Queen Mary
University of London

20 Questions

Age group: 11 – adult

Abilities assumed: Knowledge of searching using linear search

Time: around 15 minutes,
can be used within a longer workshop activity

Size of group: anything from 2 to hundreds

Focus

Search algorithms: linear search versus binary search.
Divide and Conquer problem solving
Introduction to efficiency analysis
Computational Thinking

Syllabus Links

This activity can be used

- as a general introduction to a computing topics such as what an algorithm is and how they can be compared,
- to teach specific syllabus topics such as:
  KS3: understand several key algorithms (searching) that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
  AQA A'level 3.1.1 Problem Solving: Linear Search
  AQA A'level 3.3.2 Programming concepts: Binary search
  AQA A'level 3.3.1 Problem Solving: Abstraction, comparing algorithms

Summary

Play the game of 20 questions as a way to show the audience they already instinctively know the key to efficient searching and the basis of divide and conquer problem solving.

Aims

This activity aims to teach the idea of divide and conquer problem solving in the context of search algorithms. It also introduces the idea of efficiency analysis as a way of comparing algorithms.

Technical Terms

Search algorithm, linear search, binary search, efficiency analysis, divide and conquer.

Materials

Nothing special required.

What to do

The Grab:
There are several possible grabs to use for this activity. We use our "Locked-in" activity, which also introduces linear search as the grab. It involves a discussion of helping totally paralysed people to communicate (See Locked-in activity and Searching To Speak resource in Further Reading below).

The following is a short simple grab as an alternative: Point out that there are billions of web pages out there and yet it can find things in fractions of a second. Does it do linear search checking every web page in turn? Computers are not actually fast enough to do that! There must be cleverer search algorithms than linear search! This activity is about starting to explore how.

This grab could incorporate a simple computer activity. Get each student to choose a famous person and search for them using some search engine, reporting back how quickly the search was done.

The Set-up:
Review with the class how linear search works (e.g., ask the class to explain it). Set up a problem about searching for one thing amongst a million. If the class understand arrays then it could be about finding a single name in an array of a million names. If not it could be, for example, how you find a book to read on a shelf of books in a random order, or how you find a particular student's exam paper amongst a large pile of exam papers. The "Locked-in" activity is another possibility as the set-up for this.

The activity:
Play a game of 20 questions with the class. You think of a famous person and the class try and guess who you are thinking of by asking yes/no questions. They have 20 questions to guess correctly. However, you can only answer yes or no to the questions. As you play encourage the class to think about the kind of questions they are asking and what makes a good question.

A game might go.

"Are they female?"	*No*
"Are they alive?"	*No*
"Are they a film star?"	*No*
"Are they from Britain?"	*No*
"Are they from America?"	*No*
"Are they from Asia?"	*Yes*
"Are they from India?"	*Yes*
"Are they a politician?"	*Yes*
"Is it Ghandi?"	*Yes*

Point out that they didn't ask questions like: "Is it Adele?" or "Is it Usain Bolt?" until quite late. Ask them why not?

Ask them what is the best first question to ask, did they ask it and why is it good?

They are likely to have chosen male or female either first or very early. Encourage them to think about what is different about it to the other kind of question.

What makes it a good first question?

It's because it rules out half the possibilities, ***whatever the answer***. If you ask "Is it Adele?" then you rule out all but one if you are right, but if wrong (more likely) you only rule out one person. You would have to be lottery-winning lucky to do well that way. So the secret to playing 20 questions is to ask questions that rule out half the possibilities each time.

Asking a series of questions like "Is it Usain Bolt?" is exactly the same as doing linear search. You can imagine that you are lining up all the possible names in a line and checking each in turn.

You can see how much better halving questions are by doing a simple efficiency analysis. Suppose there are a million possible famous people that you could have been thinking of at the start. Count down, halving the number of people it could still be (keeping count on your fingers the number of questions asked: 1 million, 500,000, 250,000 people left, 125,000, 64,000 (ish to keep the numbers simple) 32,000, 16,000, 8000, 4000, 2000, 1000. After ten questions there are only a thousand people it could possibly be. Keep going and you find that it takes only 20 questions and you are down to one person left.

This shows that by following a divide and conquer approach to searching for something (here searching for the person someone was thinking of) we can massively improve how quickly we find what we are looking for. Linear search (checking things one at a time) would take on average half a million questions to find the thing we are looking for if there were a million possibilities. With a divide and conquer algorithm we can find it with only 20 questions as long as they are perfect questions. It is a massively faster algorithm.

The summary

Searching for things by checking each place one by one seems an obvious thing to do – it is a simple algorithm. However, it is very slow. Divide and conquer gives much faster algorithms. It is the approach to solving a problem where you split the problem in half in a way which leaves you with a simpler version of the same problem – you are still looking for the same thing, just on a problem half the size. Do that over and over and you get the answer incredibly quickly.

