

# CS 3343 (Spring 2008) Exam 2

April 2, 2008

4:00pm - 5:30pm (90 minutes)

Name: \_\_\_\_\_ ID: \_\_\_\_\_

- Don't forget to put your name and ID on the cover page
- This exam is closed-book
- If you have a question, **stay seated** and raise your hand.
- Please try to write legibly – if I cannot read it, you may not get credit.
- Do not waste time – if you cannot solve a question immediately, skip it and return to it later.
- Try your best to answer each question. Partial credits will be given if you show that you have some ideas – but not according to the length of your answer.

1)	Quick sort		20
2)	Heaps		25
3)	Longest common subsequence		15
4)	Substitution method		10
5)	Order statistics		10
	Total		80

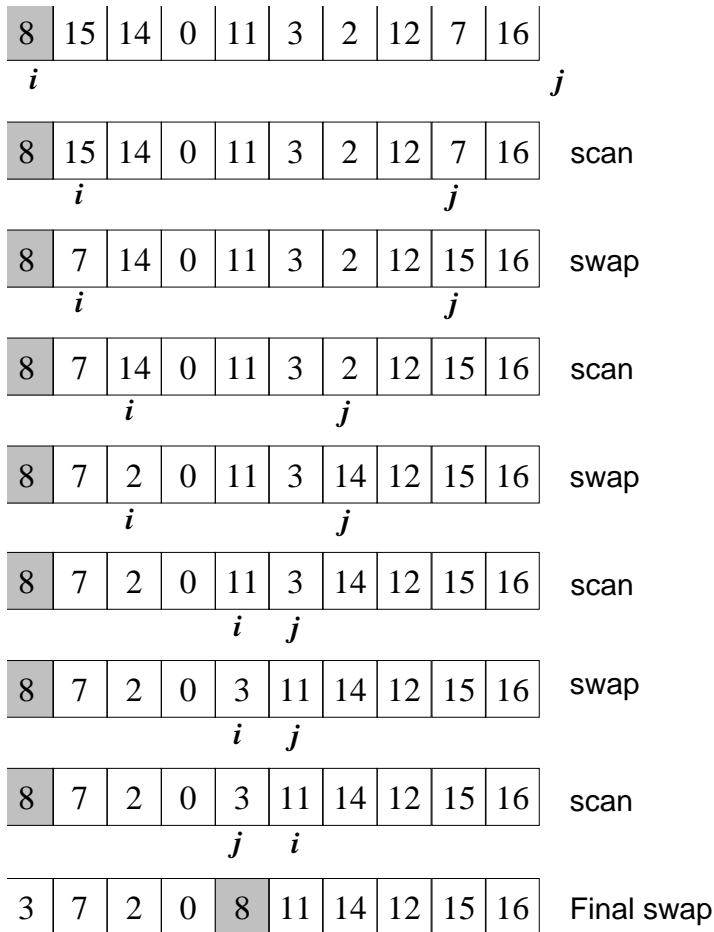
1. (20 points) Quick sort

- a. Complete the following illustration of Partition. **Show the positions of the two iterators  $i$  and  $j$  after each step.** In case you need an reference, below is the pseudocode for Partition.

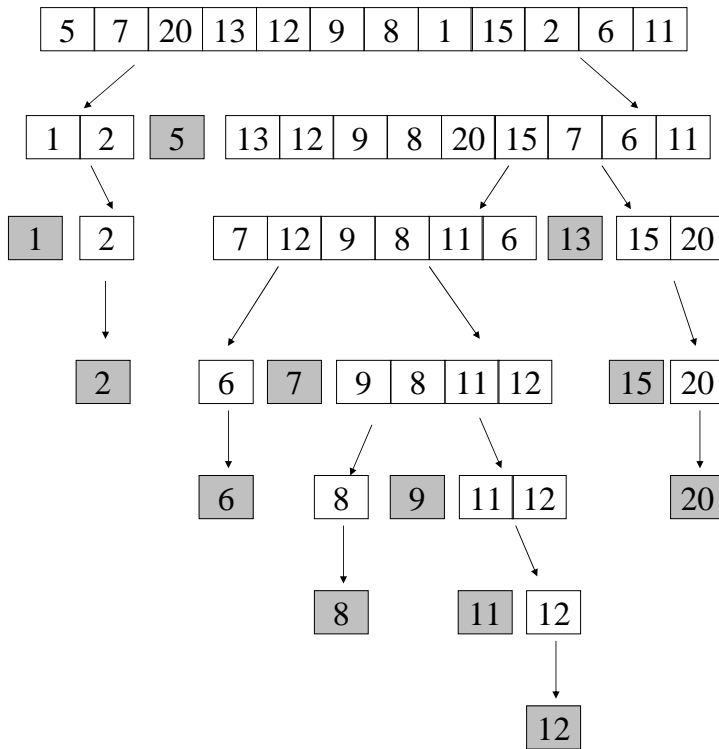
```

Partition(A, p, r)
  x = A[p]; // pivot is the first element
  i = p; j = r + 1;
  while (TRUE) {
    repeat
      i++;
    until A[i] >= x;
    repeat
      j--;
    until A[j] <= x;
    if (i < j) Swap (A[i], A[j]);
    else break;
  }
  swap (A[p], A[j]);
  return j;

