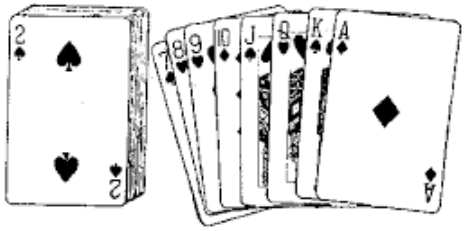
Searching Techniques

Objectives

- To understand how searching is performed using the intuitive linear search technique.
- To observe the limitations in linear search technique.
- To understand how halving the population enables efficient searching.
- To learn the Binary search technique.
- To observe the use of data decomposition in the two search techniques.

Eg. Pack of cards



- Take a pack of cards. Shuffle it thoroughly. Now find the Four of Spades.
  - ✓*How did you do it?*
- Put that card back and shuffle the pack again.
- Now find the Nine of Hearts.
  - ✓ *Did you look for the card the same way as the first time?*
- Suppose the card you were looking for was missing.
  - ✓*How would you have discovered this fact when searching?*
  - ✓*At what point in the search would you have known for certain that the card was not there?*

✓*Did you use a methodical way to find the card?*

Searching

- A *search algorithm* is a method for examining a group of data items in order to find an item with some particular property.

✓ *Most people would search for a card in a shuffled pack by starting at one end of the pack, and checking the cards one by one until they came across the card they were looking for.*

✓ *Suppose you were given a stack of 500 award certificates and told that one of the awards might be awarded to you.*

✓ *You would proceed as follows looking for your certificate:*

- Step 1 Check the name on the top certificate, if not your name then proceed to Step 2.
- Step 2 Check the name on the 2nd certificate, if not your name then proceed to Step 3.
- And …

✓ *Searching intuitively: Start at one end and search until …*

Linear search

- The most intuitive of all search algorithms is *linear search*.
- A linear search requires that the group of items be arranged one after another from first to last.
- The algorithm consists of examining the first item, then the second, the third, the fourth, and so on, until the desired item has been found.
- It involves lining up all the things you are searching through and checking them one at a time from one end to the other.

- If you find the thing you are looking for you can stop
  noting its position so you can go straight back to it.
- If you get to the end without finding it
  then you know for sure that it is not there at all.

✓*A simple and natural way to search…*

Linear search *contd*---

- Linear Search is so commonly used because it is simple and relatively quick provided the amount of things to search through is small.

- It is a natural way to search when
  - ✓*the things being searched through are in random order, and*
  - ✓*you do not expect to have to search for things very often.*

- *Does the fact that it is so common mean that*
  - *it is the only way to search for things, or*
  - *it is always the best search algorithm?*

Simple–Yes, *Efficient*?…

- ✓ *Computers store a lot of information, and they need to be able to sift through it quickly.*
- ✓ *One of the biggest search problems in the world is faced by Internet search engines, which must search billions of web pages in a fraction of a second.*
- ✓ *Computers can process information very quickly.*
- ✓ *You might think that to find something computers should just start at the beginning of their storage and keep looking until the desired information is found.*
  - ✓ *But this method is very slow—even for computers.*

Simple–Yes, *Efficient*? *contd*---

- ✓ *For example, suppose a supermarket has 10,000 different products on its shelves.*
  - When a bar code is scanned at a checkout, the computer must look through up to 10,000 numbers to find the product name and price.
  - Even if it takes only one thousandth of a second to check each code, ten seconds would be needed to go through the whole list.
    - ✓ *Imagine how long it would take to check out the groceries for a family!*

Search key

- The data that a computer is asked to look up, such as a word, a bar code number or an author's name, is called a *search key*.
  - ✓ *The key could be a letter someone is thinking of, a number in a list, a film star's web page, or a record in an employee database.*

- Given something to search for (*the 'key'*) a 'search algorithm' guarantees to find it… *if it is there*.

Is there a worst case scenario?

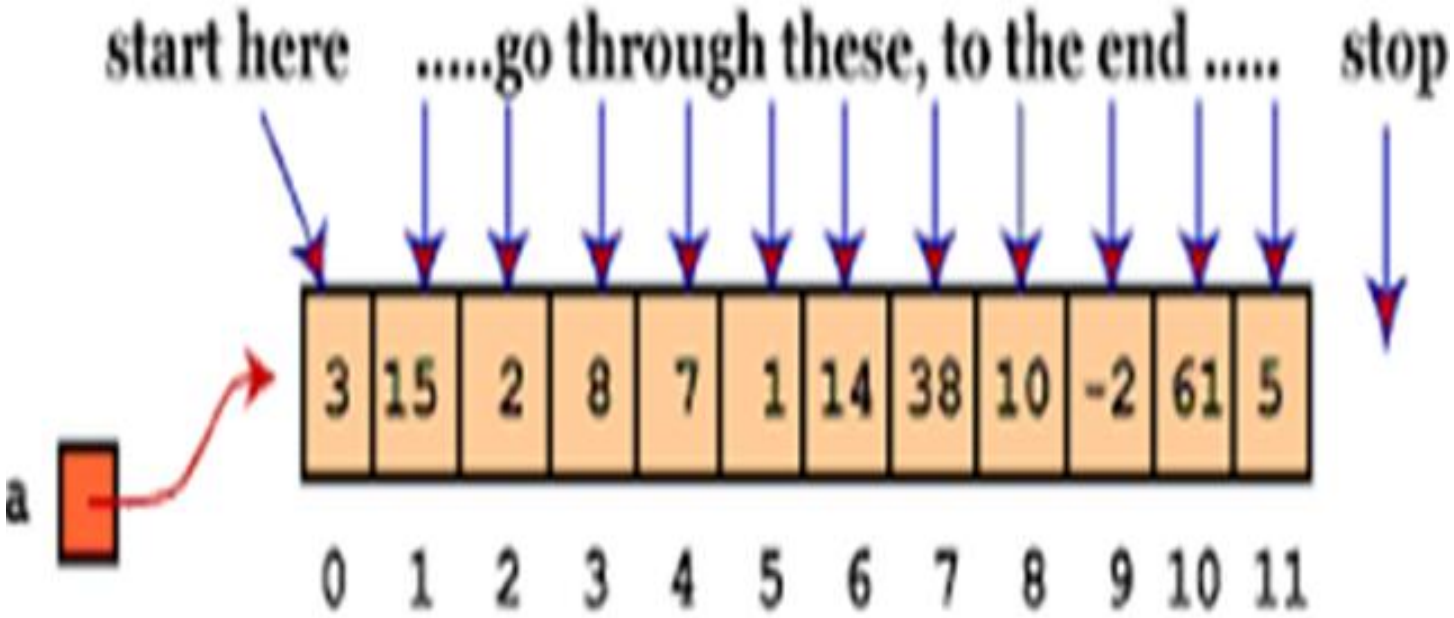
- Things certainly do not always seem to go as well as they should.

