Tentamensdatum: 2020-04-03, 09:00 - 13:00
Tillåtna hjälpmedel: Engelsk-svensk ordbok
Totalt antal uppgifter: 9 st, 100p
Jourhavande lärare: 0920 / 49 20 44

1. Heaps.

   (a) Show the array that represents the max-heap obtained by inserting $4, 11, 7, 2, 8$ *in turn* into an initially empty max-heap. **(5p)**

   (b) Show the array that represents the data structure after executing a delete-max operation on the above heap. **(5p)**

   (c) A max-heap of size n can be constructed as follows. If $n \leq 2$, build the heap by some method. Otherwise, one can first two max-heaps each of size $\frac{n-1}{2}$ *recursively*. After that, these two heaps are merged together with a singleton element into a max-heap on n elements; which costs $\Theta(\log n)$ comparisons in the worst case. Find the recurrence that computes the number of comparisons used by this algorithm in the worst case. You may assume that $n = 2^h - 1$ for some positive integer h. **(6p)**

2. Hash tables.

   (a) Show the result of inserting the numbers $10, 20, 4, 30$ *in that order* into an initially empty hash table of size 10 with the hash function $h(x, i) = (x \bmod 10 + i \cdot (1 + x \bmod 9)) \bmod 10$, $i = 0, 1, \cdots, 9$. **(5p)**

   (b) Assume that the input keys to a hash table are positive integers. Explain why the following functions are unsuitable to be hash functions.

      i. $h(x) = x$ **(2p)**

      ii. $h(x) = 2(x \bmod \lfloor \frac{m}{2} \rfloor)$, where m is the size of the hash table. **(3p)**

3. Let $f(n)$, $g(n)$ and $h(n)$ be positive integer functions. Suppose that $f(n) = O(h(n))$ and $g(n) = O(h(n))$. Which of the following are true? *No justification is needed.* **(4p)**

   (a) $f(n) = O(g(n))$

   (b) $\frac{f(n)}{g(n)} = O(1)$

   (c) Either $f(n) = O(g(n))$ or $g(n) = O(f(n))$ is true.

   (d) If the running time of an algorithm A is $f(n)$ in the average case, then the worst-case complexity of A is $O(n \times f(n))$.

4. Given a graph G with its adjacency matrix:

	a	b	c	d	e
a	–	5	2	–	–
b	–	–	–	2	–
c	1	–	–	5	–
d	–	6	–	–	3
e	3	–	4	–	–

   (a) Draw this graph and its adjacency list representation. **(5p)**

   (b) Draw a depth-first search *tree* starting at vertex e. Indicate the discovery and finishing times for each vertex. **(5p)**

(c) Execute Dijkstra's algorithm on G, starting at source node e. Show the progress of the algorithm by filling the following table. **(5p)**

Shortest-path estimates:

	e	a	b	c	d
initialization	0	∞	∞	∞	∞
stage 1					
stage 2					
stage 3					
stage 4					

5. Design algorithms (*based on different design techniques*) to merge k sorted lists into one sorted list of length n in $O(n \log k)$ time in the worst case, where n is the total number of elements in all the input lists.

   (a) Use divide and conquer technique. **(5p)**

   (b) Use data structures. **(5p)**

6. Given an unsorted array $A = \langle a_1, a_2, \cdots, a_n \rangle$ of n distinct elements, find an index i ($1 \leq i \leq n$) such that $a_{i-1} < a_i$ and $a_i > a_{i+1}$, where $a_0 = a_{n+1} = -\infty$. Your algorithm should run in $O(\log n)$ time in the worst case. Argue why your algorithm is correct. **(10p)**

7. Construct a balanced binary search tree on n distinct integers in the range $0, 1, \cdots, n^2 - 1$. Your algorithm should run in linear time in the worst case. **(10p)**

8. Given two sequences of numbers $A = \langle a_1, a_2, \cdots, a_n \rangle$ and $B = \langle b_1, b_2, \cdots, b_n \rangle$. Let sum_A and sum_B be the sum of all numbers in A and B, respectively. The problem is to determine whether or not A contains some element a_i and B contains some element b_j such that $sum_A - a_i + b_j = sum_B - b_j + a_i$. For example,

   - *Example 1:* $A = \langle 7, 1, 3, 5 \rangle$ and $B = \langle 3, 2, 4, 1 \rangle$.
     Thus, $sum_A = 7 + 1 + 3 + 5 = 16$ and $sum_B = 3 + 2 + 4 + 1 = 10$.
     Since $sum_A - a_4 + b_2 = 16 - 5 + 2 = 13$ and $sum_B - b_2 + a_4 = 10 - 2 + 5 = 13$, the algorithm should return (a_4, b_2).
   - *Example 2:* $A = \langle 7, 1, 4, 4 \rangle$ and $B = \langle 2, 8, 2, 2 \rangle$. The algorithm should return "such a pair exists".

   Design algorithms to solve this problem.

   (a) Your algorithm should run in $O(n \log n)$ time in the worst case. **(8p)**

   (b) Your algorithm should run in $O(n)$ time in the average case. **(7p)**

9. Let $G = (V, E)$ be a connected, weighted, undirected graph (with n vertices and m edges) and T a minimum spanning tree of G. Now, an edge $e = (u, w) \in E$ is deleted from G; resulted in a new connected graph H. Design an $O(m)$-time algorithm to compute a minimum spanning tree in H. **(10p)**

1. (a) The max-heap is $\langle 11, 8, 7, 2, 4 \rangle$.

   (b) The resulting array is $\langle 8, 4, 7, 2, 11 \rangle$.

