Homework #2

Illya Starikov

Date Due: February 15, 2016

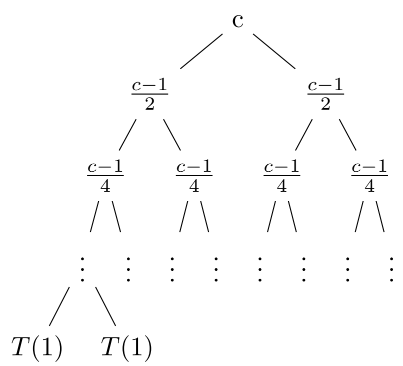
1 Programming Assignment

Code is attached. Binary search implementation is as follows.

```
extension Array where Element: Comparable {
    func binarySearch(key: Element) -> Int? {
        return binarySearch(key, low: 0, high: self.count - 1)
    }

    private func binarySearch(key: Element, low: Int, high: Int) -> Int? {
        if low > high { return nil }

        let middle = Int((low + high) / 2)

        if key == self[middle] {
            return middle
        } else if key > self[middle] {
            return binarySearch(key, low: middle + 1, high: high)
        } else {
            return binarySearch(key, low: low, high: middle - 1)
        }
    }
}
```

2 Recurrence Equation

2.1 Tower of Hanoi



The complexity of this tree is $O(\lg n)$, or alternatively, $O(2^n)$

2.2 Merge Sort

$$T(n) = 2\ T(\frac{n}{2}) + n \tag{1}$$

From the equation, we can see that

$$a = 2, b = 2, c = 1, f(n) = n \tag{2}$$

We observe that

$$f(n) \in \Theta(n^{\log_b a}) = \Theta(n^{log_2 2}) = \Theta(n) \tag{3}$$

It follows from Case #2 of the Master Theorem

$$\begin{aligned}
T(n) &\in \Theta(n^{\log_b a} \lg n) && (4)\\
&= \Theta(n^{\log_2 2} \lg n) && (5)\\
&= \Theta(n^1 \lg n) && (6)\\
&= \Theta(n \lg n) && (7)
\end{aligned}$$

Thus the recurrence relation $T(n)$ is in $\Theta(n \log n)$. QED.

3 Loop Invariant

```
Merge(A, p, q, r)
    leftIndex = q - p + 1
    rightIndex = r - q
    let L[1..left + 1] and R[1..right + 1] be new arrays
    for i = 1 to leftIndex
```

```
6           L[i] = A[p + i - 1]
7       for j = 1 to rightIndex
8           R[j] = A[q + j]
9       L[leftIndex + 1] = sentinel
10      R[rightIndex + 1] = sentinel
11      i = 1
12      j = 1
13      for k = p to r
14          if L[i] <= R[j]
15              A[k] = L[i]
16              i = i + 1
17          else A[k] = R[j]
18              j = j + 1
```

At the end of the for loops on 4-5, 6-7, new arrays will be created, holding the left and right half of the data in the passed array.

Initialization Initially we have two arrays, the left and right side of the original passed array. This holds trivially.

Maintenance During each iteration, we copy over the array's data.

Termination Upon termination, the Left array has the data from $[p + i - 1]$ and Right array has the data of $[q + j]$.

During lines 12 - 17, we merge the arrays to get a properly sorted array.

Initialization Initially we have the left and right arrays, unsorted, and the original array. This holds trivially.

Maintenance During the iterations of the array, we replace the data of the originally passed array with the smaller of the Left and Right array.

Termination Upon termination, we have a fully sorted array from $q...r$.

4 Quicksort

Let $\underline{\ }$ signify the pivot, || signify the wall.

1. || 4 2 7 6 3 5 $\underline{1}$
2. 1 || || 4 2 7 6 3 $\underline{5}$
3. 1 || 4 || 2 7 6 3 $\underline{5}$
4. 1 || 4 2 || 7 6 3 $\underline{5}$
5. 1 || 4 2 3 || 7 6 $\underline{5}$
6. 1 || 4 2 $\underline{3}$ || 5 || 7 6
7. 1 || || 4 $\underline{3}$ || 5 || 7 6

8. 1 || 2 || 4 $\underline{3}$ || 5 || 7 6
9. 1 || 2 || 3 || 4 || 5 || 7 $\underline{6}$
10. 1 || 2 || 3 || 4 || 5 || || 7 $\underline{6}$
11. 1 || 2 || 3 || 4 || 5 || 6 || $\underline{7}$
12. 1 || 2 || 3 || 4 || 5 || 6 || 7

5 Heapsort

5.1 Heap Representation

```
      7
     / \
   4     5
  / \    |
 1   2   3
```

```
    5
   / \
  4   3
 / \
1   2
```

```
  4
 / \
2   3
|
1
```

```
  3
 / \
2   1
```

```
2
|
1
```

```
1
```

5.2 Enumerated Steps

1. 7
2. 7 5
3. 7 5 4

4. 7 5 4 3
5. 7 5 4 3 2
6. 7 5 4 3 2 1