Data Structures

Final Exam. Fall 2003, Yen

作答在答案卷上； 題目不用繳回。 答案卷上務必寫上姓名、E-mail、 聯絡電話

1. **(10 pts)** After a `*delete-min*', then an `*insert 21*' operations are performed (in the given sequence) on the following array-implemented binary min-heap, what are the contents of the underlying array?

index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
key	6	10	15	12	22	20	55	14	32	25
Result	**10**	**12**	**15**	**14**	**21**	**20**	**55**	**25**	**32**	**22**

2. **(6 pts)** Answer the following questions:

(a) Which of the following sort algorithms have a worst case *O(n log n)* run time? *Quick-sort, Heap-sort, Insertion-sort, Selection-sort, Shell-sort, Merge-sort, Straight-radix-sort, Bucket-sort, Radix-exchange-sort, Bubble-sort.*

(b) Define what a *stable* sorting algorithm mean? Give an example of a stable sorting algorithm. See Textbook; bubble sort

3. **(15 pts)** Consider the following 12 data structures (heaps are assumed to be min-heaps.). Answer the following questions:

1. Sorted array	2. Simple binary search tree	3. Top-down splay tree	4. Leftist Heap
5. Unsorted array	6. AVL tree	7. Binary Heap	8. Skew Heap
9. Skip list	10. AA tree	11. Binomial heap	12. Fibonacci heap

(a) List those for which an *insertion* can be done in *O(log n)* time in the worst-case. (*** Notice that *O(log n)* includes *O(1)*.***) 4, 5, 6, 7, 10, 11, 12

(b) List those for which a *union* (i.e., merge) can be done in *O(log n)* time in the worst-case. 4, 11, 12

(c) List those for which *finding the minimum* can be done in *O(1)* time in the worst-case . 1, 4, 7, 8, 9, 12

(d) List those for which *finding an arbitrary key* may require *Ω(n)* time in the worst case. 2, 3, 4, 5, 7, 8, 9, 11, 12

(e) List those for which performing a sequence of *n insertions* can be done in *O(n log n)* time in the worst case. 3, 4, 5, 6, 7, 8, 10, 11, 12

4. **(10 pts) True / False:** For the true statements, simply write "true": no explanation is needed. For the false statements, explain *why* it is false. Your explanation should be *precise and concise.*

(1) The height of a leftist heap is O(*log N*). **False; O(N)**

(2) Sorting algorithms that only swap adjacent elements have a worst-case

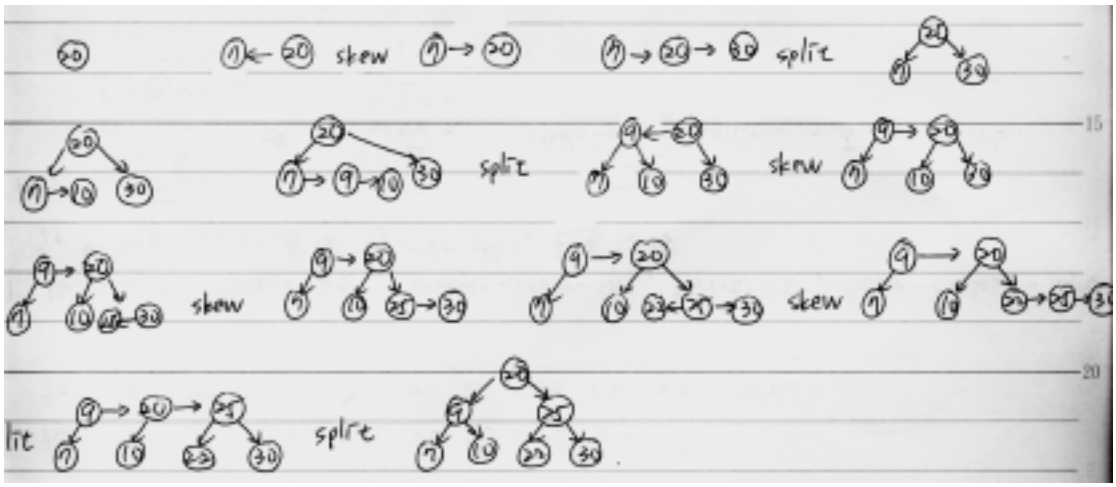
running time of *O(N log N)* operations. **False; $O(N^2)$**

