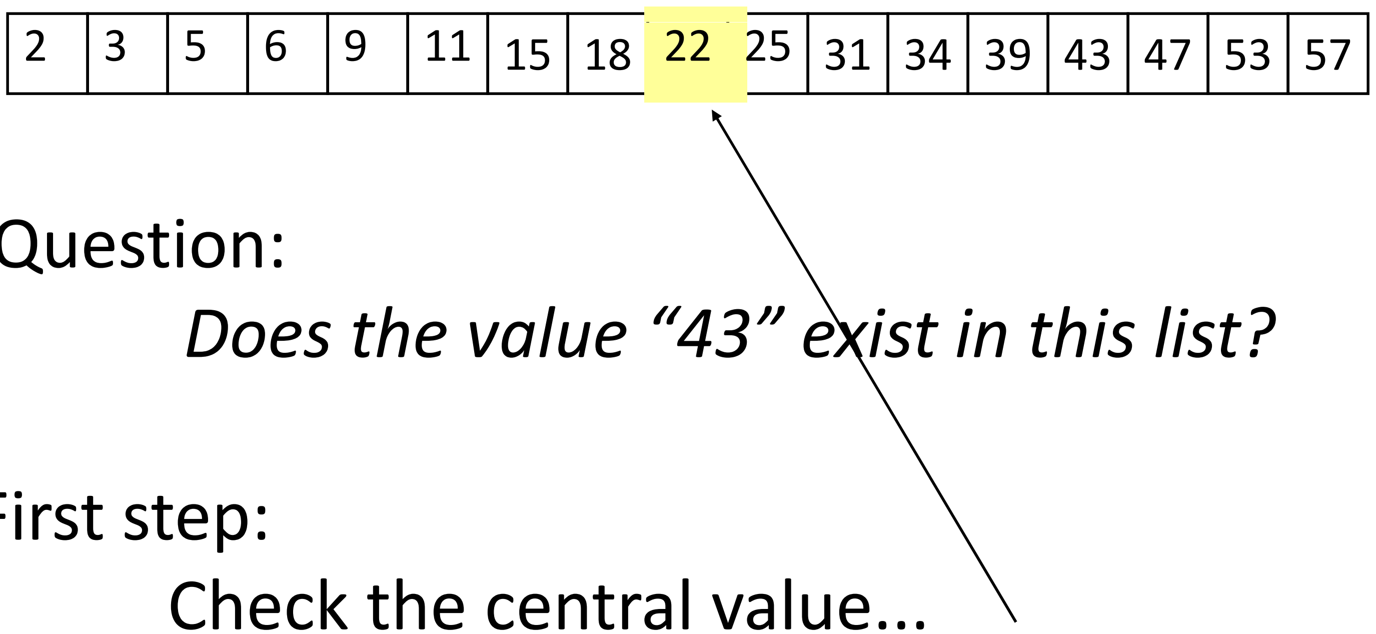
Does the value "43" exist in this list?

First step:

Check the central value...

Searching

- Assume data is stored in ascending order..



2	3	5	6	9	11	15	18	22	25	31	34	39	43	47	53	57

Question:

Does the value "43" exist in this list?

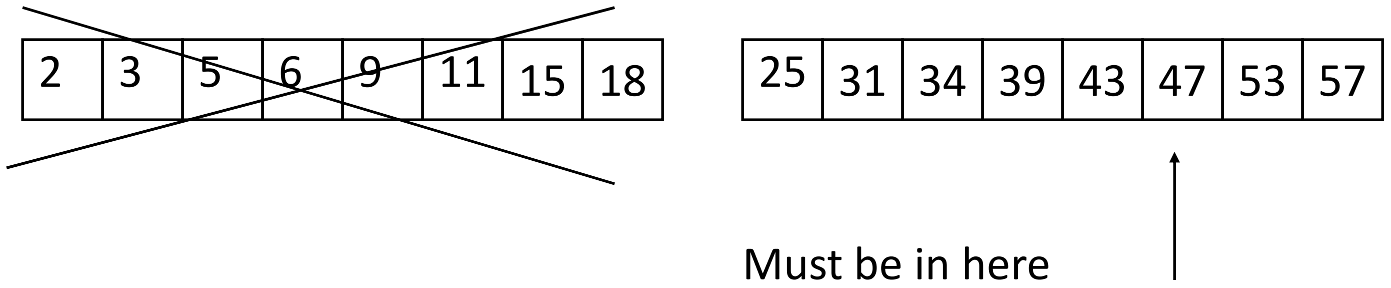
First step:

Check the central value...

Searching

If the central value 22 is not the one wanted...

The remaining list is in two pieces - all in the right half are greater than “22”. Since “43” is greater than “22”, it must be in the right half (if it exists in the list)



Must be in here

Searching

25	31	34	39	43	47	53	57

Second step..

