

3.1.3 Searching algorithms 2

Lesson plan and printable activities

Teacher notes

Lesson 2 PowerPoint contains information on how to follow the binary search (slides 11–19). Depending on the pace of learning during the earlier slides, it may be necessary to complete this section in a second lesson.

Materials needed

- 1. 3.1.3 Lesson 2 PowerPoint.
- 2. Binary search analysis Quiz.
- 3. Printable versions of slides 12-14.

Lesson aims

1. To get students to think about the mechanism of the binary search algorithm and to realise that there are differences in the efficiency of different searching algorithms.

Lesson objectives

- 1. Understand and explain how the binary search algorithm works.
- 2. Compare and contrast linear and binary search algorithms.

Starter activity (5 minutes)

1. Slide 2: Start with a short revision discussion that makes students think once more about the potential size of datasets that must be searched.

Main activities (40 minutes without extension task)

- 1. Slide 5: Definition of binary search.
- 2. Slide 6: Explain that the binary search algorithm is more complex to follow than a conventional linear search.

- 3. Slide 7: YouTube video. Watch from 3:12 minutes to 6:31 minutes.
- **4. Slide 8:** Show students an example of a binary search algorithm. This is the same algorithm used in the extension task.
- 5. Slide 9: Comparing linear and binary searches.

Plenary activity (20 minutes)

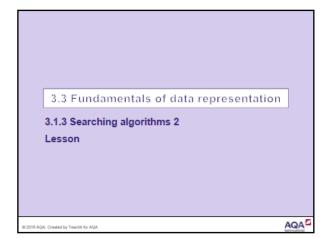
1. Slide 10: BBC Bitesize activity.

Students recap the information from this lesson and the previous one with this activity from BBC Bitesize: bbc.co.uk/education/guides/zgr2mp3/revision/2

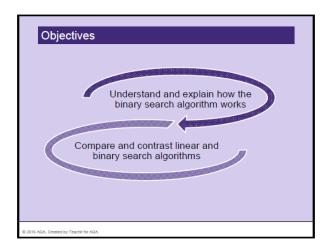
Extension task (Slides 11–19

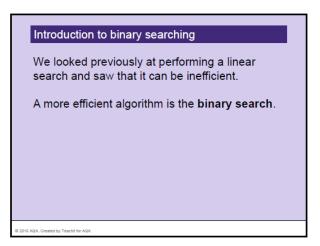
- Explain that you will be showing how a search term can be located within a 21-member array of data.
- Explain that a pre-requisite for using the binary search is that the dataset must be sorted, eg by ascending numerical order or by alphabetical order.
- Go through the example step-by-step, it is suggested that students are given a copy of the
 dry-running trace table provided and that they should come up with the variable values
 themselves. This can be tricky for many students but it's essential practice for other areas
 of the specification eg programming.
- Make it clear that the scope of the search 'area' diminishes with each iteration of the algorithm's loop.

Lesson

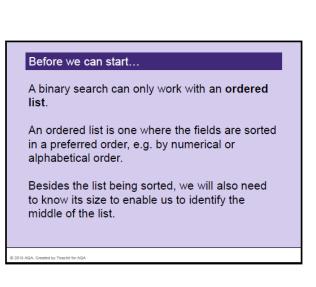


Think about this... A software company is asked to create a program for searching through a database of mobile-phone records for everyone owning a phone in the UK so that any individual's number can be retrieved. How many records might need to be searched through?





Definition of a binary search A method for searching data that splits datasets into two components repeatedly until the search term is located. What is a dataset? A collection of data, e.g. a table in a database. Why do we use the word binary? Binary implies two states – here we talk about splitting a dataset into two.



3.1 Fundamentals of algorithms



Store Search Term StartPointer <- 1 EndPointer <- DataSetSize Do MidPointer <- (StartPointer + EndPointer) / 2 (round answer down) If Record[MidPointer] < SearchTerm Then StartPointer <- MidPointer + 1 End If If Record[MidPointer] > SearchTerm Then EndPointer <- MidPointer - 1 End If Until Record[MidPointer] = SearchTerm OR StartPointer = EndPointer

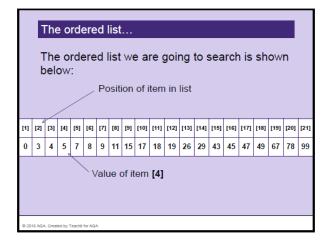
Comparing linear and binary searches

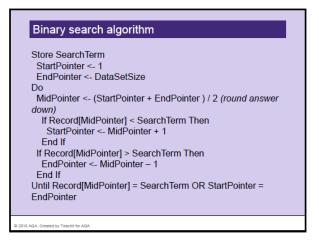
A linear search has an algorithm that is easier to understand, whereas the binary search algorithm is more complex.

The binary search will be much quicker than a linear search – particularly where the volume of data being searched is large.

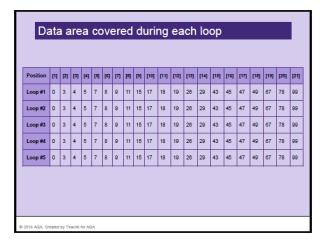
To round things off...

Recap with BBC Bitesize:
bbc.co.uk/education/guides/zgr2mp3/revision/2

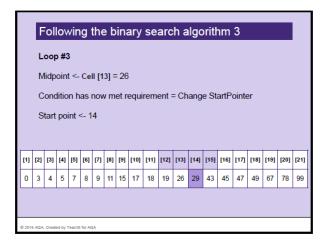




3.1 Fundamentals of algorithms



		F	ol	lo۷	vin	g	the	e b	in	ary	se	arc	h a	ılgo	ritl	nm	1				
			oop ear						ing	loop	o, St	artP	ointe	er =	1, E	ndP	ointe	er = :	21,		
	Midpoint <- Cell [11] = 18																				
	Condition now met has requirement = Change StartPointer																				
	Start point <- 12																				
Ļ	Gunt point = 12																				
ı	1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
0	2016	AQA.	Creat	ed by 1	Teachit	t for AG	QA .														



		F	ol	lo۷	vin	g	the	e b	ina	ary	se	arc	h a	ılgo	orith	ım	5				
		L	oop	#5	5																
		M	lidp	oint	t <-	Се	II [1	5] =	43	3											
	Condition met = Search term matched																				
	Exit program!																				
	Ext program:																				
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
ľ																					
6	2016	AQA.	Creat	ed by T	Teachit	t for AC	2A														

	Trace	table	:					
	DataSetSize	SearohTerm	StartPointer	EndPointer	MidPointer	Record[MidPointer] < SearchTerm	Record[MidPointer] > SearchTerm	[END] Record[MidPointer] = SearchTerm OR StartPointer = EndPointer
						Only one of	these conditions should	be met for each iteration.
Fixed	21							
Input		43						
Initial			1	21				
Loop 1			12		(1 + 21 = 22)/2 = 11	18 < 43		
Loop 2				15	(12 + 21 = 33)/2 = 16°		45 > 43	
Loop 3			14		(12 + 15 = 27)/2 = 13°	26 < 43		
Loop 4			15		(14 + 15 = 27)/2 = 14"	29 < 43		
Loop 6					(15 + 15 = 30)/2 = 15°			Met! Record[MidPointer] = SearchTerm
								* = Rounded down
© 2016 A	QA. Created by	Teachit for AQ/						

	F	ol	٥٧	vin	g	the	e b	ina	ary	se	arc	h a	lgc	orith	nm	2				
	L	oop	#2	2																
Midpoint <- Cell [16] = 45																				
Condition now met has requirement = Change EndPointer																				
End point <- 15																				
	Ena point ~ 15																			
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
																	•			
0 201	6 AQA	Creat	ed by T	Teachi	for AG	M.														

	F	ol	lo۷	vin	g	the	b	ina	ary	se	arc	h a	lgc	orith	nm	4				
	L	oop	#4	ı																
	Midpoint <- Cell [14] = 29																			
	Condition has now met requirement = Change StartPointer																			
	Start point <- 15																			
	Sant Point 10																			
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
© 2016	AQA.	Creat	ed by T	eachi	for AG	iA.														