✓ *How often have you searched for something in one place after another, only to have the thing you were looking for in the very last place you could possibly have looked?*

- In its worst case you have to check everything.
- Eg. Find the search key '3' in {6,4,1,9,7,3,2,8}.

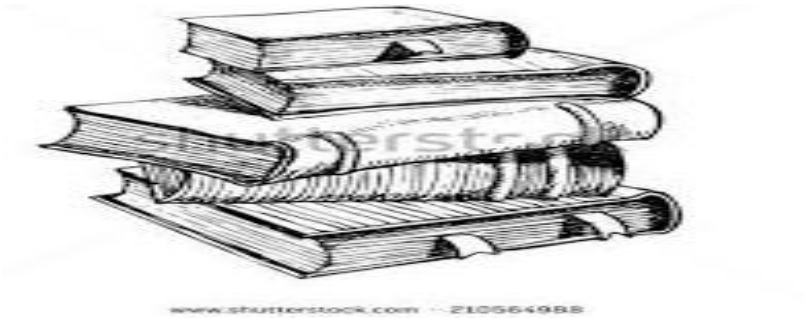
An example

- Eg: A search for '25' in {3,15,2,8,7,1,14,38,10,-2,61,5}.



Linear search through a pile of books

- ✓ *We are doing something over and over again – so we have some form of repetition.*
- ✓ *We probably do not know how many books there are to search through so it is not a counter controlled loop.*
- ✓ *We stop if we have found the thing we are looking for or we have run out of books to check.*
- ✓ *In other words we keep going while the current book is not the one we are after and while there are still books to look at.*



- ✓ *What is the repeated task?*
- ✓ *What is the loop test?*
- ✓ *Write the complete algorithm.*

A general set of instructions for linear search

To find a *thing* from a *series of things* **do the following**

1. Note that you have not found the *thing* yet.
2. Set the current *thing* to be the first *thing*.
3. **while** you have **not** found the *thing* you are searching for **and** you are **not** at the end of the *series of things*

do the following repeatedly

if the current *thing* is the *thing* you are searching for

then

you have found the *thing* you are looking for.

else

make the current *thing* the next *thing* in the sequence.

Eg. The I Spy game

- One person thinks of a word and tells the other players its first letter: "I spy with my little eye something beginning with T".

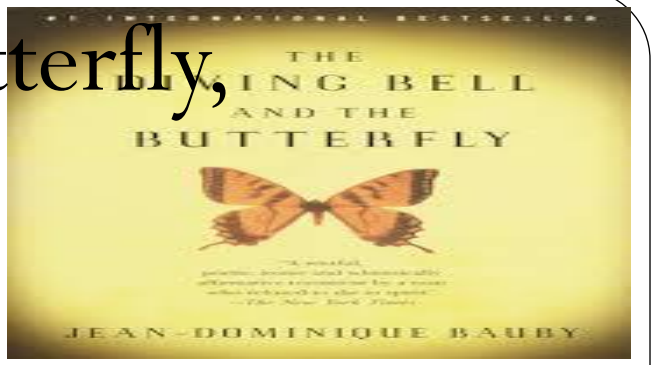
- The players then list the things they can see that start with the letter T: "*Toe*", "*Toy*", "*Television*", …

✓ *When a person names the thing the first person had thought of the search (and the game) stops.*

✓ *Otherwise the game continues until the players have tried every possibility they can think of.*

✓ *Do you think a linear search is performed here ?*

Eg. The Diving-bell and the Butterfly, (Bauby 1998)



- ✓ *It is the autobiography of Jean-Dominique Bauby, the former editor-in-chief of the French magazine Elle.*
- ✓ *He suffered a massive stroke and was left totally paralyzed, unable to move or speak, other than being able to move a single eyelid.*
- ✓ *Rather than give in to his disability, he instead decided to write his autobiography.*
- ✓ *In the book he describes what life is like for someone who is totally paralyzed, including how he communicated with people and so managed to write the book.*
- ✓ *He dictated it to a secretary, just using blinks.*
- ✓ *The secretary read through the alphabet a letter at a time.*
- ✓ *When she read out the letter that came next in the word Bauby wanted, he would blink, and she would write it down.*

The Diving-bell and the Butterfly *contd---*

- ✓ *She would then start at the beginning again looking for the next letter.*
- ✓ *Letter by letter, the whole book was written in this way.*
- ✓ *The algorithm was improved slightly in that the letters were ordered by frequency in French (Bauby's native language and that in which the book was written).*
  - ✓ *E is the most common letter so it was first in the list, then S, A, R and so on finishing with W, the least commonly used letter.*
- ✓ *All Bauby's communication to the outside world was done in this way.*

To find a letter being thought of – the Bauby way

- *Assume normal alphabetic order.*

1. Take a blank piece of paper.
2. Say the letter A.
3. **while**

   you have **not** written a letter down

   and

   you are **not** at the end of the alphabet

   do the following repeatedly

   if the person blinked

   then write down the last letter you said.

   else say the next letter of the alphabet.

