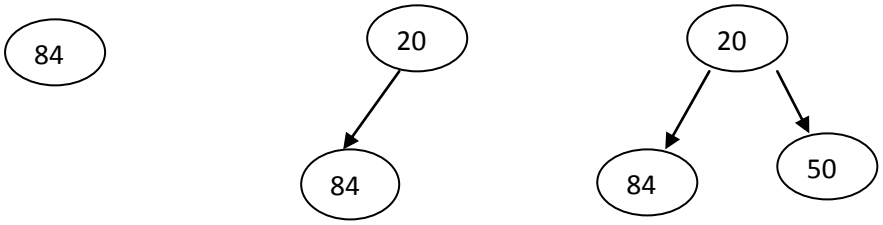
CS351: DATA ORGANIZATION AND MANAGEMENT - FALL 2011
HOMEWORK 2

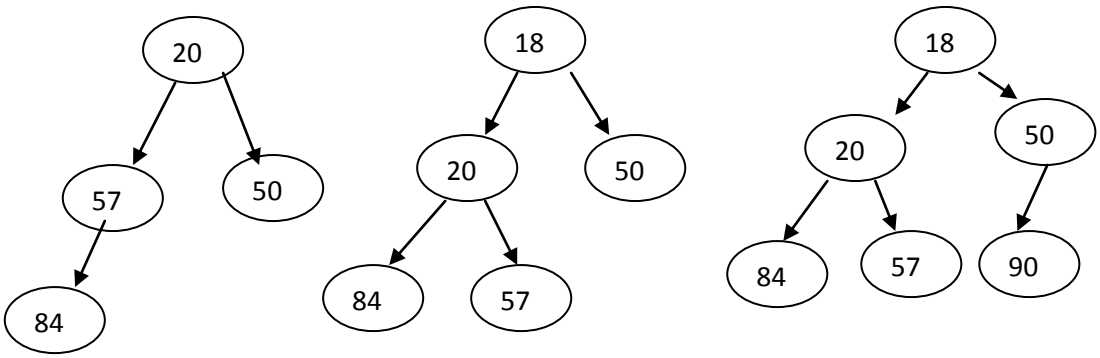
1) [20,50,57,18,90] [50,57,18,90] [57,18,90]

Insert 84 Insert 20 Insert 50



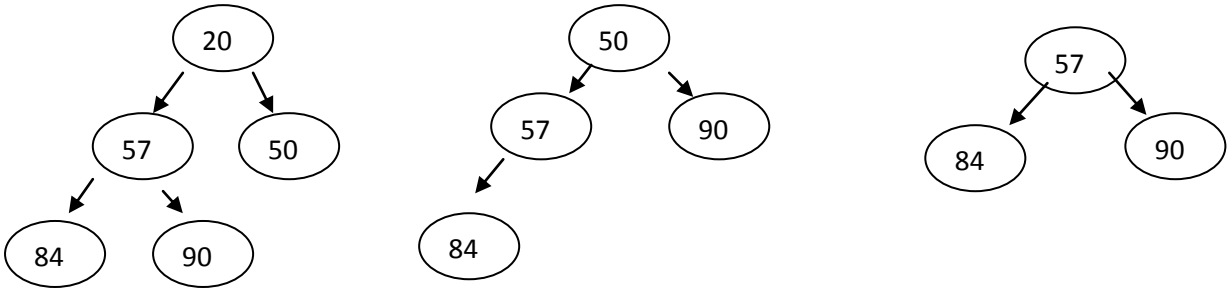
[18,90] [90] (No More Records)

Insert 57 Insert 18 Insert 90

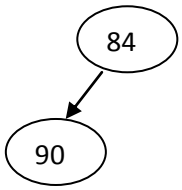


Now, the heap is full and all records are in the heap. So, if records of the heap are removed from top, an sorted segment is achieved.

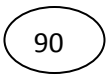
Output 18 → [18] Output 20 → [18,20] Output 50 → [18,20,50]



Output 57 → [18.20.50,57]



Output 84 → [18,20,50,57,84]



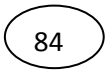
Output 90→ [18,20,50,57,84,90]

Now, we have a sorted segment of six records.

