# COMP4200 Formal Languages, Sample Midterm

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 12.5 points, 100 total points)**

1. For an arbitrary comparison sort, give a close lower bound on the height of the decision tree and explain why it is a bound?

For a list of $n$ items there are $n!$ possible orderings. That means the decision tree will have $n!$ leaves. Since it is a binary tree, its height must be at least $\log_2(n!)$
we know $\log_2(n!) = \boxed{\Theta(n \lg n)}$ from Chapter 3.

2. Give an example of a comparison sorting algorithm whose big-$\Theta$ running time complexity is optimal in the worst case and the average case. Justify your answer.

heap-sort is $\boxed{\Theta(n \lg n)}$ average case and worst-case.
Since $\Omega(n \lg n)$ is a lower bound for comparison sorts,
so it is optimal.

3. Suppose the input to quicksort is such that 3/4 of the time you get one subarray being n/4 of the elements and the other being 3n/4, and the other 1/4 of the time you get a partitioning into n/2 and n/2 size sub-arrays. What is the average running time of quicksort on that kind of input?

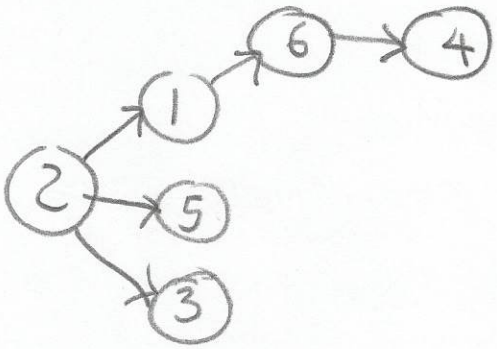
For both cases $T(n) = \Theta(n \lg n)$ by argument on p. 174 for 9/10-case. Here it doesn't matter that one case happens 3/4 of time and the other 1/4 – you get $\Theta(n \lg n)$ either way, hence $\boxed{\Theta(n \lg n)}$ overall.

4. In randomized quicksort, how many calls to the random number generator are made in the worst case? In the best case? (Worst and best refer to the total running time of the algorithm). Give your answer in big -$\Theta$ notation and justify your answer.

In worst case $T(n) = T(n-1) + \Theta(n)$ and a call to random number generator will be made in each call to PARTITION. PARTITION is called a number of times equal to depth/height of recursion tree. Worst case is $\boxed{\Theta(n)}$. Best case PARTITION is called $2^k$ times at depth $k$, and $k = \Theta(\lg n)$ giving $\boxed{\Theta(n^2)}$ in best case also.

5. For the graph whose adjacency list representation is show in Figure 1, show the predecessor subgraph for that graph, after breadth first search

starting at vertex 2.




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

$\Theta(n \lg n)$ - already sorted in increasing order is worst-case for heapsort in terms of # of swaps

7. How could you implement a stack (LIFO) using a priority queue? Implement the PUSH(n) and the n = POP() operations using the priority queue operations defined in section 6.5 of the textbook.

Define a global variable key = 1
a global array A

```
PUSH(n)
  MAX-HEAP-INSERT(A, key++)

POP()
  return HEAP-EXTRACT-MAX(A)
  key--
```

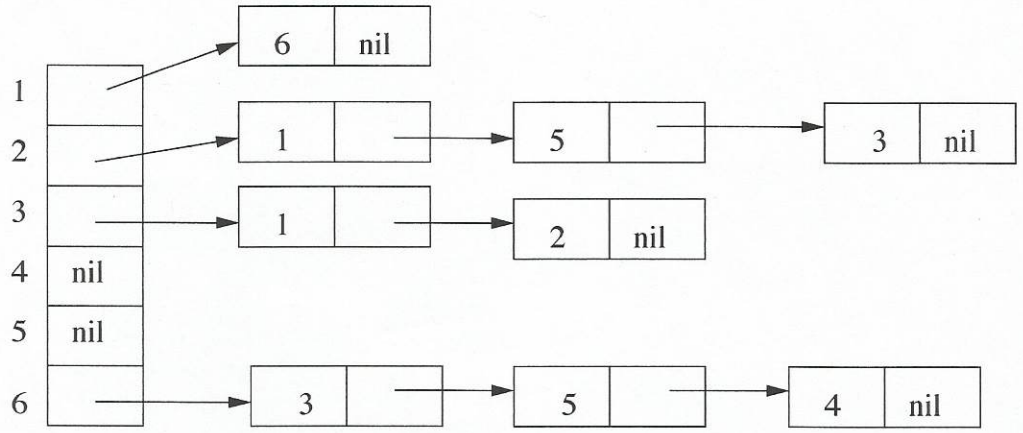



Figure 1: Graph for question 5

8. Suppose radix sort were modified to sort on the most significant bit (MSB) of the binary representation of the number, then recursively call itself to sort those numbers with a zero in the MSB, and a 2nd recursive call to sort those with a 1 in the MSB. Analyze in terms of time and space complexity.

$T(n) = 2T(n/2) = \Theta(n)$ fine by master theorem

But a large amount of space is needed to store intermediate results

$\Theta(n)$ for each level of recursion

there are $\lg n$ levels of recursion

hence $\Theta(n \lg n)$ space is needed