

**CS 146 Midterm Exam 2****Name:** \_\_\_\_\_**Grade:**

The test will be open book, open notes, 75 minute time limit. No electronic devices allowed (except a music player whose buttons you don't push). Please write your answers on the exam sheet. Ten problems, 10 points per problem. Partial credit given only when the problem has parts (a,b,c, etc.)

## 1. Analyzing recursive code using recurrence relations

Professors Howard, Fine, and Howard have proposed the following sorting algorithm:

```

StoogeSort(A, i,j)
// sort A[i]...A[j]
{ if ( A[i] > A[j])
    swap A[i] and A[j];
  if(i + 1 >= j)
    return;
  k = (j-i + 1) / 3;
  StoogeSort(A,i,j-k); // sort the first two-thirds
  StoogeSort(A,i+k,j); // sort the last two-thirds
  StoogeSort(A,i,j-k); // sort the first two thirds again
}

```

(a) Write a recursion relation for the running time  $T(n)$  of *StoogeSort*.

$$T(n) = 3 T(2n/3) + c, \quad \text{or } T(n) = 3 T(2n/3) + O(1)$$

(b) Exhibit the answer you obtain by solving that recursion relation, using O-notation for your answer. For this part, you can leave an unevaluated numerical expression in the answer since calculators aren't allowed.

The answer is,  $O(n \text{ to the } \log_{3/2} 3)$ , which I can't typeset in Word, because it won't handle a subscript in a superscript.

Now I give not just the answer but the whole solution. All the nodes in the recursion tree contribute  $c$  steps (corresponding to the first five lines of code). At the  $k$ -th level there are  $3^k$  nodes, for a total of  $3^k c$  at the  $k$ -th level. There are  $\log_{3/2} n$  levels. So the grand total is

$$T(n) = c[1 + 3^2 + 3^3 \dots + 3^p] \text{ where } p = \log_{3/2} n$$

That series can be summed:

$$T(n) = c(3^P - 1) / (3 - 1) = O(3^P).$$

Since I'm writing this in Word, I can't have a subscript inside a superscript. So I'll say it in words. 3 to the  $\log_{3/2} n = n$  to the  $\log_{3/2} 3$ . So the answer is,  $O(n \text{ to the } \log_{3/2} 3)$ .

If you use the Master Theorem, case 1 applies and you get the same answer.

- (c) Is this slower or faster than Merge Sort ?  
Slower or faster than  $O(n^2)$ ?

Now (however we got here, recursion tree or Master Theorem) we need to do the arithmetic to work out what that power is. It is

$$\log_{3/2} 3 = (\lg 3) / \lg(3/2) = (\lg 3) / (\lg 3 - \lg 2) = (\lg 3) / (\lg 3 - 1) = 1.585 / 0.585 = 2.71$$

Since that's more than 2, *StoogeSort* is even slower than insertion sort, let alone *MergeSort*, as the time for insertion sort is  $O(n^2)$  and for *MergeSort* is  $O(n \lg n)$ .

2. *Various kinds of trees.* We studied binary search trees, red-black trees, and heaps.

- (a) What relationship must the data values L and R of the children of a node have to the data D at that node?

For a binary search tree:  $L \leq D \leq R$   
For a red-black tree:  $L \leq D \leq R$   
For a max-heap:  $L \leq D$  and  $R \leq D$

- (b) What relationship exists between the length M of the longest branch and the length S of the shortest branch in a tree? Give the most accurate possible relationship.

For a binary search tree: none  
For a red-black tree:  $M \leq 2S$   
For a max-heap:  $M \leq S+1$

- (c) What relationship exists between the number of nodes N and the length S of the shortest branch?

For a binary search tree: none  
For a red-black tree:  $N \leq 2^{2S}$   
For a max-heap:  $N \leq 2^{S+1}$

3. *Merge sort.* The following array is to be sorted:

8	3	9	12	1	17	33	15	19	5	11	13	22	7	6	10
---	---	---	----	---	----	----	----	----	---	----	----	----	---	---	----

(c) What will the array look like after the return from the first recursive call? *The first half will be sorted and the second half unchanged.*

1	3	8	9	12	15	17	33	19	5	11	13	22	7	6	10
---	---	---	---	----	----	----	----	----	---	----	----	----	---	---	----

4. *Quick sort.* The following array is to be sorted (it's the same as the one above):

8	3	9	12	1	17	33	15	19	5	11	13	22	7	6	10
---	---	---	----	---	----	----	----	----	---	----	----	----	---	---	----

(a) if *quickSort* is used to sort this array, what will the array look like just before the first recursive call? *Partition will be run with pivot element 10.*

8	3	9	1	5	7	6	10	19	12	11	13	22	17	33	15
---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----

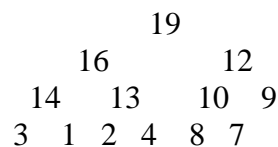
(b) What will the array look like just after the first recursive call returns? *The part before 10 is sorted.*

1	3	5	6	7	8	9	10	19	12	11	13	22	17	22	15
---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----

5. *Heap data structure.* Here is the array representation of a heap.

