Tentamensdatum: 2021-03-18, 09:00 - 13:00
Tillåtna hjälpmedel: Inga
Totalt antal uppgifter: 7st, 100p
Jourhavande lärare: 0920 / 49 20 44

1. Insert numbers $38, 23, 13, 28$,and 18 *in that order* (*that is*, *first insert* 38, *then insert* 23, $\cdots$) into each of the following data structures (initially empty):

   (a) a min-heap. Show the array that represents the min-heap obtained. **(4p)**

   (b) a hash table of size 10 with the hash function $h(x, i) = (x \bmod 10 + i) \bmod 10,\ i = 0, 1, \cdots, 9$. Draw the hash table. **(4p)**

   (c) a binary search tree. Draw the binary search tree. **(4p)**

2. Given a binary tree T on n elements, design a linear time algorithm to determine whether or not T represents

   (a) a max-heap on these elements. **(4p)**

   (b) a binary search tree on these elements. **(4p)**

3. Given a (weighted) graph G with the adjacency-matrix representation:

	a	b	c	d	e
a	–	4	5	2	–
b	4	–	–	–	3
c	5	–	–	3	2
d	2	–	3	–	–
e	–	3	2	–	–

   (a) Draw this graph. **(3p)**

   (b) Draw a depth-first search *tree* starting at vertex a. **(3p)**

   (c) Apply each of the following algorithms to find a minimum spanning tree of G, starting at vertex a. You do not need to show how the algorithm work. Just list the edges *in the order* that they are added to the minimum spanning tree by the algorithm.

      i. Prim's algorithm. **(4p)**

      ii. Kruskal's algorithm. **(5p)**

4. Short Answer Questions.

   No justification is required. For *True/False* questions, a statement is *True* only if it is true in all cases while it is *False* if it is not true in some case.

   (a) $200 \cdot n^2 + 9 \cdot \frac{n^3}{\log n} - 1 + 12 \cdot n = \Omega(n^3)$. True of False? **(3p)**

   (b) Suppose that a graph $G = (V, E)$ is represented by its adjacency matrix. Give the running time of Prim's minimum spanning tree (in $\Theta(\cdot)$ notation). **(3p)**

   (c) In a hash table of size m with n elements in which collisions are resolved by chaining, a search operation takes average-case constant time, under the assumption of simple uniform hashing. True of False? **(3p)**

(d) Give the worst-case asymptotic time complexity (in $\Theta(\cdot)$ notation) of the following algorithm, in terms of the input size n. **(5p)**

```
def f(n):
    if n < 2: return 0
    count ← 0
    if n is even:
        for i = 1 to n log n: count ← count + 10
    else:
        for i = 1 to n log^2 n: count ← count + 1
    return count
```

You may assume that addition and assignment take constant steps.

(e) Give a recurrence relation for $T(n)$, the worst-case running time of the following algorithm, where n is the input size. **(5p)**

```
def f(n):
    if n ≤ 1: return 0
    count ← 0
    for i = 1 to 2^n: count ← 1 + f(n − 1)
    return count
```

You may assume that addition and assignment take constant steps.

(f) Dijkstra's algorithm for the single-source shortest-paths problem works correctly on any weighted, directed graph with no negative-weight cycles. True or False? **(3p)**

(g) Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles. Suppose that the number of edges on any single-source shortest path from the source s is at most k. Then, we can terminate Bellman-Ford after k passes and return the shortest paths. True or False? What is the running time (in $\Theta(\cdot)$ notation)? **(5p)**

(h) Consider binary search trees on n elements. The difference between the heights of any red-black tree and any AVL tree is at most $c \cdot \log n$, where $c > 0$ is a constant. True or False? **(3p)**

5. Multiple-Choice Questions.

Each of the following questions has exactly *one* correct answer and is worth 3 points. An incorrect answer will give -1 point. An answer with more than one choices will receive -1 point as well. The complete assignment (1)-(5) will give no less than 0 points.

(1) The worst-case run time for mergesort to sort a sorted array of length n is

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

(2) Which data structure is used in breadth-first search of a graph to hold nodes?

a. tree
b. queue
c. stack
d. heap

(3) Given a weighted directed acyclic graph $G = (V, E)$ represented by its adjacency lists, we can compute shortest paths from a single source in $\Theta(V + E)$ time by

a. doing a breadth-first search on G.
b. running Dijkstra's algorithm on G.
c. running Bellman-Ford's algorithm on G.
d. relaxing the edges of G according to a topological sort of its vertices.

(4) Let A be an algorithm that solves a given problem of size n in $\Omega(n^3)$ in the worst case. Which of the following statements is true?

a. A takes at most $c \times n^3$ time on every input of size n, where $c > 0$ is a constant.
b. A takes at most $c \times n^3$ time on some input of size n, where $c > 0$ is a constant.
c. A takes at least $c \times n^3$ time on every input of size n, where $c > 0$ is a constant.
d. A takes at least $c \times n^3$ time on some input of size n, where $c > 0$ is a constant.

(5) Which of the following statements is *NOT* true about sorting algorithms?

a. Heapsort is an in-place algorithm.
b. Countingsort is stable.
c. Insertionsort runs in linear time in the average case.
d. Bucketsort runs in linear time in the average case.

When presenting an algorithm, a clear and complete high-level description will suffice. Code is not required.

6. A binary tree is *perfect* if both the left and right subtrees of any node in the tree are perfect and the sizes of these two subtrees differ by at most 5. Design a worst-case linear-time algorithm for checking whether a given binary tree is perfect or not. **(10p)**

7. Given an array $A = \langle a_1, a_2, \cdots, a_n \rangle$ of length n that contains k $(0 \leq k \leq n)$ *bad* elements and $n - k$ *good* elements in some arbitrary order. The problem is to determine which elements are *bad* using operations *OR*. The *OR* operator can take any number of elements and return *True* if and only if one or more of its operands is *bad*. Design a worst-case $O(k \log \frac{n}{k})$-time algorithm to solve this problem. Analyze your algorithm. The complexity is measured by the number of *OR* operations used and other common operations. **(10p)**

For example, $A = \left\langle \begin{array}{|c|c|c|c|c|} \hline a_1 & a_2 & a_3 & a_4 & a_5 \\ \hline good & bad & bad & good & bad \\ \hline \end{array} \right\rangle$. Your algorithm should return a_2, a_3, and a_5.