The Corner on Your Lips

Just another WordPress.com site

Five Algorithm Approaches

leave a comment »

APPROACH 1: EXAMPLIFY

Description: Write out specific examples of the problem, and see if you can figure out a general rule.

Example: Given a time, calculate the angle between the hour and minute hands.

Approach: Start with an example like 3:27.We can draw a picture of a clock by selecting where the 3 hour hand is and where the 27 minute hand is.

By playing around with these examples, we can develop a rule:

Minute angle (from 12 o’clock): 360 * minutes / 60

Hour angle (from 12 o’clock): 360 * (hour % 12) / 12 + 360 * (minutes / 60) * (1 / 12)

Angle between hour and minute: (hour angle – minute angle) % 360

By simple arithmetic, this reduces to 30 * hours – 5.5 * minutes.

APPROACH 2: PATTERN MATCHING

Description: Consider what problems the algorithm is similar to, and figure out if you can modify the solution to develop an algorithm for this problem.

Example: A sorted array has been rotated so that the elements might appear in the order 3 4 5 6 7 1 2.How would you find the minimum element?

Similar Problems:

Find the minimum element in an array.

Find a particular element in an array (eg, binary search).

Algorithm:

Finding the minimum element in an array isn't a particularly interesting algorithm (you could just iterate through all the elements), nor does it use the information provided (that the array is sorted).It's unlikely to be useful here.

However, binary search is very applicable.You know that the array is sorted, but rotated.So, it must proceed in an increasing order, then reset and increase again.The minimum element is the "reset" point.

If you compare the first and middle element (3 and 6), you know that the range is still increasing.This means that the reset point must be after the 6 (or, 3 is the minimum element and the array was never rotated).We can continue to apply the lessons from binary search to pinpoint this reset point, by looking for ranges where LEFT > RIGHT.That is, for a particular point, if LEFT < RIGHT, then the range does not contain the reset.If LEFT > RIGHT, then it does.

APPROACH 3: SIMPLIFY & GENERALIZE

Description: Change a constraint (data type, size, etc) to simplify the problem.Then try to solve it.Once you have an algorithm for the "simplified" problem, generalize the problem again.

Example: A ransom note can be formed by cutting words out of a magazine to form a new sentence.How would you figure out if a ransom note (string) can be formed from a given magazine (string)?

Simplification: Instead of solving the problem with words, solve it with characters.That is, imagine we are cutting characters out of a magazine to form a ransom note.

Algorithm:

We can solve the simplified ransom note problem with characters by simply creating an array and counting the characters.Each spot in the array corresponds to one letter.First, we count the number of times each character in the ransom note appears, and then we go through the magazine to see if we have all of those characters.

When we generalize the algorithm, we do a very similar thing.This time, rather than creating an array with character counts, we create a hash table.Each word maps to the number of times the word appears.

APPROACH 4: BASE CASE AND BUILD

Description: Solve the algorithm first for a base case (e.g., just one element).Then, try to solve it for elements one and two, assuming that you have the answer for element one.Then, try to solve it for elements one, two and three, assuming that you have the answer to elements one and two.

Example: Design an algorithm to print all permutations of a string.For simplicity, assume all characters are unique.

Test String: abcdefg

Case “a” –> {a}

Case “ab” –> {ab, ba}

Case “abc” –> ?

This is the first “interesting” case.If we had the answer to P(“ab”), how could we generate P(“abc”).Well, the additional letter is “c”, so we can just stick c in at every possible point.That is:

merge(c, ab) –> cab, acb, abc

merge(c, ba) –> cba, bca, bac

Algorithm: Use a recursive algorithm.Generate all permutations of a string by “chopping off” the last character and generating all permutations of s[1… n-1].Then, insert s[n] into every location of the string.

NOTE: Base Case and Build Algorithms often lead to natural recursive algorithms.

APPROACH 5: DATA STRUCTURE BRAINSTORM

Description: This is hacky, but it often works.Simply run through a list of data structures and try to apply each one.

Example: Numbers are randomly generated and stored into an (expanding) array.How would you keep track of the median?

Data Structure Brainstorm:

Linked list? Probably not – linked lists tend not to do very well with accessing and sorting numbers.

Array? Maybe, but you already have an array.Could you somehow keep the elements sorted? That’s probably expensive.Let’s hold off on this and return to it if it’s needed.

Binary tree? This is possible, since binary trees do fairly well with ordering.In fact, if the binary search tree is perfectly balanced, the top might be the median.But, be careful – if there’s an even number of elements, the median is actually the average of the middle two elements.The middle two elements can’t both be at the top.This is probably a workable algorithm, but let’s come back to it.

Heap? A heap is really good at basic ordering and keeping track of max and mins.This is actually interesting – if you had two heaps, you could keep track of the biggest half and the smallest half of the elements.The biggest half is kept in a min heap, such that the smallest element in the biggest half is at the root.The smallest half is kept in a max heap, such that the biggest element of the smallest half is at the root.Now, with these data structures, you have the potential median elements at the roots.If the heaps are no longer the same size, you can quickly “rebalance” the heaps by popping an element off the one heap and pushing it onto the other.

Note that the more problems you do, the better instinct you will develop about which data structure to apply.

Written by liushiyu

July 1, 2010 at 11:37 pm

Posted in Uncategorized