19	16	12	14	13	10	9	3	1	2	4	8	7			
----	----	----	----	----	----	---	---	---	---	---	---	---	--	--	--

Draw this heap as a tree: (It's too hard to get the diagonal lines in Word)



6. *Heap algorithms.* The following array is not a heap, but the heap properties are violated only once (involving two numbers).

19	16	12	14	13	10	9	3	1	2	4	8	7	23		
----	----	----	----	----	----	---	---	---	---	---	---	---	----	--	--

(a) Which two numbers (parent and child) in the array violate the heap properties?

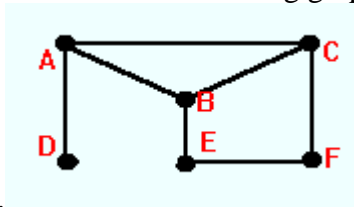
23 and its parent 9.

(b) After calling MAX-HEAPIFY to correct that violation, the resulting array will be a heap. Show it in array form:

23	16	19	14	13	10	12	3	1	2	4	8	7	9		
----	----	----	----	----	----	----	---	---	---	---	---	---	---	--	--

8. *Depth-first and breath-first search, conceptual understanding.*

Consider the following graph.



(a) If this graph is searched by *recursive* depth-first search starting from C, and if neighbors are searched in alphabetical order, what will be the order in which the nodes are visited?

*Answer:* C, A, B, E, F, D

(b) If the same graph is searched by breadth-first search starting from C, and if neighbors are searched in alphabetical order, what will be the order in which the nodes are visited?

*Answer:* C, A, B, F, D, E

9. In the Bob's Way Home problem that we studied as an example of breadth-first search, we returned the length of the shortest path. But that wouldn't help Bob get home: he needs the (or "a") path itself. What has to be added to the code to compute the best path as well as its length? Answer this question by adding *one line* of code somewhere where a blank is left in the code shown below. You may reference an array that hasn't been declared.

```

// next has just been dequeued
for(int k=0;k<4;k++)
  { u = next.i + dy[k];
    v = next.j + dx[k]; // so (u,v) is the k-th neighbor of next
    if(u < 0 || u >= n || v < 0 || v >= m || map[u][v] == 'X')
      continue; // (u,v) is illegal
    if(dd[next.i][next.j] + 1 < dd[u][v])
      { dd[u][v] = dd[next.i][next.j]+1;
        pred[u][v] = next; // this is the answer
      }
    if(visited[u][v] == false)
      { visited[u][v] = true;
        s.add(new Node(u,v)); // enqueue the unvisited neighbors
      }
  }
}

```

10. *Recognizing that a problem involves a graph.* No doubt you often use Google Maps or Mapquest. Have you ever thought about how it works? Suppose we define a graph  $M$  this way: we just take vertices to be locations, edges to be  $(A,B)$  if you can drive from  $A$  to  $B$ , and weights to be the driving time.

The problem with this is that graph  $M$  is very large, as every address in the US is a vertex, and there are hundreds of millions of those, and every pair of addresses is an edge.

Let's try to solve that problem this way. When the user requests directions from  $A$  to  $B$ , we will generate a *custom graph*  $G$  just for this problem, which should be of a reasonable size. The weights will still be driving times, but the edges and vertices, for example if  $A$  and  $B$  are both in Northern California, won't need to include locations in Boston.

(a) The vertices of  $G$  should be: *locations  $A$  and  $B$ ; all freeway access points in the same city (or for large cities, part of the city) as  $A$  or  $B$ ; all freeway intersections. Include all street intersections in same city or part of city as  $A$  or  $B$ .*

(b) The edges of  $G$  should be: *pairs of vertices. Hopefully this is now down to a manageable number. Keeping in mind the example where you try to go from Santa Clara to Walnut Creek, so you must get from 880 to 680, you can see that cutting it down more will be at least a little tricky.*

(c) Once  $G$  is generated, what algorithm among those we have studied could we use to generate directions for the user? *Dijkstra.*