(3) An advantage of splay trees over AVL trees is that it is not necessary to maintain information about the height or balance of nodes. **True**

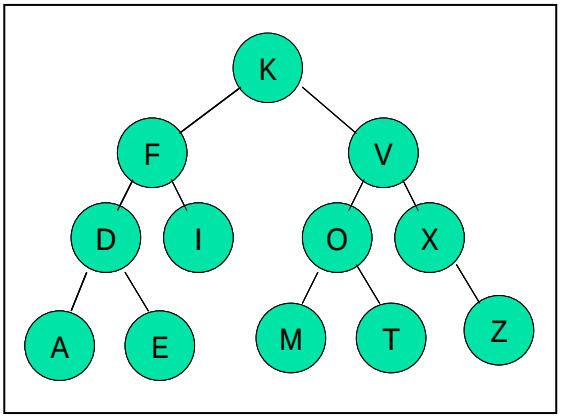
(4) Because inserting a key into a binary heap requires *Ω(log N)* time worst case, building a heap of size N from scratch requires *Ω(N log N)* time worst case. **False; O(N) using bottom-up approach**

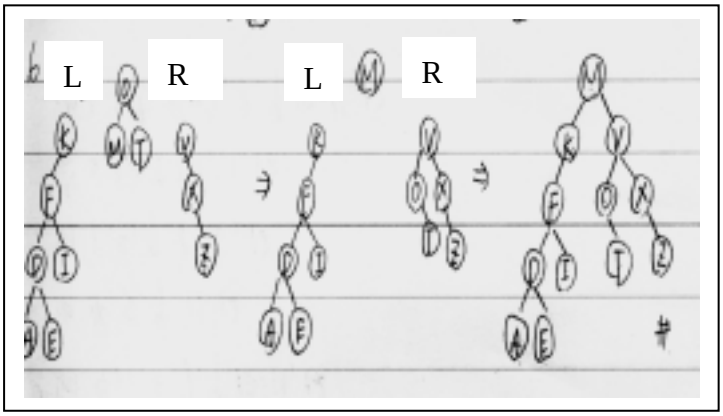
(5) Path compression is a technique for improving the amortized time of a sequence of insert and deleteMin operations on a heap.**False; for search in union-find**

5. **(10 pts)** Insert the list of numbers *20, 7, 30, 10, 9, 25, 22* into an initially empty *AA tree*. Show the tree after each insertion and each skew or split, specifying which one you performed.



6. **(10 pts)** Apply *top-down splaying* to node M of the following tree. Show your derivation in sufficient detail.





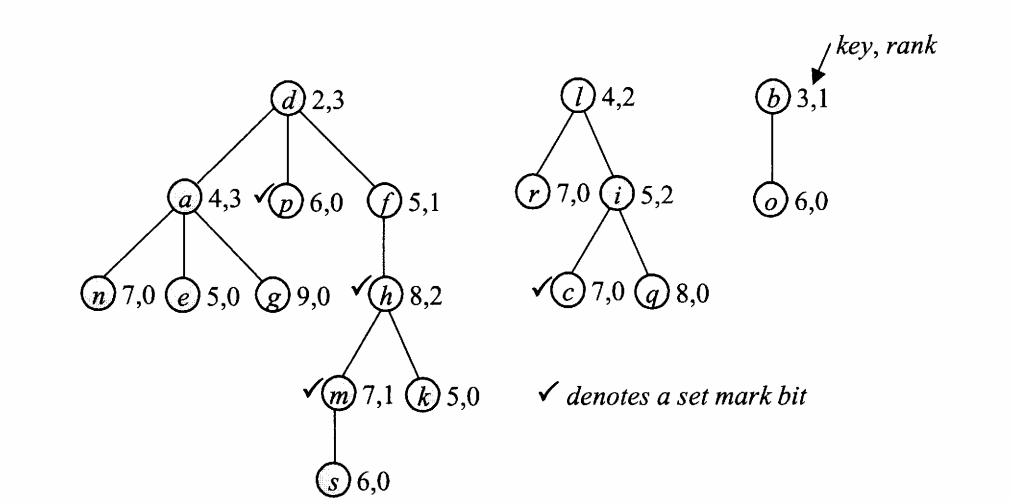
7. **(5 pts)** Consider the following decision tree of some sorting algorithm. Suppose the elements being input to some comparison based sorting algorithm are 5, 7, and 9. What would the input order of these three numbers be, for the *circled leaf* to be reached after the

