CISC 320 Midterm Exam

Wednesday, Mar 25, 2015

Name: ________________________________

There are 19 questions. The first 15 questions count 4 points each. For the others, points are individually shown. The total is 100 points.
Multiple Choice: write the letter of the best choice in the blank.
Short Answer: write the short answer in the blank.

1. _c_ A binary heap is stored in an array. If the array indexing begins at 1, for the node at location i (assuming i has two children), the left child is placed
   (a) at location ceil($i/2 + 1$)
   (b) at location floor($i/2 + 1$)
   (c) at location $2i$.
   (d) none of the above

2. _d_ Algorithm B manipulates an array of n items. It requires $\Theta(2^n)$ time. For arrays of 20 items algorithm B takes about 1 second. On an array of 352 items how long do you expect algorithm B to take? Hint: $2^{332} \approx 10^{100}$.
   (a) about 100 seconds.
   (b) about 1000 seconds.
   (c) about 1000000 seconds.
   (d) more time than the age of the universe.

3. _c_ Suppose $f(n) = \Theta(n)$. Let $g(n) = f(n) + n$. Which is the strongest statement you can make?
   (a) $g(n) = O(n)$
   (b) $g(n) = \Omega(n)$
   (c) $g(n) = \Theta(n)$, i.e. both (a) and (b) are true.
   (d) None of the above.

4. _b_ Let $g(n)$ be an unknown function which is positive for all positive n. Suppose $f(n) = n * g(n)$. Which is the strongest statement you can make?
   (a) $f(n) = O(g(n))$
   (b) $f(n) = \Omega(g(n))$
   (c) $f(n) = \Theta(g(n))$, i.e. both (a) and (b).
   (d) None of the above.

5. _c_ Karatsuba's algorithm is a divide and conquer method to multiply long integers (or polynomials). The idea is to chop each of the two input numbers in half, commbine the half size numbers with additions/subtractions, and thus reduce the problem to several multiplications of numbers half. How many multiplications of numbers half as long?
(a) 1, (b) 2, (c) 3, (d) 4

6. _a_ The data structures used by Prim's and Kruskal's algorithms are

(a) priority queue and union-find, respectively.
(b) hash table and priority queue, respectively.
(c) union-find and priority queue, respectively.
(d) dictionary and hash table, respectively.

7. What does it mean to topologically sort a directed graph? Put the vertices on a line so that all the edges go left to right. (It is not possible if there is a cycle.)

8. _c_ Which category of graph can be topologically sorted?

(a) bipartite graphs
(b) complete graphs.
(c) DAGs
(d) none of the above.

9. _c_ What does the function mystery(G) do (for undirected graph G)?

```
bool mystery(graph *G) {
    int n = G->nvertices;
    initialize_search(G).
    bfs(G, 1);
    for (i = 2; i <= n; ++i)
    {   if (not discovered[i]) return false;  }
    return true;
}
```

(a) Determine if G is bipartite.
(b) Determine if G is a DAG.
(c) Determine if G is connected.
(d) Determine if G has n-1 components.

10. _a_ What happens in the above question if bfs is replaced by dfs?

(a) Same thing (every time).
(b) Different thing (sometimes, at least).
(c) Never terminates.
(d) Crashes

11. _c_ Which is true?

(a) Prim's's algorithm can be made to find maximal weight spanning trees just by reversing the signs of the edge weights.

(b) Kruskal's's algorithm can be made to find maximal weight spanning trees just by reversing the signs of the edge weights.

(c) Both A and B

(d) none of the above.

12. _a_ A (min) binary heap is an effective implementation of the priority queue interface (which consists of the functions insert() and extractMin()). A binary heap of n items can be build by a sequence of n insert(x) operations, but there is a faster way to heapify n items all at once. How fast is it?

    (a) $\Theta(n)$
    (b) $\Theta(n\log(n))$
    (c) $\Theta(n/\log(n))$
    (d) $\Theta(n^2)$

13. _c_ Which of the following is *not* an invariant maintained in the balanced binary search tree system called 2-3-4 trees?

    (a) All leaves are at the same depth.
    (b) Every internal node has 2, 3, or 4 children.
    (c) Every internal node contains 2, 3, or 4 keys.
    (d) The key at each node is no less than any key in the tree rooted at it's left child and no greater than any key in the tree rooted at it's right child.