Take the “center” value...

25	31	34	39	43	47	53	57

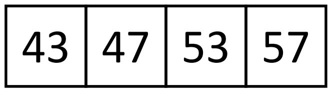
If this is not “43”, the list is split again...

25	31	34

43	47	53	57

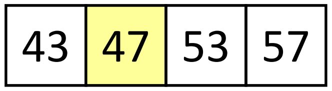
Since “43” is greater than the center value, take the right half...

Searching

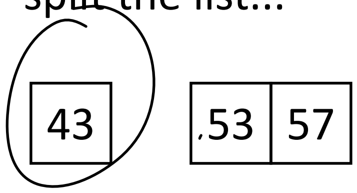


Third step…

Take the “center” value…

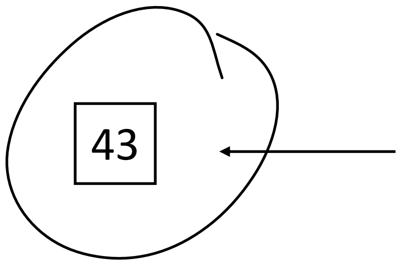


If it is not “43” split the list…



Fourth step...

“43” is smaller than the center value so look in the left list..



Success!

Searching

- The value was found in 3 subdivisions of the list and 4 compares!
- If “43” had not been in the list, the algorithm would have failed at the 4th compare and terminated.
- *Note that a linear search to find “43” would have taken 14 compares - linear searches are O(N).*

Searching

- In general
  - A list with 8 items will take a maximum of 3 tries..
  - A list with 16 items will take a maximum of 4 tries..
  - A list with 32 items will take a maximum of 5 tries..
  - ….

Searching - Complexity

- In general, for n items it will take t compares where t is given by:

$$t = \log_2(n)$$

or $n = 2^t$

1000

lin = 500 ←

bin = 10 ←

This process has a time complexity of?

$O(\log_2 N)$

Searching - Complexity

- In general, for n items it will take t compares where t is given by:

$$t = \log_2(n)$$

or $$n = 2^t$$

This process has a time complexity of...

$O(\log_2(N))$

The algorithm is also recursive.

Sorting

- So …
  - How did the data get into ascending order?
- It was sorted first!
- Two sorts
  - Exchange or Bubble Sort
  - Quick Sort
  - (There are also, insertion, selection, shell,…)

The Exchange Sort

- Extremely simple in concept:
    - Basically, take each pair of elements in the data set in turn. If the order of the pair is not ascending, exchange them....
    - Repeat the process until there remain no pairs to be re-arranged.

$$O(N^2) \qquad \text{'}N\text{'} \cdot \frac{N}{2}$$

The Exchange Sort

- Given a data set of N items there are N-1 steps for the first pass through.
- At the end of the first pass, the largest number will be at the right hand end.
- The next pass has N-2 steps…
- What is the total?

The Exchange Sort

- Given a data set of N items there are N-1 steps for the first pass through.
- At the end of the first pass, the largest number will be at the right hand end.
- The next pass has N-2 steps…
- What is the total?
    - Sum = (N-1) + (N-2) + (N-3) + …. + 0

      = N(N-1)/2

This is an O(N^2) process - but very simple to program

$N \log N$

The Cost of Finding Data

- What is the work to find S items in an array?
  - With an unordered set and a linear search

    $\rightarrow \boxed{O(SN)} \leftarrow \boxed{N^2}$

  - Exchange Sort plus binary search

    $\rightarrow \boxed{O(N^2)} + \boxed{O(S\log_2 N)}$

  - If S is approximately the same size as N both strategies yield $O(N^2)$ results.
  - Can we do better?

The Quick Sort

- Works on a divide and conquer strategy
- A member of a family of partition sorts
- Keep splitting the list into two lists and work on these independently

Recursion again

- There are several variants...

The Quick Sort

- Method with an extra array:
  - Take a randomly ordered array and choose one element.
  - Copy the array into temporary storage such that all those elements smaller than the chosen one are written from the left end to the middle, all those elements larger are written from the right end to the middle. At the end, copy the chosen element into the last space.

The Quick Sort

- The copy is an O(N) operation.
- After the copy the chosen element is in the right place and the array is partitioned into two pieces.
- The process can be repeated on each piece of the list leaving us with 4 pieces.
- Eventually, there will be only one item in each partition.

The Quick Sort

- This requires O($\log_2 N$) divisions....
- Overall, the time complexity is *O($N\log_2 N$)...*
- However, this process, as described, requires a second scratch array as big as the original..

- Try again...

The Quick Sort

- Take the original array and set index i to the left element, j to the right element
- Compare element i with element j. If j is greater than i, decrement j and repeat.
- If i is greater than j, exchange elements and increment i.
- Keep going until all elements have been considered -- the original left hand element will now be in its correct place and the array will be partitioned...