```



- b. Complete the following illustration of Quick sort (assuming that the function Partition on the previous page is used). The grey cells are the pivot elements.



Sorted:

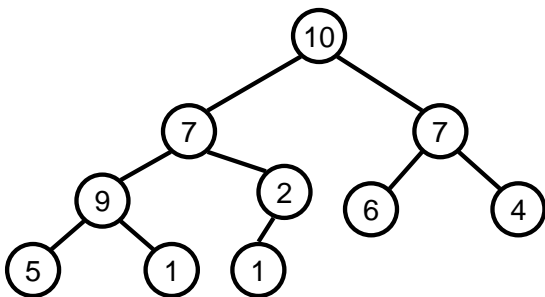


## 2. (25 points) Heaps

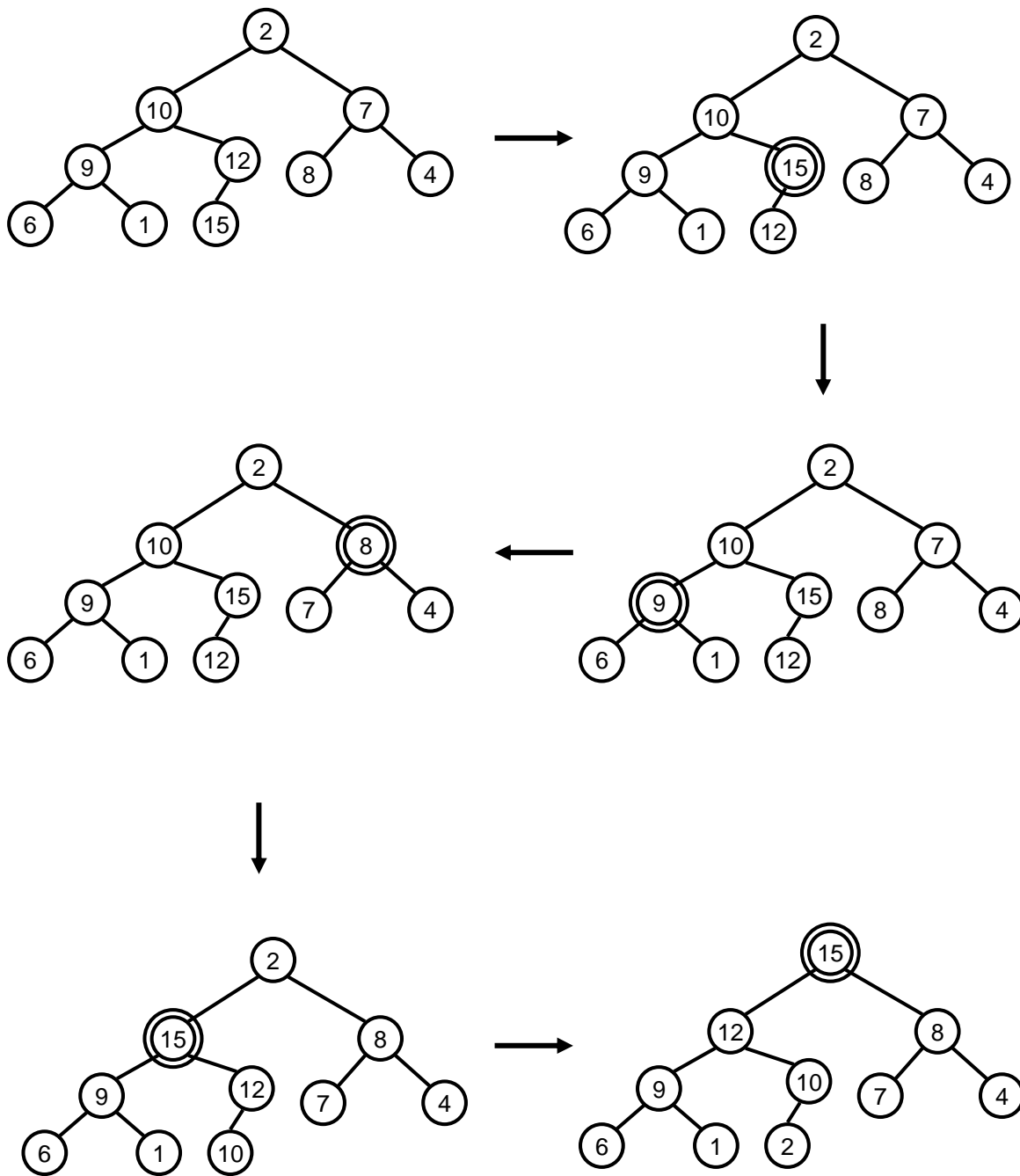
Note: by default, we mean a max-heap, i.e., the max (rather than the min) element is at the root.