✓*Do you think a linear search is performed here ?*

Eg. The game of 20 Questions

- This involves one person thinking of a famous person.
- The other player has to work out who it is just by asking *Yes* and *No* questions.
- You try to do it in as few questions as possible.
- If you take more than 20 questions you lose.
- *Is this a search problem?*

✓ *This is just a search problem*

✓ *we are searching for the name of a person out of the millions of famous people there are in the world.*

✓ *Can you play it the linear way?*

The game of 20 Questions the linear way

- We would ask a series of questions of the form "*Is it X*?" as with I-spy.
- For example a typical game might go:
  - "*Is it Nelson Mandella?*"
  - - "*No*"
  - "*Is it Arundhati Roy?*"
  - - "*No*"
  - "*Is it Freddy Mercury?*"
  - - "*No*“

✓*Is there any chances of winning the game?*

Asking the good questions

- Suppose I played the 20 questions with you.
- *What would I ask first?*

  A common first question is: "*Is the person male*?".

✓ Other questions that I might ask early on are:

- " *Is the person alive*?"
- " *Is the person fictional*?".

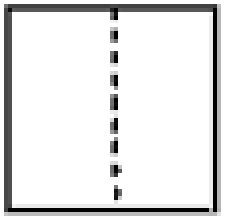
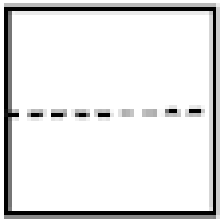
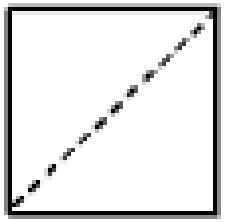
✓ *These questions in particular better than giving a series of names.*

✓ *Now, what are the chances of winning on that turn?*

Can we really manage to get the answer in less than 20 questions?

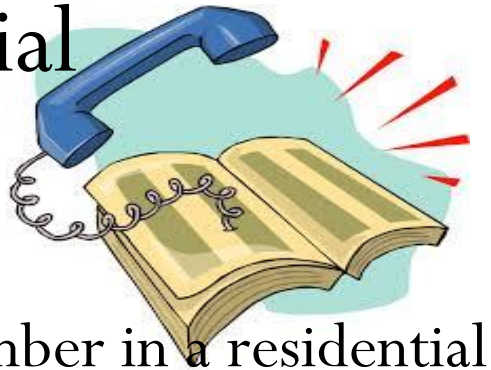
- ✓ *Ask the ideal question.*
  - ✓ *The ideal question is one that rules out half of all people if the answer is **yes,** and the other half if the answer is **no.***
  - ✓ *That means it does not matter to you what the answer is, you are equally close to the person's identity.*
  - ✓ *The more questions that divide the remaining possibilities in half that you come up with, the quicker you will get to the answer.*

It is all about halving the population

- ✓ *Halving the population repeatedly very quickly takes you to a single person.*
- ✓ *The skill of the game is of course to come up with good questions that do keep splitting the field in half.*
- ✓ *If a question does not split the field in half every time it could take more questions, and you no longer have that guarantee.*

Eg. Finding a number in a residential telephone directory



- Assume you want to find a friend's telephone number in a residential telephone directory.
  - *How do you do it? Did you use linear search?*

✓ *Did you start at the first page and checking every entry until you found your friend's?*

  ✓ *If you did, it probably took you a very long time (unless their name happens to be something like Aahann, Aammir or Aaronovitch – and it is very bad news if their name is Zwiebel, Zygovistinos or Zykun).*

✓ If the name was *Steinbeck*, for example, you would not start at the first page, but open the directory somewhere in *the middle*.

  ✓ *There is little point starting at the beginning, after all, as a name starting with S is more likely to be near the end.*

✓ However, it is possible that you have *over shot the page* you wanted.

A search for "*Steinbeck*"

- ✓ *Let us suppose I opened the book at McDonald.*
- It is before Steinbeck, *so I ignore all pages before that point*.
- I move to a page half way between *McDonald* and the end of the book.
- This time the first name on that page is *Tambe*, *so I overshot*.

- I now go half way between *McDonald* (*assuming I remembered to keep my finger in that place*) and *Tambe* finding *Poulter*

I overshot in the other direction this time.

- I go forwards again to a point halfway between *Poulter* and *Tambe* and find *Shah*.
- Not far enough, so I split between *Shah* and *Tambe* and find *Smith*.

Smith not far enough….

A search for "*Steinbeck*" *contd----*

- I split between *Smith* and *Tambe* and find *Stanton*.