14. _d_ Let G be an undirected graph having 4 components. What is the smallest number of components the graph may have after adding a single edge to the graph?

    (a) 0, (b) 1, (c) 2, (d) 3

15. _b_ Consider directed graphs having 4 strong components. Considering all possible weak connections among the components, what is the smallest number of strong components such a graph might have after adding a single directed edge?

    (a) 0, (b) 1, (c) 2, (d) 3

16. (10 points) Fill in the start and end times recorded by the depth first search algorithm on this graph when starting at node a. Write the start time for the visit to the left of each node and the end time to the right of the node. Assume the neighbors adjacent to a given node are listed (and visited) in alphabetical order.

```
     ---->  2 b 5
    /         ^  \--> 3 e 4
   /          |  /-->
1 a 12 -->  6 c 9
  \           |  \--> 7 f 8
   \          v  /-->
    ---->  10 d 11
```

17. (10 points) Kruskal's MST algorithm uses a dynamic disjoint set system called union-find. Union(a,b) joins the sets containing a and b. Find(a) returns the root (CEO) of the set containing a. Assume the sets are maintained as trees with each node v having a parent pointer $boss[v]$. The root node of a set is identified by $boss[v] = v$. Weights are stored in $wt[v]$, with initial value $wt[v] = 1$ for each v. [Note: find function = ceo function, union function = merger function.]

   (a) (5 pts) By filling in the gaps below, implement union (merger) using the union by weight heuristic. Pseudocode is acceptable. Assume a and b are ceo's.

```
void union (vertex a, vertex b) {
  if (wt[a] < wt[b]) {
      boss[a] = b;
      wt[b] = wt[b] + wt[a];




  } else {
      boss[b] = a;
      wt[a] = wt[a] + wt[b];




  }
}
```

   (b) (4 pts) Implement find (ceo) using the path compression heuristic. Here is the primitive version.

```
void find (vertex a)
{ if (a == boss[a] return a; else return  boss[a] =   find(boss[a]); }
                                         -----------
```

   (c) (1 pt) With those two heuristics, a sequence of n union and find operations runs in O($n$log(n)) time. Actually it is better than that. Describe a better big-O bound for this. $O(n \log *(n))$

18. (10 points) (a) Give a linear time algorithm to compute the chromatic number of graphs where each vertex has degree at most 2. Example graph with degrees no more than 2:

```
a    b--c   d--e--f--g  h--i  l--m
                        |  |   \/
                        j--k    n
```

One possibility: Run the connected components algorithm. If there are no edges charomatic number is 1.
If there are only paths and even cycles, chromatic number is 2.
If any component is an odd cycle (like l,m,n) it requires 3 colors, chromatic number is 3.

In detail: Here m is O(n). Thus the components algorithm runs in O(n) and constructs array c of size n, containing a component number for each vertex. Run through c[] to add up, say in a second array cs[], how many times each component number occurs. This costs O(n). Run through cs[] checking for odd entry greater than 1. Again O(n).

(b) Must such graphs be bipartite? Give proof or counterexample.

```
a----b
 \  /
  \/         requires 3 colors (bipartite = colorable with 2 colors)
  c
```

19. (10 points) Do exactly one of A, B, C. If you work on more than one, cross out all but one answer. We will look at ONLY the first non-crossed-out answer. Grading is based on correctness, clarity, thoroughness of answer.

A. A tournament is a directed graph formed by taking the complete undirected graph and assigning arbitrary directions on the edges, i.e., a graph G = (V, E) such that for all u,v in V, exactly one of (u, v) or (v, u) is in E. Show that every tournament has a Hamiltonian path, that is, a path that visits every vertex exactly once.

B. Suppose we are given the minimum spanning tree T of a given graph G (with n vertices and m edges) and a new edge e = (u, v) of weight w that we will add to G. Give an efficient algorithm to find the minimum spanning tree of the graph G + e. You may assume that T is rooted and given by parent pointers. Your algorithm should run in O(n) time to receive full credit.

C. Design a data structure that can perform a sequence of m union and find operations on a universal set of n elements, consisting of a sequence of all unions followed by a sequence of all finds, in time O(m + n).

Solutions: Discuss in class.