COMP3270 Algorithms, Midterm 2

Richard Chapman

Fall Semester, 2014

NAME SOLUTION

Directions The test is open book and open notes, but NOT open phone or open computer (e-reader or computer used solely as e-reader is ok). For each problem, show your work completely. Give reasons for all answers – this is how I give partial credit. **Each part of each question is worth 10 points, 100 total points)**

1. Is counting sort stable? If so, prove it; if not, prove that or show a counterexample.

Counting sort is stable. Elements are copied from A[] to B[] in an order that is same as they were in A[] in line 10-12 of the algorithm

for j = A.length downto 1

B[C[A[j]]] = A[j]

C[A[j]] --

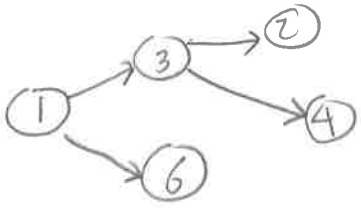
2. For the directed graph whose adjacency matrix is given below, draw the predecessor tree that would be generated by Breadth First Search. You can assume that the adjacency list for a given node is explored in order of increasing number of the head vertex for each edge (That is,

if vertex 7 has edges to vertices 10,3,2, and 5, those will be explored in the order 2,3,5,10). As in 22.2(c) in the textbook, if there is an entry in row 1, col 4, that means there is a directed edge from vertex 1 to vertex 4.

	1	2	3	4	5	6
1	1		1			1
2		1	1			
3		1	1	1		
4				1		
5	1			1		
6						1

* start at vertex 1

~~1, 6, 2, 4~~



3. What is the running time of heap sort on an array of length n that is already sorted in increasing order?

$O(n \lg n)$ – the ordered array is heapified, then max elt extracted n times, each time $\Theta(\lg n)$.

4. How could you implement a (non-priority) queue (FIFO) using a priority queue? Implement the ENQUEUE(n) and the n = DEQUEUE() operations using the priority queue operations defined in section 6.5 of the textbook.

Add a global variable key

Implement MIN-HEAP

key = 0 initially, on attribute key

```
ENQUEUE(x)
    MIN-HEAP-INSERT(A, x)
    key ++
DEQUEUE()
    return HEAP-EXTRACT-MIN(A)
```

5. Give pseudocode to modify quick sort to sort in decreasing rather than increaing order?

only PARTITION needs to be modified

```
PARTITION(A, p, r)
    x = A[r]
    i = p - 1
    for j = p to r-1
    if A[j] ≥ x
        i = i+1
        exchange A[i] with A[j]
exchange A[i+1] with A[r]
return i+1
```

6. Show that $\Theta(nlg(n))$ is a lower bound for the worst-case running time of a comparison sort.

Proved in text book - theorem 8.1.

7. In Chapter 8 we looked at several sorting algorithms that take linear time: counting sort, radix sort, and bucket sort. Suppose someone claims to discover a sequential (i.e. not parallel) algorithm to sort in $O(lgn)$ time. Is that possible? Why or why not?

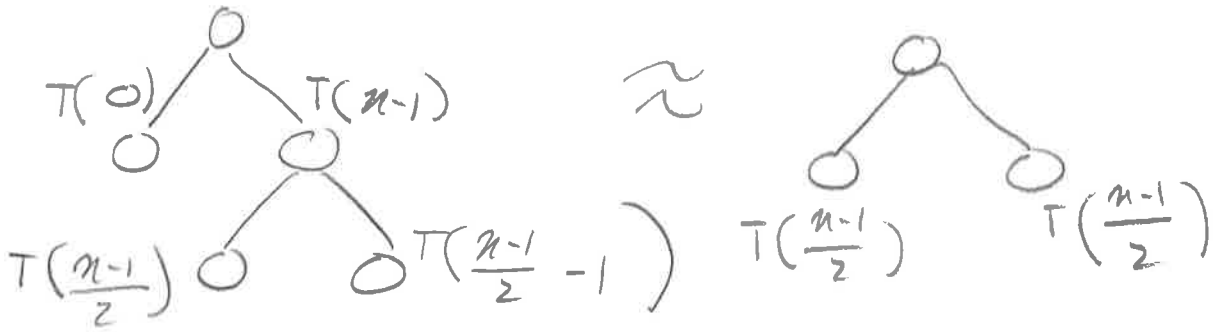
Not possible - it takes $O(n)$ just to read in the input - you could not reverse an array that was backward in $O(\lg n)$

8. If, for a given input data set, the PARTITION routine in quicksort gave you a partition into an n-1 element subarray and a 0 element subarray every other time it was called, and two (n-1)/2-sized subarrays the

other times it was called, what is the running time of quick sort with that data? Justify your answer.

$O(n \lg n)$

This analysis is on pp. 176-7 of text book.



depth is $O(2 \lg n) = O(\lg n)$