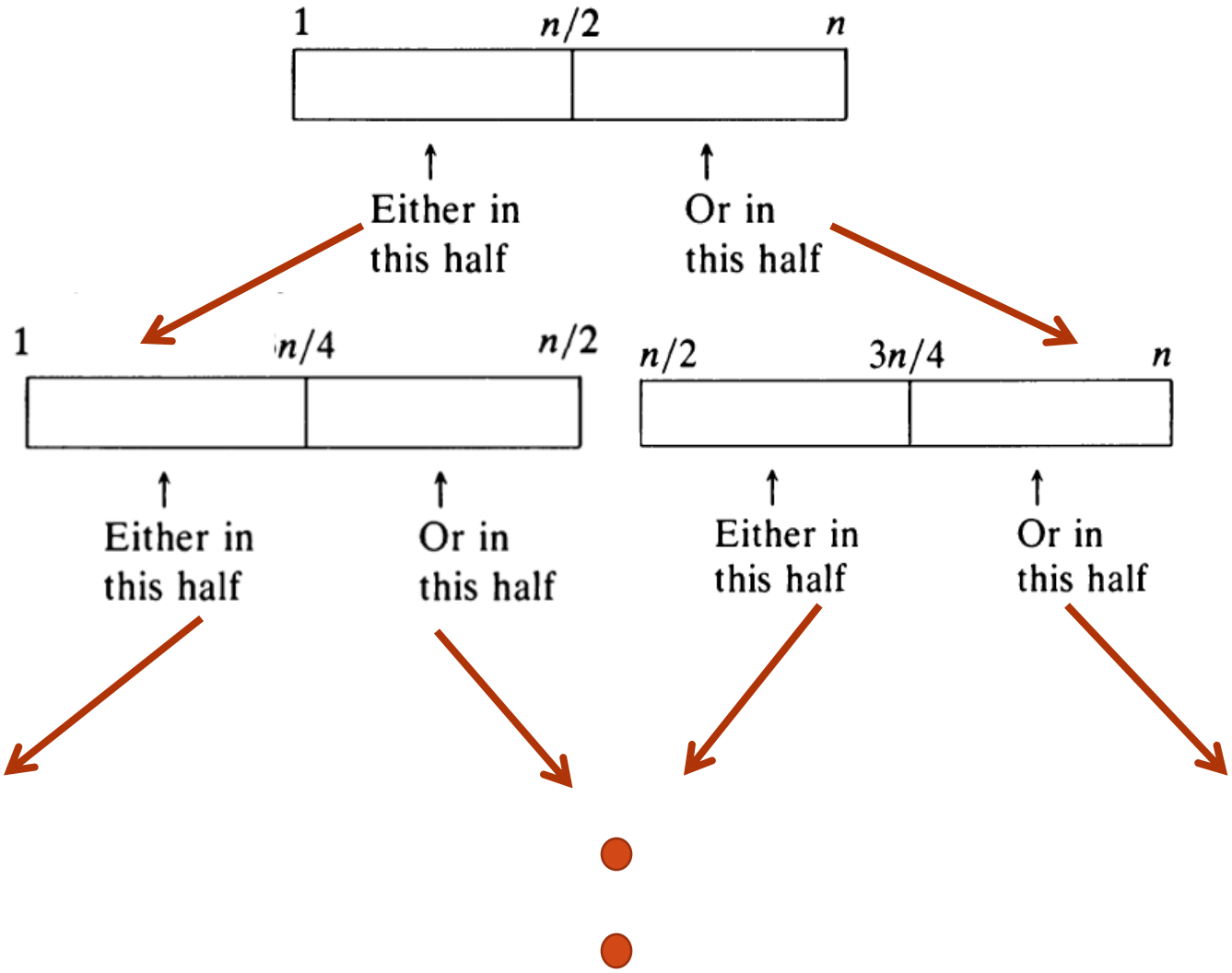
✓ ***I am now on the correct page.***

✓ *I have narrowed down a whole telephone directory to a single page.*

✓ *A few more and I would be down to a single entry.*

✓ *With Linear Search I would only have made it as far as Aarrons in the same time!*

Repeatedly perform halving



Organized entries

- ✓ *Today it seems inconceivable that a dictionary or telephone directory would be in anything but alphabetical order.*
- ✓ *However, one of the first ever dictionaries (in the 16th century),* ***John Withals' Shorte Dictionarie for Yonge Begynners****, was actually ordered by subject (Winchester, 1999).*
- ✓ *So the realization that alphabetical order would be the most useful organization was clearly not immediately obvious.*

Steps – Searching residential telephone directory

- *You pick a page.*
- *You would check what the names started with on the page you had opened the directory at.*
- *If you are at a **page before the name** you want,then*
  - ✓*you can rule out the first half of the directory and concentrate on the second half.*
    - ✓*This is possible because the telephone directory is sorted.*
    - ✓*If it were in a random order, you would be no nearer finding the right entry.*
- *If you have **overshot**, then*
  - ✓*you can rule out the second half and concentrate on the first half.*

Steps – Searching residential telephone directory *contd---*

- ✓ *And you can continue in the same way,*
  - ✓ *go to a page roughly half way through the part you have not discarded and discard another half.*
  - ✓ *keep doing this until you get to the single entry that is your friend's name.*

- ✓ *How many entries to search?*

Remember

- ✓ *Entries or elements are organized - you have a organized or sorted list.*
- ✓ *You get the middle entry or element.*
- ✓ *If the middle element equals to the searched value, then stop.*
- ✓ *Otherwise, two cases are possible:*
  - ***Searched value is less, than the middle element.***
    - ✓ *In this case, go to the step 1 for the part of the list, before middle element.*
  - ***Searched value is greater, than the middle element.***
    - ✓ *In this case, go to the step 1 for the part of the list, after middle element.*

Eg. Searching an alphabetized stack of award certificates

- ✓ *At each step of the search examine the middle item of the remaining group.*
- ✓ *By comparing the middle item to the sought item, half of the group can be eliminated from further consideration in this search.*

- Consider that your name is *Jones, Susan* and the name in the middle is *Smith, John*.
- Since the stack has been alphabetized and since *Jones* precedes *Smith* alphabetically, there is no need to consider the half of the data from Smith to the end.
- Similarly, if the middle name was *Gomez*, *Marc*, then the first half of the data (through *Gomez*) can be eliminated from further consideration.

