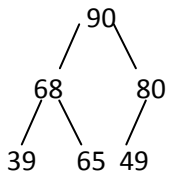
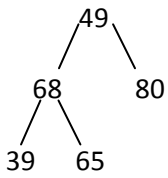
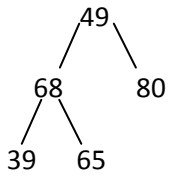
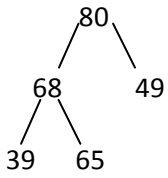
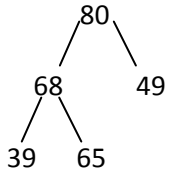
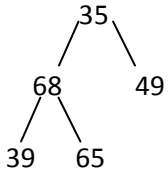
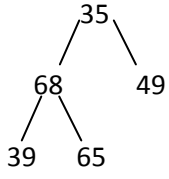
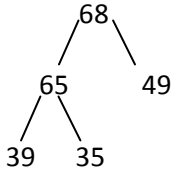
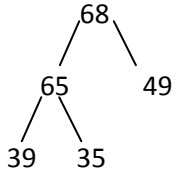
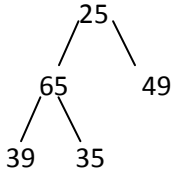
CS351 HW2 Solutions - October 30, 2010

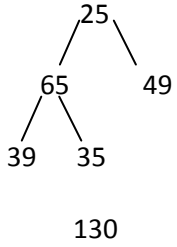
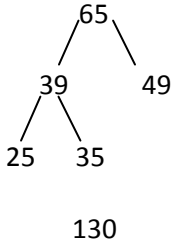
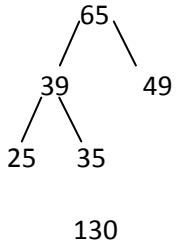
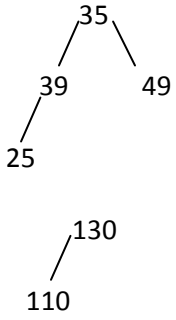
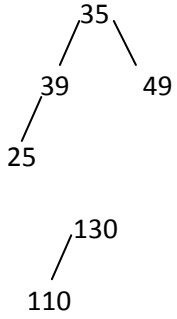
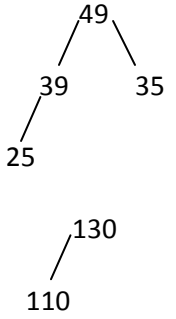
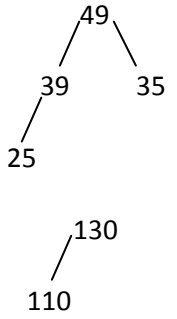
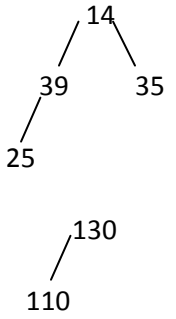
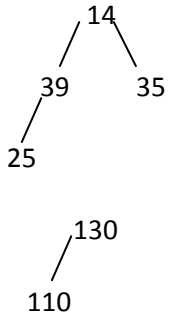
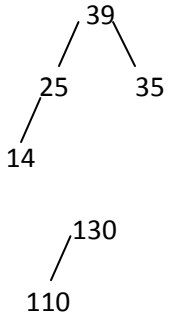
Solutions are mostly due to
Emre Nevayeshirazi

1 -

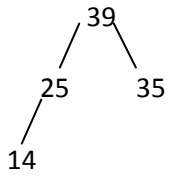
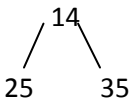
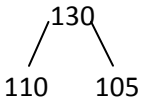
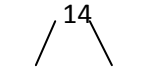
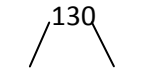
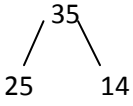
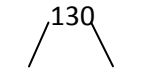
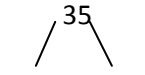
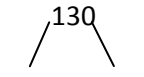
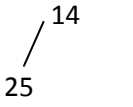
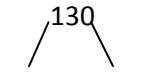
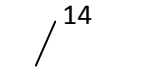
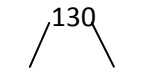
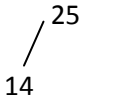
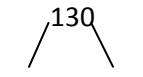
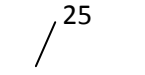
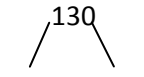
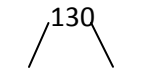
 130	90 goes to output file. 130 comes as new input, since it is greater than output 90 it goes to new heap. 49 is the new root, since 90 has gone to output file.	 130
 130	We need to reorganize old heap to keep priority queue structure. Swap 80 and 49.	 130
 130	80 goes to output file. 35 comes as new input, since it is less than output 80, it goes to old heap. 35 is the new root for old heap, since 80 has gone to output file.	 130
 130	We need to reorganize old heap to keep priority queue structure. Swap 68 and 35 first. Swap 65 and 35 then.	 130
 130	68 goes to output file. 25 comes as new input, since it is less than output 68, it goes to old heap. 25 is the new root for old heap, since 68 has gone to output file.	 130

CS351 HW2 Solutions - October 30, 2010

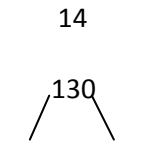
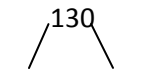
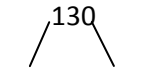
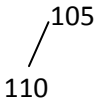
Solutions are mostly due to
Emre Nevayeshirazi

	We need to reorganize old heap to keep priority queue structure. Swap 65 and 25 first. Swap 39 and 25 then.	
	65 goes to output file. 110 comes as new input, since it is greater than output 65, it goes to new heap. 35 is the new root for old heap, since 65 has gone to output file.	
	We need to reorganize old heap to keep priority queue structure. Swap 35 and 49.	
	49 goes to output file. 14 comes as new input, since it is less than output 49, it goes to old heap. 14 is the new root for old heap, since 49 has gone to output file.	
	We need to reorganize old heap to keep priority queue structure. Swap 39 and 14 first. Swap 14 and 25 then.	