   (c) Let $T(n)$ be the number of comparisons used by this algorithm in the worst case. $T(n) = O(1)$ if $n \leq 2$. For $n > 2$, we have $T(n) = 2T(\frac{n-1}{2}) + \Theta(\log n)$.

2. (a)

0	1	2	3	4	5	6	7	8	9
10			20	4				30	

   (b) i. The function is not a valid hash-function, since it maps to values outside the table range.

   ii. The function does not use the whole address space (e.g. only even table entries are used).

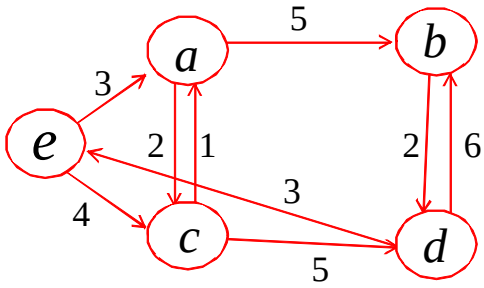
3. (a) False.

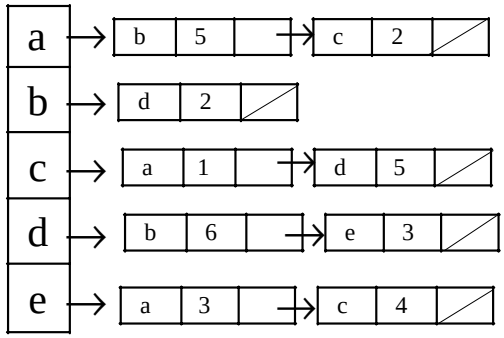
   (b) False.

   (c) False.

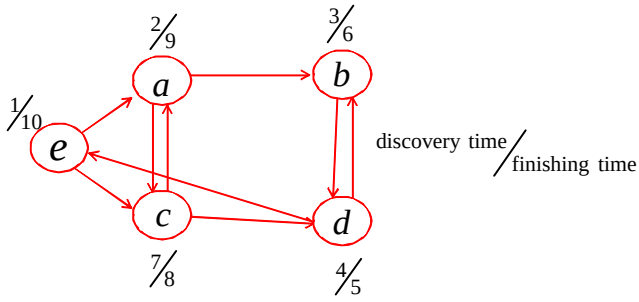
   (d) False.

4. (a)

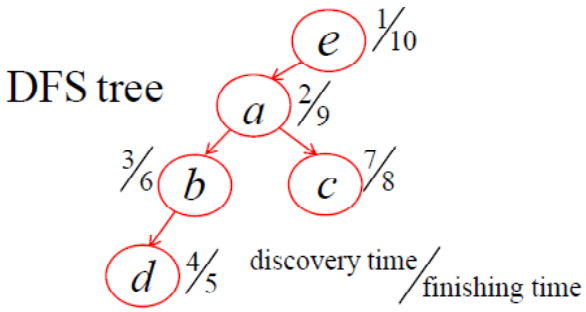




(b)



Answer:



(c)

	e	a	b	c	d
initialization	0	∞	∞	∞	∞
stage 1	0	3	∞	4	∞
stage 2	0	3	8	4	∞
stage 3	0	3	8	4	9
stage 4	0	3	8	4	9

5. (a) Repeated pair-wise merging (as the intermediate steps of merge sort).
   (b) Build a heap of size k on all the smallest elements from the given lists, and then repeated delete-min and insertion (inserting the next element from the list corresponding to the current minimum in the heap).

6. Notice that for any input array, there is always at least one such an element.

   $f\ (A, \ell, r)$:

   if $\ell = r$, then return a_ℓ
   $i \leftarrow \left\lceil \frac{\ell+r}{2} \right\rceil$
   if $a_i < a_{i+1}$, then return $f\ (A, i+1, r)$
   if $a_i > a_{i+1}$ and $a_{i-1} > a_i$, then return $f\ (A, \ell, i-1)$
   return a_i

   Correctness:

   During each recursive call of $f(A, left, right)$, we have maintained the invariant that $a_{left-1} < a_{left}$ and $a_{right} > a_{right+1}$ (which ensures the existence of such elements).

7. Sort first all the integers using radix sort, and then build a perfect balanced BST.

   Regarding the sorting, we use base-n representation, then each integer has $d = \log_n n^2 = 2$ digits. Thus, radix sort takes $2 \times \Theta(n+n) = \Theta(n)$ time.

8. Compute the sums of all the elements in A and B respectively; sum_A and sum_B

   Let $x = \frac{1}{2}(sum_A - sum_B)$ and $B' = \{x\} + B = \langle x + b_1, x + b_2, \cdots, x + b_n \rangle$

   What is needed to be done is to determine whether all the elements in

   $\langle a_1, a_2, \cdots, a_n,\ x + b_1, x + b_2, \cdots, x + b_n \rangle$ are distinct.

   (a) First sort $\langle a_1, a_2, \cdots, a_n,\ x + b_1, x + b_2, \cdots, x + b_n \rangle$ and then scan and check the sorted output.
   (b) Construct a hash table on all the elements in A.
   For every element of B', search for it in the hash table; return a hit.

9. If $e = (u, w) \notin T$, then the tree T is a MST of G'. Checking whether the edge $e = (u, w)$ is in T or not takes $O(n)$ time.

   If $e = (u, w) \in T$, then deleting the edge e from T leaves two connected components. Add the minimum weight edge with one vertex in each component (this can be done in $O(m)$ time). By the cut property, we have a MST of G'.