Steps

Step 1

- Check the name on the *middle certificate in the stack*.
- If the middle certificate alphabetically *precedes* your name, then
  - set aside the top half of the stack.
- If the middle certificate alphabetically *follows* your name, then
  - set aside the bottom half of the stack.

Steps *contd*---

Step 2

- Check the name on the *middle certificate in the remaining stack*.
- If the middle certificate alphabetically *precedes* your name, then
  - set aside the top half of the remaining stack.
- If the middle certificate alphabetically *follows* your name, then
  - set aside the bottom half of the remaining stack.

✓ *Repeat Step 2 until either your certificate is found or the remaining stack has only one certificate that is not yours.*

Binary search

- ✓ *It involves having a lined up list of items in a known order (numerical or alphabetical order).*
- ✓ *This allows us to do checks that rule out half of the list at every step.*
- We check the *middle entry*.
- If the key is before the middle entry in the order
  - then the key must be in the first half of the list (because they are in order).
- If the key comes after the middle entry
  - then it must be in the second half.
- ✓ *We rule out half the list and do the same again on the part that remains – repeatedly until there is only one thing left.*
- ✓ *It is either the key, or the key is not there.*

✓ *A better search strategy …*

General strategy

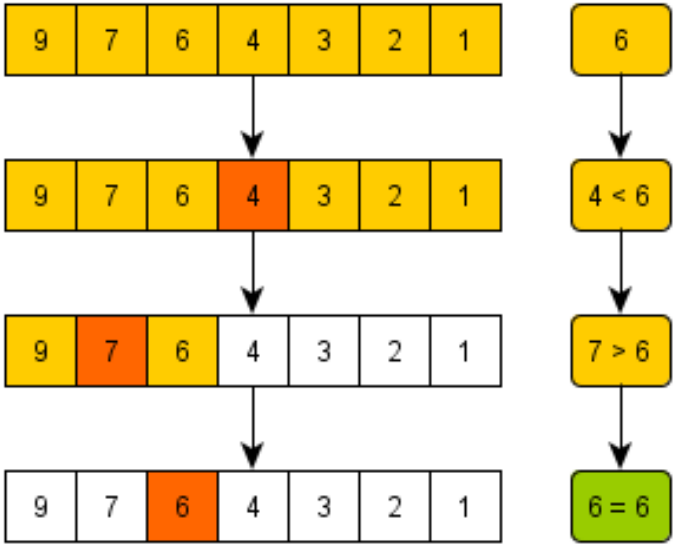
↑ Either in this half

↑ Or in this half

repeatedly

"examine middle value of remaining data and on the basis of this comparison eliminate half of the remaining data set"

Eg: Search for 6 in {9,7,6,4,3,2,1}



✓ *Find* 102 , 103 *in*

{-1, 5, 6, 18, 19, 25, 46, 78, 102, 114}

Binary search algorithm

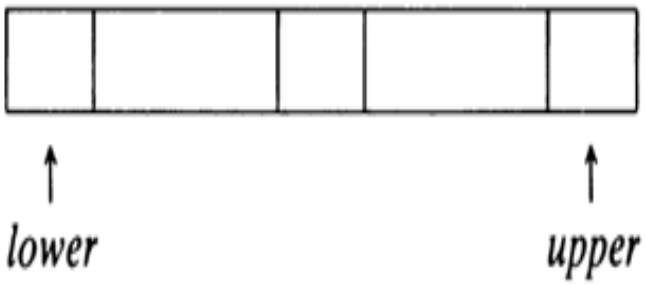
1. Establish ordered array size *n* and the value sought *x*.
2. Establish the ordered data set *a*[1..*n*].
3. Set *upper* and *lower* limits.

   Assign the *upper* and *lower* variables to the array limits.
4. Repeatedly
   (a) calculate middle position of remaining array
   (b) if value sought is greater than *middle* value
   then adjust *lower* limit to one greater than *middle*,
   else adjust *upper* limit to one less than *middle*

   until value sought is *found* or *lower* becomes greater than *upper*.
5. Set *found* accordingly.

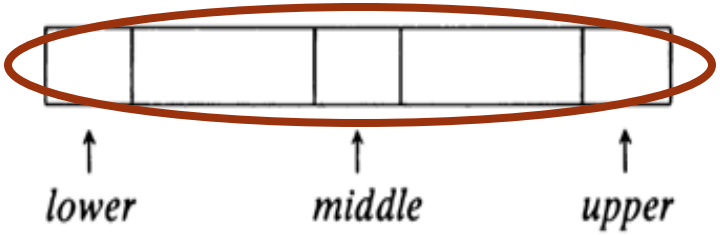
Binary search algorithm *contd----*

1. List of elements with lower and upper markers



array *a*[*lower*..*upper*]

2. Lower, upper and middle markers



middle := (*lower*+*upper*) **div** 2

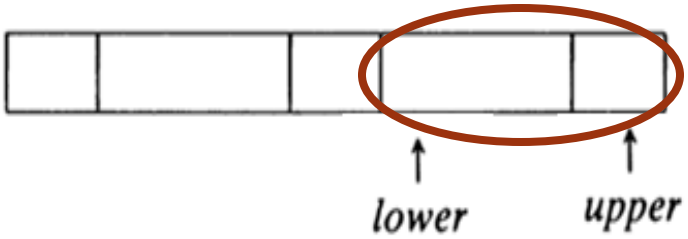
3. Test condition $x > a[middle]$

Binary search algorithm *contd----*

3. Test condition $x > a[middle]$

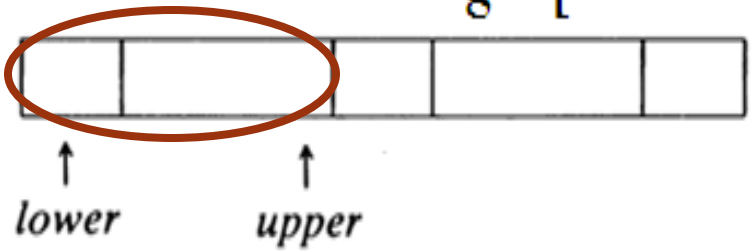
If this condition is true then x must be in the range $a[middle+1\,..\,upper]$