CS351 HW2 Solutions - October 30, 2010
Solutions are mostly due to
Emre Nevayeshirazi

 	39 goes to output file. 105 comes as new input, since it is greater than output 39, it goes to new heap. 14 is the new root for old heap, since 39 has gone to output file.	 
 	We need to reorganize old heap to keep priority queue structure. Swap 14 and 35.	 
 	35 goes to output file. No input. Therefore,14 is the new root.	 
 	We need to reorganize old heap to keep priority queue structure. Swap 25 and 14.	 
 	25 goes to output file. No input. Therefore, 14 is the new root.	14 

CS351 HW2 Solutions - October 30, 2010
Solutions are mostly due to
Emre Nevayeshirazi

	14 goes to output file. No input. Therefore, old heap is destroyed.	
	130 goes to output file as a new segment.. No input. Therefore,105 is the new root.	
	We need to reorganize old heap to keep priority queue structure. Swap 105 and 110.	
	110 goes to output file. No input. 105 is the new root.	105
105	105 goes to output file. No input. Heap is destroyed.	

Output Segment 1 : 90, 80, 68, 65, 49, 39, 35, 25, 14

Output Segment 2 : 130, 110, 105

2 –

No of Sorted Segments : 800 / 10 = 80

b = ((800 * 10^6) / 2400)

= 333,333.3

Sort Time : 2 * b * ebt

= 2 * 333,333.3 * 0.84 msec

= 560,000 msec = 9.3 min.

CS351 HW2 Solutions - October 30, 2010
Solutions are mostly due to
Emre Nevayeshirazi

3 –

A –

No of Passses : $\lceil \log_2 80 \rceil = 7$

Pass	1	2	3	4	5	6	7
Segment Size	10 MB	20 MB	40 MB	80 MB	160 MB	2 x 320 MB 1 x 160 MB	1 x 640 MB 1 x 160 MB
No of Segments	80	40	20	10	5	3	2

B –

Time Needed to Merge : (No of Passes) * 2 * b * ebt

b = ((800 * 10^6) / 2400)

= 333,333.3

= 7 * 2 * 333,333.3 * 0.84 msec => 65 min.

C –

Number of Seeek and Rotations : (No of Passes) * p * 2 * (nsg)

Total Time for Seek and Rotations : (No of Passes) * p * 2 * (nsg) * (r + s)

= 7 * 2 * 2 * 80 * (16 + 8.3) = 54,432 msec

⇨ 54.43 sec.

CS351 HW2 Solutions - October 30, 2010
Solutions are mostly due to
Emre Nevayeshirazi

4 –

A –

No of Passses : $\lceil \log_4 80 \rceil = 4$

Pass	1	2	3	4
Segment Size	10 MB	40 MB	160 MB	1 x 640 MB 1 x 160 MB
No of Segments	80	20	5	2

B –

Time Needed to Merge :

b = ((800 * 10^6) / 2400) = 333,333.3

(No of Passes) * 2 * b * ebt = 4 * 2 * 333,333.3 * 0.84 = 2,239,999.776 msec ➔ 37.33 min.

C –

Number of Seeek and Rotations : (No of Passes) * p * 2 * (nsg) * (r + s)

= 4 * 4 * 2 * 80 * (16 + 8.3) = 62,208 msec

=62.208 seconds ➔ 1 min.

5 –

If p = nsg then we are using 80 way merge.

In this case,

Total Time Needed for Merge : $\lceil \log_{80} 80 \rceil$ * [p * 2 * nsg * (s + r) + (2 * b * ebt)]

= 80 * 2 * 80 * (24.3) + 2 * (333,333.3) * (0.84)

= 311,040 + 559,999.944 msecs

=871,039.944 msecs

= 14.51 minutes

P= 80 means that we read one record from each sorted segment (i.e., read 80 records) and write the record with the smallest key to the output file (assuming that we are sorting in ascending order) and read the next record from the sorted segment corresponding to the record written to the output file. Note that we have buffering provided by the operating system, so we write a block when the buffer is full (after many logical writes –writing to the buffer- there will be a physical write –writing a block to the output file-). For selecting the record to be written to the output file the use of a min-heap looks reasonable. The memory requirement is small: we need room to keep 80 records in main memory, so it is practically doable. (See p. 111-112 in Salzberg's book. The discussion in the book points out that p= nsg is the best for all numbers less than 196.)

6 –

If our file is allready in desired order, then there will be only one output segment. We are using replacement selection sort and we can overlap write and read operations. Since our file is in desired order, there will be no input record that is less than output record. Therefore, new heap tree will never be created. **Therefore, we will have only one output segment. Its expected memory size is 800 MB.**

7 –

First, we need to fill 10 MB of memory that is available to us.

Number of records that 10 MB can hold = ((10 * 10^6) / 200) = 50,000 records

Total Number of Records = ((800 * 10^6) / 200) = 4,000,000 records

For the first 50,000 records we will not have any output. After first 50,000 records we will start outputing. During the output of this 50,000 records, new inputs will go to new heap that is started with 50,001st record. For the first 50,000 record we will have 1 output segment. For the second 50,000 records we will have another output segment and so on. **Therefore, 4,000,000/50000 = 80 number of sorted segments will occur. The memory size of each is exactly 10 MB.**