Interesting algorithms to develop CS1110

Submit your solutions, in any language, to gries or lee. Please try to give some documentation to help the reader understand correctness

These problems are for those who have had previous programming experience to be able to compare their programming skills now with their skills in 3-4 weeks. Doing these problems is voluntary, but we too are interested in how well you do now, so please send us your solutions.

Do them using your current knowledge. It doesn't help to look up the solutions elsewhere and copy. No one learns anything from that.

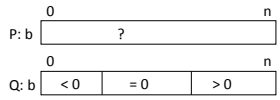
1

Dutch National Flag

Given is an int array segment b[0..n]. Write an algorithm to swap the elements to put the negative ones first, then all the zeros (there may be many of them, who knows?) and then the positive ones. The only way the elements should be changed is to swap two of them. You can write this, for example, as

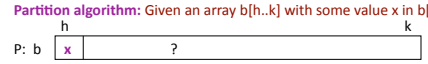
Swap b[i] and b[j].

Below, P shows the precondition and Q the postcondition, as pictures

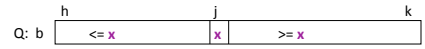


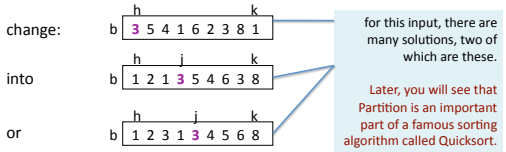
2

Partition algorithm: Given an array b[h..k] with some value x in b[h]:



Swap elements of b[h..k] and store in j to truthify Q:





for this input, there are many solutions, two of which are these.

Later, you will see that Partition is an important part of a famous sorting algorithm called Quicksort.

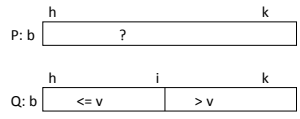
x is called the pivot value.

x is not a program variable; x just denotes the value initially in b[h].

3

Binary search: Vague spec: Look for v in **sorted** array segment b[h..k].
Better spec:
Precondition P: b[h..k] is sorted (in ascending order).
Store in i to truthify:
Postcondition Q: b[h..i] <= v and v < b[i+1..k]

Below, the array is in non-descending order:

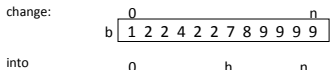


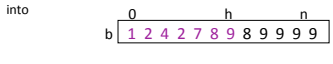
Called binary search because each iteration of the loop cuts the array segment still to be processed in half

So, at the end, if v is in b[h..k], b[i] is the rightmost occurrence of v. If v is not in b[h..k], it belongs between b[i] and b[i-1].

4

Remove adjacent duplicates

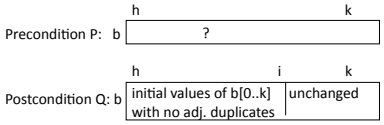




don't care what is in b[k+1..n]

Truthify:

b[0..h] = initial values in b[0..n] but with adjacent duplicates removed



5

Saddleback search

Given is a two-dimensional array b[0..m][0..n]. Each row and column is in ascending order. An example appears below. A value x is guaranteed to be in b[0..m, 0..n].

Write an algorithm to find x –store in i and j so that x = b[i][j]. How fast is your algorithm? Are you using the fact that the rows and columns are sorted to get a fast algorithm?

b	0	1	2	3
0	2	4	4	5
1	2	4	5	6
2	3	4	5	7
3	3	5	5	8
4	3	6	7	9

if x is 7, there are two possible answers: [2, 3] and [4, 2]. Either one may be used.

6