- a. Is the tree below a heap? Why or why not?

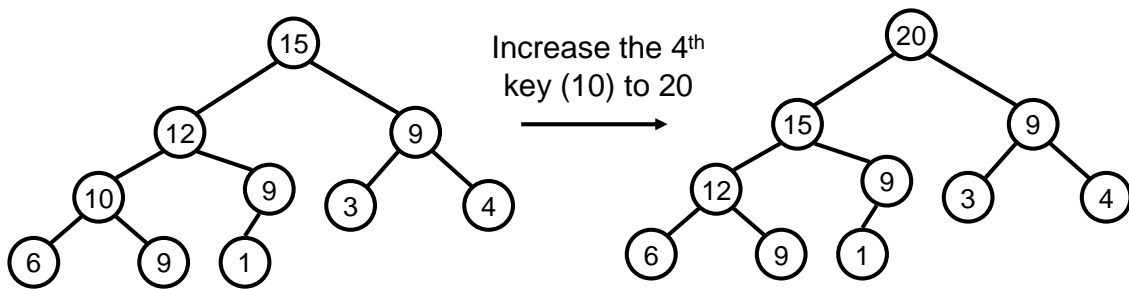
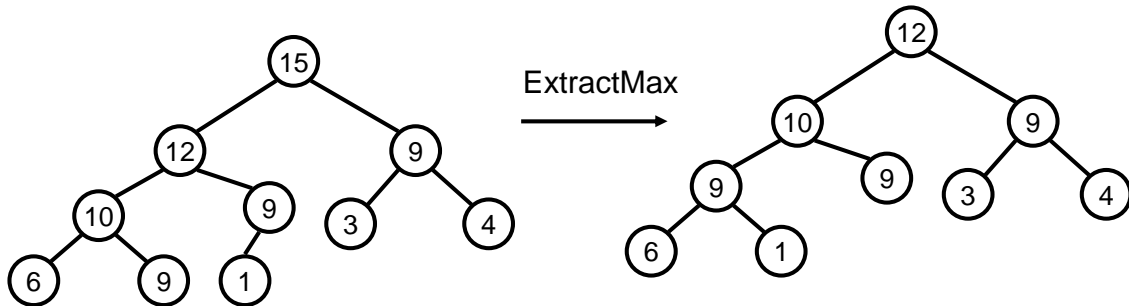
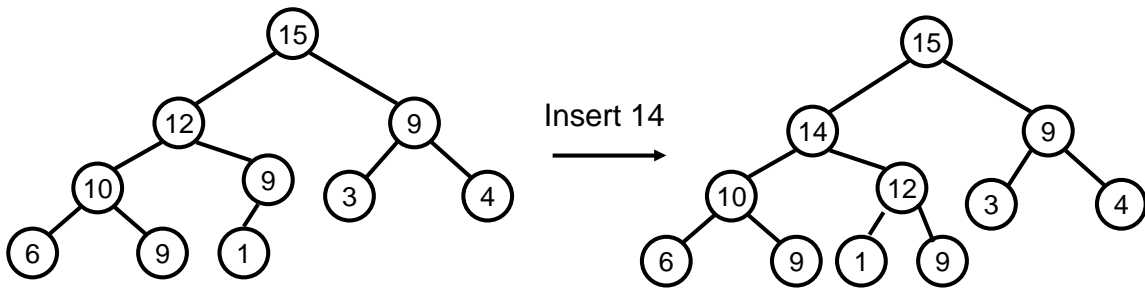
Answer: This is not a heap, since the element 7 is smaller than one of its children, 9, thus violating the heap property.