$lower := middle+1$



If this condition is false then x must be in the range a[lower .. middle]

$upper := middle-1$

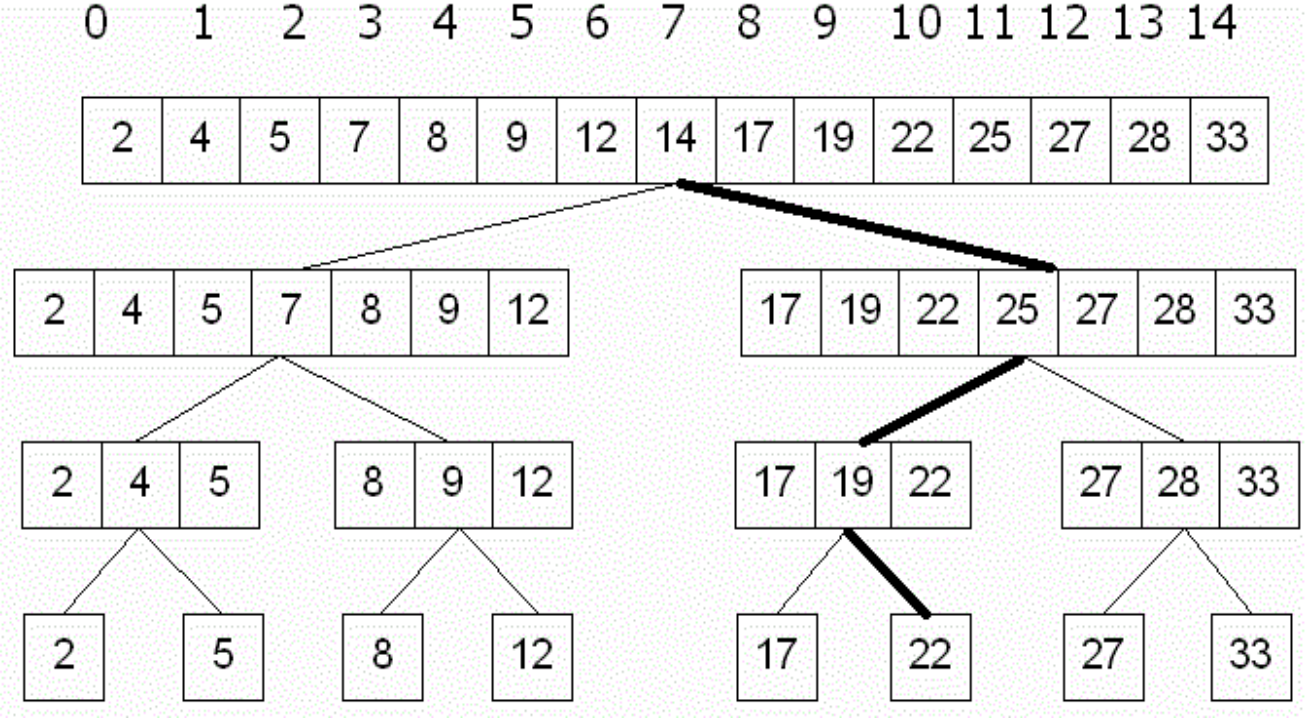


Binary search algorithm *contd*---

4. Repeat until $\textbf{until } (a[middle] = x) \textbf{ or } (lower > upper)$

5. Setting found $found := (a[middle] = x)$

Eg: Search for # in {2,4,5,7,8,9,12,14,17,19,22,25,27,28,33}



Searching the telephone directory-revisited

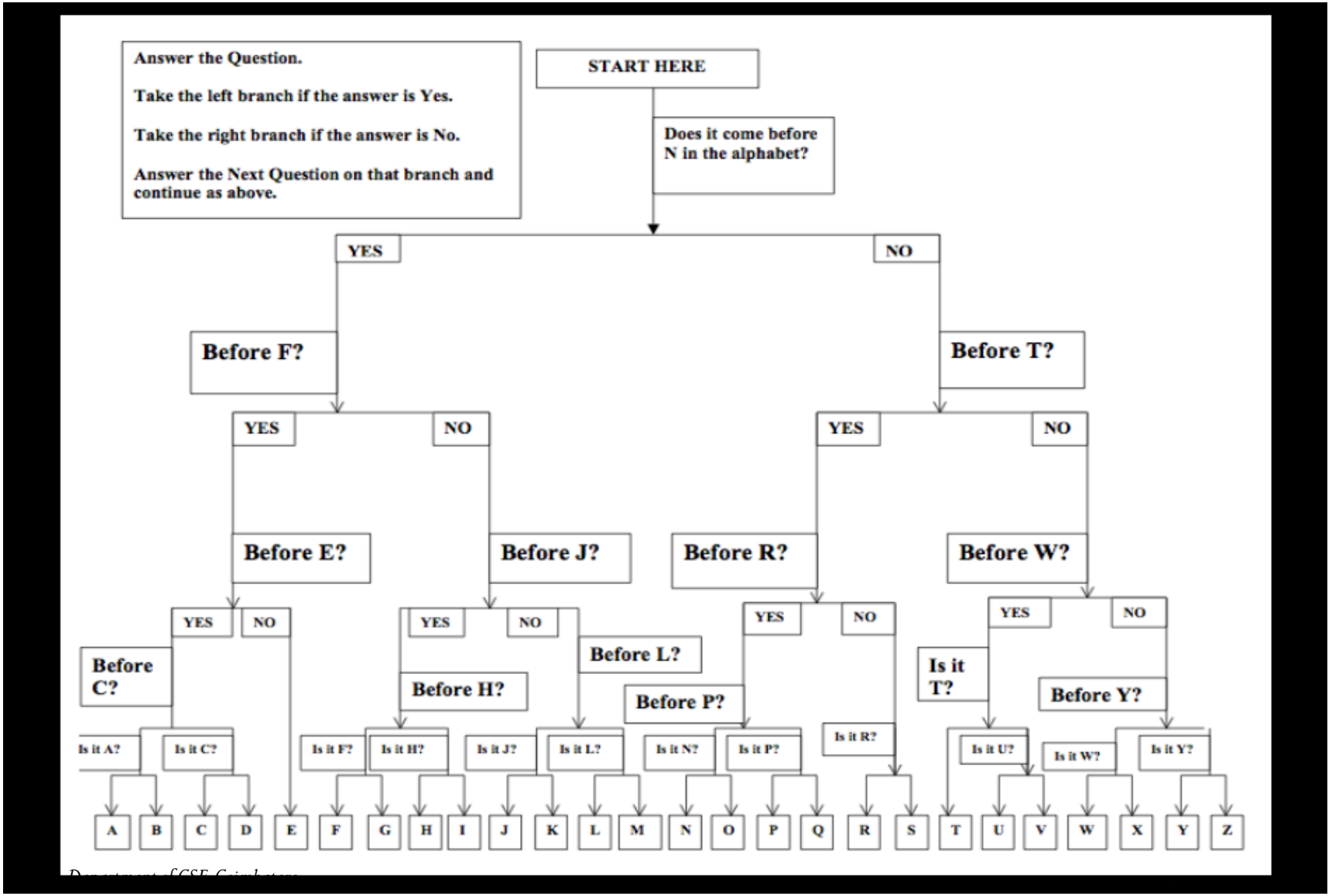
- ✓ *It probably occurred to you that when you actually search for a name in a telephone directory, (or a word in a dictionary) you would probably do it slightly differently.*

- ✓ *If looking for Steinbeck, you would not open it in the middle as that would almost certainly take you to the middle letter – M – as it did in the example above.*
- ✓ *S is nearer the end of the alphabet so you might open the directory two-thirds of the way through.*
- ✓ *You would do this on each step: using your knowledge of the positions of letters in the alphabet, to make a better guess than halfway each time.*