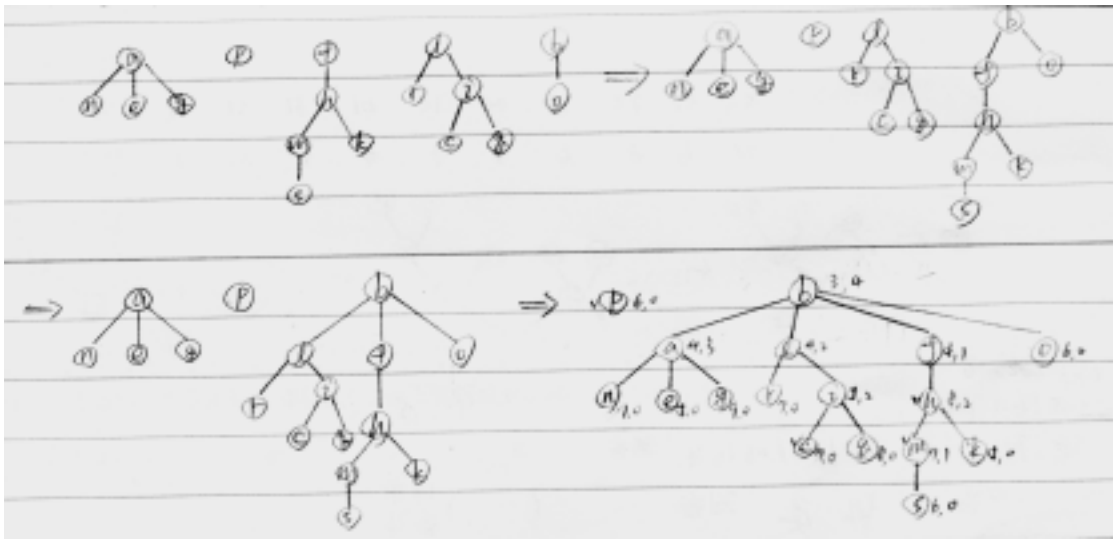
comparisons performed during the sort? I.e. what are a_1, a_2 and a_3? Answer: 5, 9, 7

8. **(12 pts)** Consider the Fibonacci heap given below. In using the potential function to perform amortized analysis, we define the potential of a Fibonacci heap to be $\Phi(H) = t(H) + 2m(H)$, where $t(H)$ = *# of trees* and $m(H)$ = *# of marked nodes.*
Answer the following questions:

(1) What is the potential of the heap shown below? 11

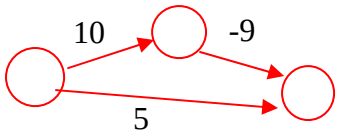
(2) Suppose a *delete-min* operation is performed. Show the resulting heap. (Be sure that you include the mark and rank information of each node.) What is the potential of the new heap? 10





9. **(10 pts)** Answer the following two questions:

(a) Give a graph with negative edges but without negative cycles such that Dijkstra's algorithm fails to find the shortest paths.



(b) Explain how you can use Bellman-Ford algorithm to detect whether a graph has a *negative cycle* (i.e., a cycle whose total weights is negative) or not. Why?
Answer: Run Bellman-Ford and if the labels remain *unstable* after |V| iterations, then the graph has a negative cycle.

10. **(12 pts)** Consider the following *leftist heap*.

(a) What are the *nlp (null path length)* values of nodes *4 (2), 19 (1), 6 (1), 27* (0)?

(b) Perform a *delete-min* operation and draw the resulting heap. Show your derivation in sufficient detail.