CS469 Assignment 1

Tianye Zhao

February 10, 2019

Exercise 1

```
Create an integer array $A[100]$
**for** $i \leftarrow 0$ **to** $99$ **do**
    $A[i] \leftarrow 2 \times i + 1$
**end for**
$pointer \leftarrow A$
**while** $pointer$ **not** $= A.end + 1$ **do**
    **print** $*pointer$
    $pointer \leftarrow pointer + 1$
**end while**
$pointer \leftarrow A$
$counter \leftarrow 0$
**while** $pointer$ **not** $= A.end + 1$ **do**
    **if** $15 \leq *pointer \leq 65$ **then**
        $counter \leftarrow counter + 1$
    **end if**
    $pointer \leftarrow pointer + 1$
**end while**
**return** $counter$
```

Exercise 2

```
Create an integer array $A[10]$
**for** $i \leftarrow 0$ **to** $9$ **do**
    $A[i] \leftarrow$ user input
**end for**
$v \leftarrow$ user input
$pointer \leftarrow A$
$position \leftarrow 0$
**while** $pointer$ **not** $= A.end + 1$ **do**
    **if** $*pointer = v$ **then**
        **print** $position$
        **while** $pointer$ **not** $= A.end$ **do**
            $*pointer \leftarrow *(pointer + 1)$
        **end while**
        $*pointer \leftarrow 0$
    end if
    $pointer \leftarrow pointer + 1$
    $position \leftarrow position + 1$
end while
if $pointer = A.end + 1$ then
    print $position \leftarrow -1$
end if
$pointer \leftarrow A$
while $pointer$ not $= A.end + 1$ do
    print $*pointer$
    $pointer \leftarrow pointer + 1$
end while
```

Exercise 3

```
Given an integer array $A$
$pointer \leftarrow A$
$minValue \leftarrow A[0], maxValue \leftarrow A[0]$
$position \leftarrow 0, minPosition \leftarrow 0, maxPosition \leftarrow 0$
while $pointer$ not $= A.end + 1$ do
    if $*pointer < minValue$ then
        $minValue \leftarrow *pointer$
        $minPosition \leftarrow position$
    else if $*pointer > maxValue$ then
        $maxValue \leftarrow *pointer$
        $maxPosition \leftarrow position$
    end if
    $pointer \leftarrow pointer + 1$
    $position \leftarrow position + 1$
end while
return $minPosition, minValue, maxPosition, maxValue$
```

Exercise 4

```
Given an ascending integer array $A[N]$
$val \leftarrow$ user input
$pointer \leftarrow A$
while $pointer$ not $= A.end + 1$ do
    if $*pointer < val$ then
        $pointer \leftarrow pointer + 1$
    else if $*pointer \geq val$ then
        $pointer \leftarrow pointer - 1$
        $pointer1 \leftarrow A.end + 1$
        $pointer2 \leftarrow A.end$
        while $pointer1$ not $= pointer$ do
            $*pointer1 \leftarrow *pointer2$
        $pointer1 \leftarrow pointer1 - 1$
        $pointer2 \leftarrow pointer2 - 1$
      **end while**
      $*pointer \leftarrow val$
    **end if**
**end while**
**if** $pointer = A.end + 1$ **then**
    $*pointer \leftarrow val$
**end if**
```

Exercise 6

Linear search

```
Given an integer array $A$
$val \leftarrow$ user input
$pointer \leftarrow A$
$pos \leftarrow 0$
**while** $pointer$ **not** $= A.end + 1$ **do**
    **if** $*pointer = v$ **then**
        **return** $pos$
    **end if**
    $pointer \leftarrow pointer + 1$
    $position \leftarrow position + 1$
**end while**
**if** $pointer = A.end + 1$ **then**
    **return** $pos \leftarrow -1$
**end if**
```

Binary search

```
Given an ascending integer array $A$ **and** $val \leftarrow$ user input
$binarySearch(A, low, high, val)$
**if** $low > high$ **then**
    **return** $-1$
**end if**
$pointer \leftarrow A + low$
$mid \leftarrow round\big((low + high)/2\big)$
**if** $*(pointer + mid) = val$ **then**
    **return** $mid$
**else if** $*(pointer + mid) < val$ **then**
    $binarySearch(A, mid + 1, high, val)$
**else**
    $binarySearch(A, low, mid - 1, val)$
**end if**
```

We can rebuild the array as a binary comparison tree model whose root is the middle element in array. Then make the middle element of the lower half become the left child of root and the middle element of the upper half become the right child of root. Build the remaining in the same method.

From the root, the left or right subtree would be traversed depending on whether the target value is less or more than current node. Each time we only choose one path from two, it significantly eliminates the searching elements. Assume there are n elements in array, the worst case for binary search is traversing from root to one of the leaves, which would pass through $\log n + 1$ nodes. This is much less than linear search whose worst case is checking all elements n.

Exercise 7

```
Given an integer array $A$
Create an integer array $valueArray$ holds values in $A$
Create an integer array $counterArray$ holds counters for respective values in $valueArray$
$pointer \leftarrow A$
$pointer1 \leftarrow valueArray$
$pointer2 \leftarrow counterArray$
while $pointer$ not $= A.end + 1$ do
  if $*pointer$ not in $valueArray$ then
    $*pointer1 \leftarrow *pointer$
    $*pointer2 \leftarrow 1$
    $pointer1 \leftarrow pointer1 + 1$
    $pointer2 \leftarrow pointer2 + 1$
  else
    $pos \leftarrow$ position of $*pointer$ in $valueArray$
    $counterArray[pos] \leftarrow counterArray[pos] + 1$
  end if
  $pointer \leftarrow pointer + 1$
end while
$pos \leftarrow$ position of max value in $counterArray$
return $valueArray[pos]$
```