CS 180 Homework 3

February 26, 2015

1 Rectangles

(a) Design an $O(n \log n)$ algorithm that finds the outline of the rectangles.

```
Function mergeRect(array)
        If sizeof(array) is 1
                Return array[0]
        midpoint = sizeof(array)/2
        first = array[:midpoint]
        second = array[midpoint:]
        Return merge(mergeRect(first), mergeRect(second))

Function merge(arrayOne, arrayTwo)
        Let Output be an empty integer array
        For each i from 0 to sizeof(arrayOne)−1
                For each j from 0 to sizeof(arrayTwo)−1
                        Push arrayOne[i] to Output

Let I be the array of rectangle arrays
Sort array I by the first element of each rectangle array in ascending
Return mergeRect(I)
```

(b) Design an O(n) algorithm that, given an outline, finds a rectangle of maximal area that fits within the outline. Implement your algorithm with a single left-to-right scan through the outline data.

```
Let O be the array representing the outline
Let S, i be 0
While O[i+2] is not null
        S += (O[i+2] − O[i]) / O[i+1]
        i += 2
Return S
```

2 Interview Questions

(a) You are given a 3-pint container and a 5-pint container, and as much water as you want. Specify a sequence of filling and emptying steps that leave the containers holding exactly 7 pints of water.

The first step is to fill up the the 5-pint container and empty as much as possible into the 2-pint container. This leaves you with 3 pints and 2 pints in the two containers. The next step is to empty the full 3-pint container and pour the 2 pints from the 5-pint container into the 3-pint container. Finally, you fill the empty 5-pint container, leaving us with 2 pints in the 3-pint container and 5 pints in the 5-pint container for a total of 7 pints of water

(b) There are many problems like the one above; it is from the 1916 Stanford-Binet IQ test. A common interview question uses 3, 5, 4. Design an algorithm that, given three small integers like these as input, finds a sequence with the minimum number of steps.

```
Let a, b be the container sizes
Let z be the target
Let G be a directed graph consisting of one vertex (0, 0)
For each vertex in G
        Create an edge from (V1, V2) to (V1, V2 + V1 mod b) if not expl
        Create an edge from (V1, V2) to (V1, b) if not explored
        Create an edge from (V1, V2) to (V1, 0) if not explored
        Create an edge from (V1, V2) to (V1 + V2 mod a, V2) if not expl
        Create an edge from (V1, V2) to (a, V2) if not explored
        Create an edge from (V1, V2) to (0, V2) if not explored
        Mark (V1, V2) as explored
Let T be a Dijkstra's shortest path tree for G from (0, 0)
Use DFS on T until first coordinate + second coordinate is z
Return the stack used for DFS
```

(c) You are given an array A of n integers, and another integer z, and you want to determine whether the array contains two elements a and b such that $a + b = z$.

i. Give an algorithm that uses a min-heap and a max-heap to determine this in time $O(n \log n)$.

```
Let A be the array of integers
Let Min be a min-heap
Let Max be a max-heap
For each i from 0 to n-1
        Min.Insert(A[i])
        Max.Insert(A[i])

While FindMin(Min) < FindMax(Max)
        Sum = FindMin(Min) + FindMax(Max)
        If Sum > z
```

```
                ExtractMin(Min)
        Else if Sum < z
                ExtractMax(Max)
        Else
                Return true
Return false
```

ii. Give an algorithm that runs in time $O(n)$, assuming that A is given to you in sorted order.

```
Let A be the array of integers
Let i be 0
Let j be n-1
While i < j
        Sum = A[i] + A[j]
        If Sum > z
                j -= 1
        Else if Sum < z
                i += 1
        Else
                Return true
Return false
```

(d) You are given an array A of n integers (possibly negative) and you want to determine whether the array contains three elements a, b, and c such that $a + b + c = 0$. Give an algorithm that solves this problem in $O(n^2)$ time.

```
Let A be the array of integers
Use merge-sort to sort in ascending order
For i from 0 to n-1
        j = i+1
        k = n-1
        While j < k
                Sum = A[i] + A[j] + A[k]
                If Sum > 0
                        k -= 1
                Else if Sum < 0
                        j += 1
                Else
                        Return true
Return false
```

(e) You are given an array of size n containing every number in $0, 1, 2, ..., n$ except for one. Give an algorithm to find the missing number in time $O(n)$, using only 1 memory cell that has $\lceil 2 \log_2 n \rceil$ bits. (For example, when $n = 50000$, the cell has 32 bits, and can represent numbers from 0 to $2^{32} - 1$.)

```
Let I be the input array
Let CELL be the memory cell
CELL = 0
For each i from 1 to n−1
        CELL = CELL XOR I[i]
For each j from 0 to n−1
        CELL = CELL XOR j
Return CELL
```

3 Optimal Submatrix

(a) Find a maximal positive rectangular submatrix - i.e., a submatrix containing only positive values that has the most elements.

(b) Find a maximum sum rectangular submatrix - i.e., a submatrix whose elements have maximal sum. (Hint: This is a generalization of the 'maxsum' problem discussed at the start of this course.).

4 Going Beyond the Master Theorem

(a)