Binary search is one search algorithm that uses such a divide and conquer approach. It assumes your data is sorted. Rather than checking each piece of data in turn, you go to the middle element and use that as a signpost. You check whether the thing you are looking for is bigger or smaller that it and then only search the bottom or top half of the data as appropriate. Because the data is in order you can tell which side it must be. You keep doing that until you only have one piece of data left and check if it is the thing you are after. This is the same kind of divide and conquer style of algorithm as used playing 20 Questions, ruling out half the data with every question asked.

There are also some computational thinking lessons here. We have seen an algorithmic approach to problem solving (divide and conquer) that gives a massive improvement in speed of a task: here searching. It can be used more generally than just this situation. We've also seen a way of analysing an algorithm that allows us to compare different algorithms that do the same thing. This is based on an abstraction idea too, where rather than counting time we pick an operation (questions asked) and use it as our measure of work done. Finally we've seen how we can translate problems from one domain to another. A solution we naturally knew for how to play 20 questions gives us a way to solve other search problems like searching an array.

Variations and Extensions

Locked-in
This activity naturally follows the locked-in activity (see below). That can be used to introduce linear search as a solution to working out a letter a locked-in person only able to blink is thinking. After looking at its efficiency – on average 13 questions to work out a letter, point out that any computer scientist knows it takes 5 questions at worst. Point out also everyone knows the right questions to ask – but to show this you need to switch problems to the game of 20 Questions. After completing the 20 Questions exercise switch back to working out a letter someone is thinking – is there a divide and conquer solution? If so what are the questions? The answer is to ask questions that halve the alphabet – you ask "Is it before M in the alphabet?" first. This illustrates the idea of translating solutions from one problem to another. See "Further Reading" for a booklet giving this combined story.

Acting out binary search
You can get the class to act out binary searching of an array. Give each a random number on a card and get them to stand in a line in an order – they are just an array. Cards on the floor numbered from 0 can give the array index. Another student is given a value to search for. They go to the middle element and ask if it is larger or smaller. All students on one side sit down. They then go to the mid point of those left standing and keep repeating this process until only one person is standing. It is the card they are after. A follow up is to ask how they would know if the card searched for wasn't there. You could also contrast with linear search by doing that first.

Searching a telephone directory
To make the link to searching data more concrete hand out a paper telephone directory (or other similar alphabetically sorted list such as an index in a large book, or a list of contacts in a phone) and ask the student to look up different names. Have the class think about what algorithm they use – is it linear search or is it a divide and conquer approach. Do they go to the middle of the telephone directory whatever name is being looked for? Or do they follow an algorithm that is more efficient? If so get them to explain it. Have them think about at what point they switch to linear search and start running their fingers down the list of names. Have them consider why a telephone directory (or an index) is sorted in the first place. What other lists of things can they think of that are sorted and why – what are those lists used for?

CS Unplugged: Binary Search activity
Another follow-up activity is the CS Unplugged (www.csunplugged.org) activity on Binary Search. Having introduced the general idea of divide and conquer this gives a direct way of the students applying their understanding to a problem that directly mirrors binary search on an array.

Explanations and Pseudocode
Get the students to write clear descriptions of the algorithm used to play 20 Questions well in their own words. Students who have studied pseudocode could be asked to write pseudocode descriptions of the algorithm.

Programs

For more advanced students who can program, have them implement the different algorithms – eg searching an array which stores the letters of the alphabet in order. This could be incorporated into a program that flashes up the appropriate questions on the screen or visualises the alphabet and the questions in some easy to follow way that shows the possibilities remaining that might be used by a person with locked-in syndrome if connected to an appropriate input system.

Further Reading

Computational Thinking: Searching To Speak

This activity combined with the Locked-in activity written up in a booklet. It is available from http://teachinglondoncomputing.org/

Links to other activities

Locked-in

Explore the design of an algorithm to allow someone who is totally paralysed to communicate.

This gives an introduction to computational thinking based problem solving, leading to an understanding of what an algorithm is and how algorithms can be compared on the basis of efficiency. It also illustrates how computational thinking is about more than just creating computer-based solutions. Computing is about solving problems for people.

The intelligent piece of paper

Take part in a test of intelligence against an intelligent piece of paper!

This is a good introduction to what an algorithms is and what a computer program is before looking at search algorithms. It can also be used to start a discussion on what it would mean for a computer to be intelligent.

Winning Games: the perfect Tic-tac-toe player

Create a set of instructions that would allow anyone to play Tic-tac-toe perfectly.

This is a good follow on activity from the intelligent piece of paper – now create your own! It introduces programming and explores how a computer is able to win at board games like chess. The emphasis is on programming being about solving the problem rather than about being able to write in a programming language.

Live demonstration of this activity

Teaching London Computing give live sessions for teachers demonstrating this and our other activities. See http://teachinglondoncomputing.org/ for details. Videos of some activities are also available or in preparation.

Department for Education
SUPPORTED BY MAYOR OF LONDON
COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE