Chapter 5: Algorithms

ANSWERS TO QUESTIONS IN THE STUDENT’S BOOK

Page 52

Question
1. This chapter looks at searching, sorting and shortest path algorithms. Find four other types of algorithm.

Answer
- Algorithms applied to graphics (e.g. blur)
- Compression algorithms
- Hashing algorithms
- Encryption algorithms
- Algorithms for generating pseudo random numbers
- Algorithms for removing noise from a signal
- Error detection/correction algorithms

Question
2. Perform a linear search and a binary search to find Peru in the following list:
   Argentina, Bolivia, Brazil, Chile, Colombia, Ecuador, French Guiana, Guyana, Paraguay, Peru, Suriname, Uruguay, Venezuela

Answer
   Should demonstrate checking one by one until Peru is found.

Question
3. Describe the circumstances in which you might choose to use a linear search over a binary search.

Answer
   Linear search would be used over binary search if the list is not sorted in order by the field being sought.

Page 53

Question
1. Demonstrate how to do a bubble sort on the following lists:
   (a) B, A, E, D, C, F
   (b) F, A, B, C, D, E
   (c) B, C, D, E, F, A

Answer
   Students should demonstrate the sort. List (b) should only take two passes whereas (c) should take the most work.
Question

2.  
   (a) Write a program that creates a random array of integers and performs a bubble sort on them.
   (b) Amend the program so it allows you to specify the size of the array and outputs the time taken to perform the sort.
   (c) Compare the time taken to sort lists of 10, 100, 1000 and 10 000 integers.

Answer

Code used will vary by language. For examples of bubble sort in different languages see: rosettacode.org/wiki/Sorting_algorithms/Bubble_sort

   (a) Bubble sort has a time complexity of $O(n^2)$ so time should increase significantly as list size goes up by a factor of 10.

Question

3. Various methods have been used to improve the efficiency of bubble sort. Try to find out some of these and comment on their effectiveness.

Answer

Methods include:
- bubbling through left to right, then right to left (i.e. cocktail sort)
- sectioning off one extra from the right each iteration as ‘sorted’, meaning you need to iterate through a list one smaller each pass.

Both methods give noticeable improvements. The former gets rid of the problem of having low values at the right of the list trickling back one by one. It should be noted that even with these amendments bubble sort still performs poorly compared to alternative algorithms.

Page 55

Question

1. Demonstrate an insertion sort on:
   (a) D, G, F, B, A, H, C, E
   (b) A, B, C, D, H, G, F, E
   (c) B, C, D, E, F, G, H, A

Answer

Students should demonstrate how the sort is performed following the algorithm in the book.
Question
2.
   (a) Write a program that creates a random array of integers and performs an insertion sort on
       them.
   (b) Amend the program so it allows you to specify the size of the array and outputs the time
       taken to perform the sort.
   (c) Compare the time taken to sort lists of 10, 100, 1000 and 10000 integers.

Answer
(b) Details of implementation will vary by language. Examples of insertion sort in
    different languages can be found here: rosettacode.org/wiki/Sorting_algorithms/Insertion_sort
(c) Students should note that the time taken is much less than bubble sort but it will
    still show a worse than linear time increase, i.e. O(n^2).

Page 57

Question
1. Demonstrate a merge sort on:
   (a) D, G, F, B, A, H, C, E
   (b) A, B, C, D, H, G, F, E
   (c) B, C, D, E, F, G, H, A

Answer
Students should demonstrate how the sort is performed following the algorithm in the book.

Question
2.
   (a) Write a program that creates a random array of integers and performs a merge sort on
       them.
   (b) Amend the program so it allows you to specify the size of the array and outputs the time
       taken to perform the sort.
   (c) Compare the time taken to sort lists of 10, 100, 1000 and 10000 integers.

Answer
(b) Details of implementation will vary by language. Examples of merge sort in
    different languages can be found here: rosettacode.org/wiki/Sorting_algorithms/Merge_sort
(c) Merge sort grows with linearithmic complexity (linearithmic is beyond the
    specification). This means students should observe a growth slower that the
    polynomial of bubble and insertion sort but worse than linear growth. Try to get
    students to consider the difference between absolute time and growth.
**Question**

1. Demonstrate a recursive or in-place quicksort on:
   - (b) A, B, C, D, H, G, F, E
   - (c) B, C, D, E, F, G, H, A

**Answer**

Students should demonstrate how the sort is performed following the algorithm in the book.

**Question**

2. 
   - (a) Write a program that creates a random array of integers and performs a quicksort on them.
   - (b) Amend the program so it allows you to specify the size of the array and outputs the time taken to perform the sort.
   - (c) Compare the time taken to sort lists of 10, 100, 1000 and 10 000 integers.

**Answer**

(b) Details of implementation will vary by language. Examples of (mainly in place) quick sort in different languages can be found here: [rosettacode.org/wiki/Sorting_algorithms/Quicksort](http://rosettacode.org/wiki/Sorting_algorithms/Quicksort)

(c) Students should observe it is faster than merge sort but it shows the same growth.

**Question**

1. An algorithm takes $2n^4+n-1$ steps to run on a data set $n$ big. Express its time complexity in Big-O notation.

**Answer**

$O(n^4)$

**Question**

2. An algorithm takes $6n+3$ steps to run on a data set $n$ big. Express its time complexity in Big-O notation.

**Answer**

$O(n)$

**Question**

3. An algorithm takes $2n^2+2n+2$ steps to run on a data set $n$ big. Express its time complexity in Big-O notation.
Answer
O(n²)

Question
4. An algorithm takes 10 steps to run on a data set $n$ big. Express its time complexity in Big-O notation.

Answer
O(1)

Page 62

Question
1. Algorithm A blurs a 1 000 000 pixel image in 1 second; Algorithm B blurs the same image in 0.7 seconds. One algorithm has a time complexity of O(n) the other O(n²).
   (a) Is it possible to determine which algorithm has which complexity?
   (b) If the answer to (a) is yes, which algorithm has which complexity? If no, what additional information would you need?

Answer
(a) No.
   (b) You need to know the time taken for the algorithm on different-sized images (preferably several).

Question
2. Find out the time complexities in Big-O notation to: bubble sort, insertion sort, merge sort and quicksort. For each, decide if they are linear, constant, polynomial, logarithmic or exponential.

Answer
Bubble sort and insertion sort are polynomial O(n²).
Merge sort and quicksort use a different complexity called linearithmic, which performs almost (but not quite) as well as linear O(n log n).

Question
3. Find out the time complexities of binary search and linear search. For each, decide if they are linear, constant, polynomial, logarithmic or exponential.

Answer
Linear search O(n)
Binary search O(log n)

Page 71

Question
Apply Dijkstra’s algorithm to find the shortest path from A to J on the graph shown.
### Answer

<table>
<thead>
<tr>
<th>Node</th>
<th>Distance from A</th>
<th>Previous node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (V)</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>B (V)</td>
<td>75</td>
<td>A</td>
</tr>
<tr>
<td>C (V)</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>D (V)</td>
<td>85 80</td>
<td>B</td>
</tr>
<tr>
<td>E (V)</td>
<td>65</td>
<td>C</td>
</tr>
<tr>
<td>F (V)</td>
<td>40</td>
<td>C</td>
</tr>
<tr>
<td>G (V)</td>
<td>50</td>
<td>F</td>
</tr>
<tr>
<td>H (V)</td>
<td>90</td>
<td>G</td>
</tr>
<tr>
<td>I (V)</td>
<td>90</td>
<td>D</td>
</tr>
<tr>
<td>J (C)</td>
<td>135 105</td>
<td>H</td>
</tr>
</tbody>
</table>

The shortest route is \(A \rightarrow C \rightarrow F \rightarrow G \rightarrow H \rightarrow J\), which is 105.

### Question

Apply Dijkstra’s algorithm to find the shortest path from A to J on the graph provided.

### Answer

Following the algorithm \(A \rightarrow G \rightarrow J\) is the shortest route. Students should note that they have to expand many nodes in the opposite direction before they get there.

### Question

Use A* search to find the shortest path from A to J on the graph provided.

### Answer

<table>
<thead>
<tr>
<th>Node</th>
<th>Distance from A</th>
<th>Heuristic distance</th>
<th>Distance from A+ Heuristic distance</th>
<th>Previous node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (V)</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>–</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>65</td>
<td>140</td>
<td>A</td>
</tr>
<tr>
<td>C (V)</td>
<td>10</td>
<td>60</td>
<td>70</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>85</td>
<td>55</td>
<td>140</td>
<td>F</td>
</tr>
<tr>
<td>E</td>
<td>65</td>
<td>50</td>
<td>115</td>
<td>C</td>
</tr>
<tr>
<td>F (V)</td>
<td>40</td>
<td>50</td>
<td>90</td>
<td>C</td>
</tr>
<tr>
<td>G (V)</td>
<td>50</td>
<td>45</td>
<td>95</td>
<td>F</td>
</tr>
<tr>
<td>H (V)</td>
<td>90</td>
<td>10</td>
<td>100</td>
<td>G</td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J (C)</td>
<td>105</td>
<td>0</td>
<td>105</td>
<td>H</td>
</tr>
</tbody>
</table>

The shortest route is 105 going \(A \rightarrow C \rightarrow F \rightarrow G \rightarrow H \rightarrow J\).

### Question

Perform an A* search to find the shortest path from A to J.
**Answer**

Again it is $A \rightarrow G \rightarrow J$.

Students should note it finds the answer much quicker than with Dijkstra’s.

**Question**

1. What would happen if A* used a heuristic that wasn’t admissible (that is, overestimated the distance to the end node)?

**Answer**

The best route could get overlooked as the heuristic might suggest it is much worse than it actually is.

**Question**

2. As a heuristic underestimates the distance more and more, how does this affect A*’s effectiveness?

**Answer**

A* becomes slower at finding the correct answer as the heuristic underestimates more and more. The extreme is where the heuristic is simply 0 – this is equivalent to Dijkstra’s algorithm. Note that as long as the heuristic is admissible, A* will always find the best route.

**Page 80**

**Question**

For the given starting order show the first ten nodes to be generated by:

(a) a depth-first search

(b) a breadth-first search.

**Answer**

Exact configurations shown will vary depending on which order students choose to slide the tiles. Depth-first should from the starting node generate the left-most and continue deepening the left-most node. The most able students may take into account not repeating states and so when a repeated state is encountered will go up a level.

Breadth-first will generate all three children shown plus their children. If students are avoiding repeating nodes there may be some children on a fourth layer.

**Question**

Research and describe an alternative heuristic that could be used: The sum of the Manhattan distances. Is this better or worse than the one suggested?
Answer

The sum of the Manhattan distances involves calculating the number of places vertically and horizontally each tile is from its actual place then adding up the total. Whilst Manhattan distance is computationally more expensive to calculate it tends to give a better estimate than tiles out of place (which is more likely to underestimate). The Manhattan distance is therefore the preferable heuristic.

Crucially both using the Manhattan distances and tiles out of place are admissible (i.e. don’t overestimate the number of moves needed).