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.

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

- **Precondition:** $b[h..k]$ is sorted (in ascending order).
- **Vague spec:** $b[h..i] \leq x$ and $x < b[i+1..k]$.

**Postcondition:** $b[h..i] \leftrightarrow x$ and $v < b[i+1..k]$

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

<table>
<thead>
<tr>
<th>$b$</th>
<th>$h$</th>
<th>$i$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

In this case, it's not possible to satisfy the postcondition.

**Truthify:** $b[h..i] \leftrightarrow x$ and $v < b[i+1..k]$

**Remove adjacent duplicates**

- **Precondition:** $b[0..n]$ is initial values in $b[0..n]$ but with adjacent duplicates removed
- **Postcondition:** $b[0..n]$ is initial values in $b[0..n]$ unchanged with no adjacent duplicates

**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

```java
Swap b[i] and b[j].
```

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

<table>
<thead>
<tr>
<th>$b$</th>
<th>$\leq v$</th>
<th>$&gt; v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 $x$, if $v$ is not in $b[h..k]$, it belongs between $b[i]$ and $b[i-1]$.

**Binary search:** Vague spec: Look for $v$ in sorted array segment $b[h..k]$.

Better spec:

- **Precondition:** $b[h..k]$ is sorted (in ascending order).
- **Postcondition:** $b[h..i] \leftrightarrow v$ and $v < b[i+1..k]$

Below, the array is in non-descending order:

<table>
<thead>
<tr>
<th>$b$</th>
<th>$h$</th>
<th>$i$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

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

**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?

<table>
<thead>
<tr>
<th>$b$</th>
<th>$0$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

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

$$\leq$$

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