b. Illustrate how buildheap works using the following example. Each step is the result of one call to the heapify function.



c. Starting from the heaps on the left, show the contents of the new heaps after each heap operation.



3. (15 points) Longest common subsequence (LCS).

- a. Complete the following dynamic programming table to compute the length of LCS between two strings ACDBADAC and ACBADDCAB.

		$j$	0	1	2	3	4	5	6	7	8
		$y[j]$	A	C	B	A	D	C	A	B	
$i$	$x[i]$	0	0	0	0	0	0	0	0	0	0
1	A	0	1	1	1	1	1	1	1	1	1
2	C	0	1	2	2	2	2	2	2	2	2
3	D	0	1	2	2	2	3	3	3	3	3
4	B	0	1	2	3	3	3	3	3	3	4
5	A	0	1	2	3	4	4	4	5	5	5
6	D	0	1	2	3	4	5	5	5	5	5
7	A	0	1	2	3	4	5	5	6	6	6
8	C	0	1	2	3	4	5	6	6	6	6

- b. What is the actual longest common subsequence? If there are multiple longest common subsequences, report all of them.

Answer: as shown above, there are two LCSs, ACBADA and ACBADDC.

4. (10 points)  $T(0) = \Theta(1)$ , and  $T(n) = T(n/3) + T(n/4) + n$ . Using the **substitution method**, prove that  $T(n) \in O(n)$ .

**Proof:**

According to the definition of Big-oh, this means we must prove that there exist a positive  $c$  and a positive  $n_0$  such that  $T(n) \leq cn$  for all  $n > n_0$ .

Let's assume that the above inequality is valid for for all  $k < n$ , i.e.,  $T(k) \leq ck$ . This means  $T(n/3) \leq cn/3$  and  $T(n/4) \leq cn/4$ . Substitute  $T(n/3)$  and  $T(n/4)$  into the recurrence, we obtain

$$T(n) = T(n/3) + T(n/4) + n \leq cn/3 + cn/4 + n = 7cn/12 + n = cn + (n - 5cn/12).$$

If we choose  $c > 12/5$ , we will have  $(n - 5cn/12) < 0$ , which implies  $T(n) = cn + (n - 5cn/12) \leq cn$ . Therefore,  $T(n) \in O(n)$ .

### 5. (10 points) Order statistics.

The pseudocode below is the selection algorithm we studied in class – it returns the  $i$ -th smallest element in an array.

```
1  RAND-SELECT(A, p, q, i)          // find the i-th smallest of A[p . . q]
2      if (p == q & i != 1) then quit and report error!
3      r = RAND-PARTITION(A, p, q) // r is the index of the pivot
4      k = r - p + 1                // k = rank(A[r])
5      if i == k then
6          return A[r]
7      if i < k then
8          return RAND-SELECT(A, p, r - 1, i)
9      else
10         return RAND-SELECT(A, r + 1, q, i - k)
```

- a. Modify the algorithm so that, instead of returning the  $i$ -th smallest element, it returns the  $i$  smallest elements, i.e., the first, second, ..., and  $i$ -th smallest elements. We are not interested in their relative orders. You can either describe your modifications using lay language or modify the above pseudocode.

**Answer:** We can simply change Line 5 to

```
return A[p..r]
```

and change Line 10 to

```
return concatenate(RAND-SELECT(A, r + 1, q, i - k), A[p..r]).
```

- b. How does the asymptotic time complexity of your new algorithm compare to the one above? Justify your answer briefly.

**Answer:** The expected running time would still remain in  $\Theta(n)$ . Concatenating the subarray with another subarray and returning the subarray to the calling function take time linear to the subarray size. The function `Rand_Partition` also takes time linear to the the subarray size. Therefore, the time complexity will only be increased by a constant factor.