Quiz – Binary search analysis

Question 1

Here is an array of sorted numerical data that is to be searched until the term '78' is located. How many times does the loop get **entered** before the search term is found? Show your working as well as the final answer.

Note: Not all the loops provided in the table may be needed.

Algorithm

Store SearchTerm
StartPointer <- 1
EndPointer <- DataSe

EndPointer <- DataSetSize

Do

MidPointer <- (StartPointer + EndPointer) / 2 (Rounded down if needed)

If Record[MidPointer] < SearchTerm Then

StartPointer <- MidPointer + 1

End If

If Record[MidPointer] > SearchTerm Then

EndPointer <- MidPointer – 1

End If

Until Record[MidPointer] = SearchTerm OR StartPointer = EndPointer

Array contents

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
23	35	67	78	99	101	102	109	123	145	178	179	200	201	202

Trace table

	StartPointer	EndPointer	MidPointer	Record[MidPointer] < SearchTerm	Record[MidPointer] > SearchTerm	Record[MidPointer] = SearchTerm OR StartPointer = EndPointer
Initial						
Loop #1						
Loop #2						
Loop #3						
Loop #4						

Quiz – Binary search analysis – answers

Question 1

Correctly identifies 15 elements and labels.

Correct values on each line for Loops only StartPointer and EndPointer and MidPointer.

States that the loop was entered twice.

Algorithm

Store SearchTerm
StartPointer <- 1
EndPointer <- DataSetSize

Do

MidPointer <- (StartPointer + EndPointer) / 2 (Rounded down if needed)

If Record[MidPointer] < SearchTerm Then

StartPointer <- MidPointer + 1

End If

If Record[MidPointer] > SearchTerm Then

EndPointer <- MidPointer - 1

End If

Until Record[MidPointer] = SearchTerm OR StartPointer = EndPointer

Array contents

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
23	35	67	78	99	101	102	109	123	145	178	179	200	201	202

Trace table - [DataSetSize = 15, SearchTerm = 78]

	StartPointer	EndPointer	MidPointer	Record[MidPointer] < SearchTerm	Record[MidPointer] > SearchTerm	Record[MidPointer] = SearchTerm OR StartPointer = EndPointer
Initial	1	15				
Loop #1		7	8		109 > 78	
Loop #2			4			Record[MidPointer] = SearchTerm

Extension task

Binary search algorithm

Store SearchTerm
StartPointer <- 1
EndPointer <- DataSetSize
Do
MidPointer <- (StartPointer + EndPointer) / 2 (Rounded down if needed)
If Record[MidPointer] < SearchTerm Then
StartPointer <- MidPointer + 1
End If
If Record[MidPointer] > SearchTerm Then
EndPointer <- MidPointer - 1
End If
Until Record[MidPointer] = SearchTerm OR StartPointer = EndPointer

Data area covered during each loop

Position	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]
Loop #1	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
Loop #2	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
Loop #3	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
Loop #4	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99
Loop #5	0	3	4	5	7	8	9	11	15	17	18	19	26	29	43	45	47	49	67	78	99

Trace table

	DataSetSize	SearchTerm	StartPointer	EndPointer	MidPointer	Record[MidPointer] < SearchTerm	Record[MidPointer] > SearchTerm	[END] Record[MidPointer] = SearchTerm OR StartPointer = EndPointer
Fixed	21							
Input		43						
Initial			1	21				
Loop #1			12		(1 + 21 = 22)/2 = 11	18 < 43		
Loop #2				15	(12 + 21 = 33)/2 = 16*		45 > 43	
Loop #3			14		(12 + 15 = 27)/2 = 13*	26 < 43		
Loop #4			15		(14 + 15 = 27)/2 = 14*	29 < 43		
Loop #5					(15 + 15 = 30)/ 2 = 15*			Met! Record[MidPointer] = SearchTerm

^{* =} Rounded down