- ✓ ***This is an optimization of binary search** where you are using even more knowledge (the positions of letters in the alphabet) about the thing being sorted to speed things further.*

Diving-bell and the Butterfly-revisited

- ✓ *Jean-Dominique Bauby could have made his task much easier if he had known of such a search.*
- ✓ *They could have started with M.*
- ✓ *Instead of a blink meaning "**Yes**", it could have meant "**later in the alphabet**".*
- ✓ *A no blink could have meant "**earlier in the alphabet**".*
- ✓ *A double blink would mean "**that is the letter**".*
- ✓ *Each time the next letter read out would be roughly half way through the interval remaining.*
- ✓ *This would have taken the secretary longer to learn to do, perhaps, but it would have meant every letter of the alphabet could have been identified in only **5** blinks.*

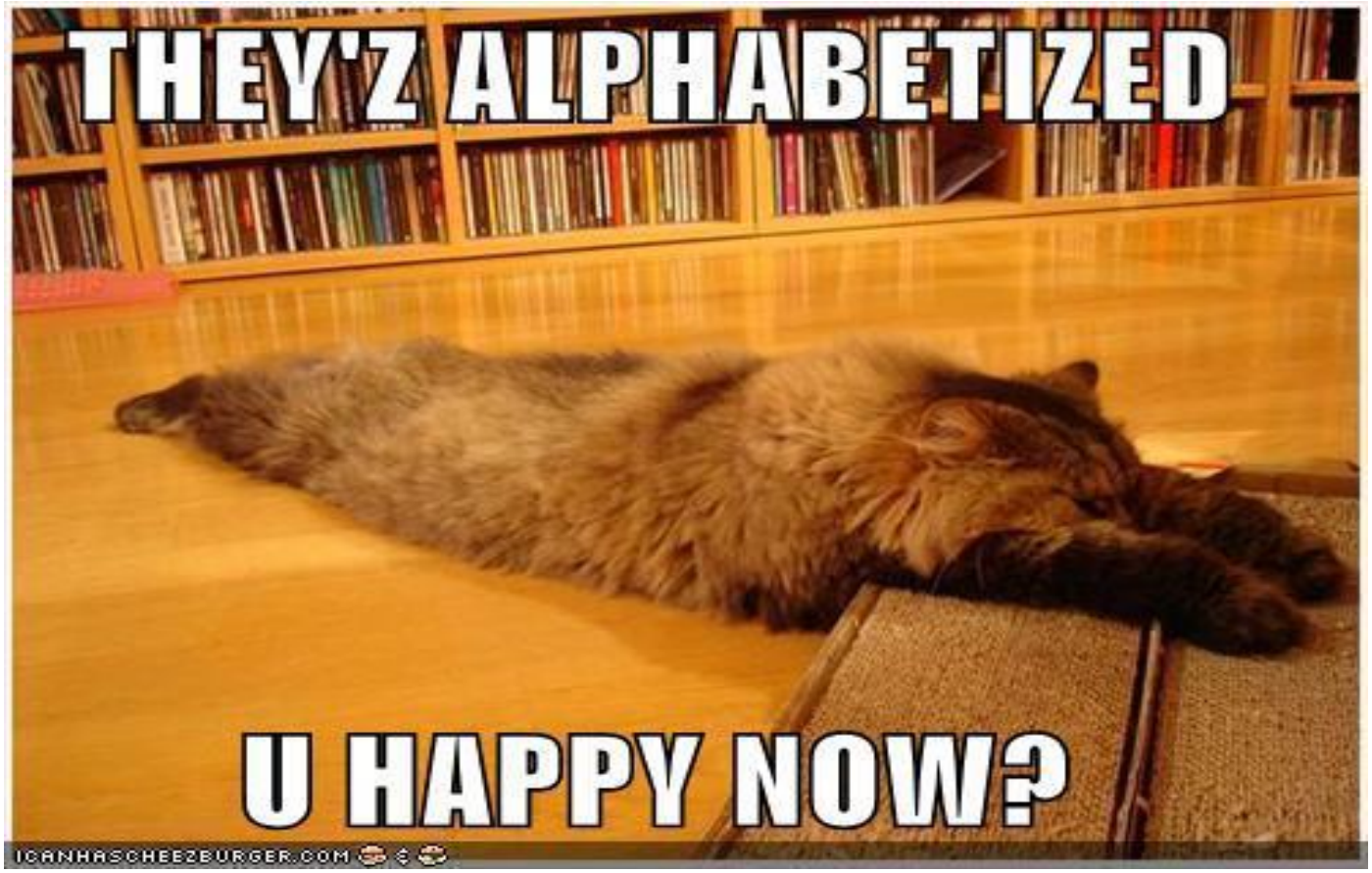
Eg. Divide the population and search for the letters



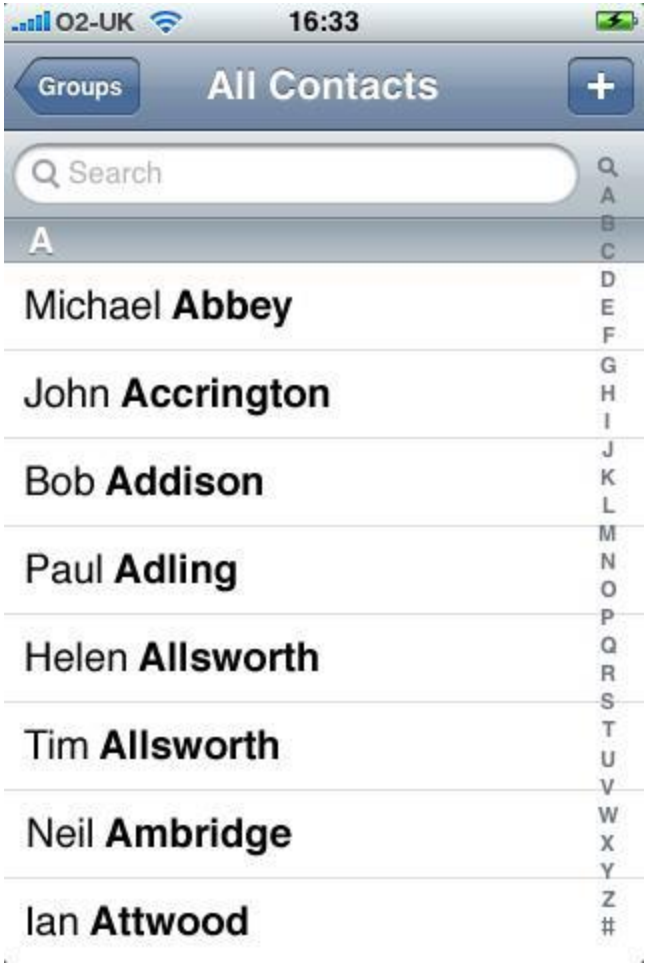
Data decomposition

- *Decomposition* is used in software design, multitasking etc.
- It involves dividing a program into separate instructions or groups of instructions.
- *Data decomposition* is another use of the concept of divide and conquer.
- The two search algorithms use data decomposition to perform searching.
  - ✓ *They divide the data into individual parts to be searched, but the way that this division occurs is quite different.*
  - ✓ *Linear search removes one item from the remaining group of items for each step of the algorithm, whereas binary search removes roughly half of the group at each step.*
  - ✓ *This distinction in the way that divide and conquer is applied leads to the significantly improved performance of binary search.*

Find a CD when sorted



Use name to find number



Using the call number to find book

Case study time …..

Example: Find the best fruit from the bin

Binary Search

given: a sorted list of data to search
given: a value to find

```
set foundAnswer  to no
set left to 0, right to listSize-1
while (foundAnswer  is no) and (left <= right):
    set mid to (left + ((right – left) / 2))
    if (item at mid) < (value to find):
            left = mid + 1
    otherwise if (item at mid) > (value to find):
            right = mid – 1
    otherwise:
            set foundAnswer  to yes
    output  mid
```

What has been described?

- How searching is performed using the intuitive linear search technique.
- The limitations in linear search technique.
- How halving the population enables efficient searching.
- The Binary search technique.
- Use of data decomposition in the two search techniques.

Credits

•Computing without computers, Paul Curzon;
http://www.eecs.qmul.ac.uk/~pc/research/education/puzzles/reading/
•2002 Computer Science Unplugged (www.unplugged.canterbury.ac.nz)
•How to solve it by computer, RG Dromey
•Google images