Jake Schinto

Home work 1

1. MAX(A, i, j)
   1. m = i
   2. for k = i+1 : j
   3. if A[k] > A[m]
   4. m = k
   5. return m

2. The loop invariant: At the start of each iteration m has the index for the max value of the subarray A[i...k-1]

Initialization: A[i...k-1] = A[i] has a trivial max value at index i stored into m

Maintenance: By the loop invariant, m is the index for the max of subarray A[i...k-1]. So, if A[k] is greater than A[m], k becomes the new max and stored into m. So, the loop invariant remains true for the next iteration.

Termination: For the loop to terminate k = j+1. So, by the loop invariant m must contain the index for the max value for A[i...j] and the algorithm is correct.

3.

step	cost	time
1	c_1	1
2	c_2	n
3	c_3	n-1
4	c_4	n-1
5	c_5	1

Let n = j - i

$$T(n) = (c_2 + c_3 + c_4)n + (c_1 + c_5 - c_3 - c_4)$$

4. SortByMax(A)

```
1. for k = A.length : 2
2.     temp = A[k]
3.     m = MAX(A, 1, k)
4.     A[k] = A[m]
5.     A[m] = temp
6. return A
```

5. Loop invariant: At the start of each iteration the subarray $A[1...k]$ will have values all being less than or equal to $A[k+1]$ if it exists and $A[k+1...n]$ be in sorted order

Initialization: $k = n$, so $A[k+1]$ does not exist yet and $A[1...k]$ contains the values of the unsorted array.

Maintenance: By the loop invariant all values in $A[1...k]$ are less than or equal to $A[k+1]$ if it exists. So, by finding the max value in subarray $A[1...k]$ and swaping the value into $A[k]$, it maintains the loop invariant that $A[1...k-1] \leq A[k]$ and that $A[k...n]$ is sorted because $A[k] \leq A[k+1...n]$ assuming $A[k+1...n]$ is sorted by the loop invariant. So, the loop invariant remains true for the next iteration.

Termination: The loop terminates when $k = 1$. By the loop invariant $A[1]$ must be $\leq A[2...n]$ and $A[2...n]$ must be sorted. Therefore, $A[1...n]$ must be in sorted order and the algorithm is correct.

6.

step	cost	time
1	C_6	n
2	C_7	$n-1$
3	C_8	$n-1$
4	C_9	$n-1$
5	C_{10}	$n-1$
6	C_{11}	1

$$T(n) = T_{SBM}(n) + \sum_{j=2}^{n} T_{MAX}(j)$$
$$T_{SBM}(n) = (C_6+C_7+C_8+C_9+C_{10})n + (C_{11}-C_7-C_8-C_9-C_{10})$$

$$T(n) = (C_6+C_7+C_8+C_9+C_{10})n + (C_{11}-C_7-C_8-C_9-C_{10}) + \sum_{j=2}^{n}\left((C_2+C_3+C_4)j + (C_1+C_5-C_3-C_4)\right)$$

7.
```
find-in-halves(A, i, j, K)
1.  mid = (j+i)/2
2.  if A[mid] == K
3.      return mid
4.  else if A[mid] > K and mid != i
5.      return find-in-halves(A, i, mid, K)
6.  else if A[mid] < K and mid != j
7.      return find-in-halves(A, mid, j, K)
8.  else
9.      if A[j] == K
10.         return j
11.     return -1
```

8. $k = 4$

1	2	3	4	5
2	3	5	7	11

$= A$

mid $= (5+1)/2 = 3$

$A[3] = 5 > K$, mid $!= 1$

return find-in-halves$(A, 1, 3, 4)$

1	2	3
2	3	5

mid $= (3+1)/2 = 2$

$A[2] = 3 < K$, mid $!= 1$

return find-in-halves$(A, 2, 3, 4)$

2	3
3	5

mid $= (3+2)/2 = 2.5 \rightarrow 2$

$A[2] = 3 < K$, mid $== 2 \rightarrow A[3] != K \rightarrow$ return -1

9.

steps	cost
1	$\Theta(1)$
2	$\Theta(1)$
3	$\Theta(1)$
4	$\Theta(1)$
5	$T(n/2)$
6	$\Theta(1)$
7	$T(n/2)$
8	$\Theta(1)$
9	$\Theta(1)$
10	$\Theta(1)$
11	$\Theta(1)$

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ T(n/2) + \Theta(1) & \text{else} \end{cases}$$