2) [84,30,50,57,28,90,92,15,24,88,12,12,98] → Input

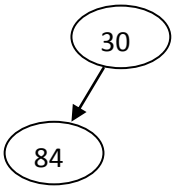
Read 84

[30,50,57,28,90,92,15,24,88,12,98]



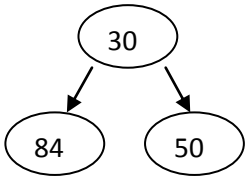
Read 30

[50,57,28,90,92,15,24,88,12,98]



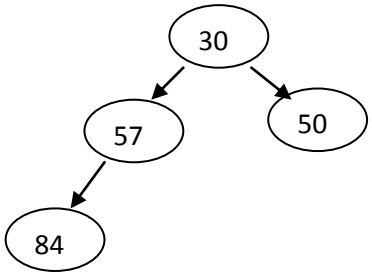
Read 50

[57,28,90,92,15,24,88,12,98]

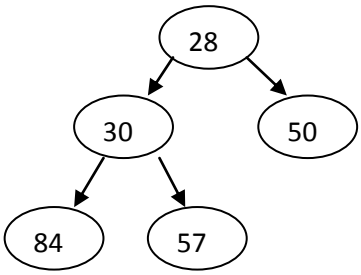


Read 57

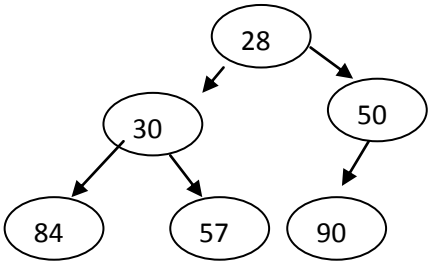
[28,90,92,15,24,88,12,98]



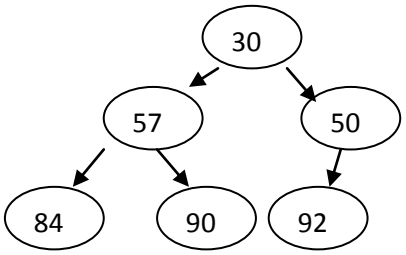
Read 28 → [90,92,15,24,88,12,98]



Read 90 → [92,15,24,88,12,98]

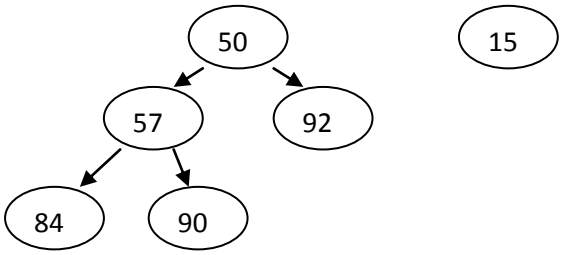


Write 28, Read 92 → [15,24,88,12,98]



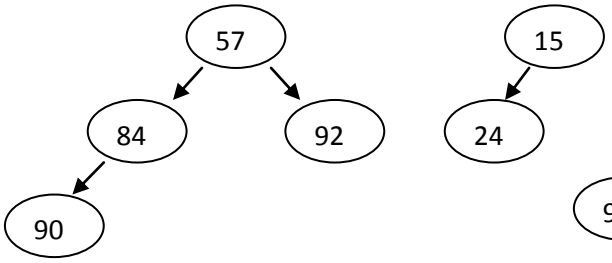
Output → [28]

Write 30, Read 15 → [24,88,12,98]



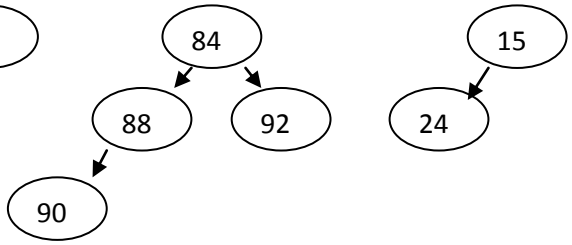
Output → [28,30]

Write 50, Read 24 → [88,12,98]



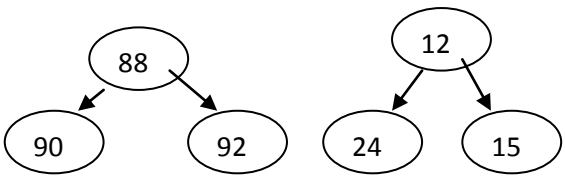
Output → [28,30,50]

Write 57, Read 88 → [12,98]



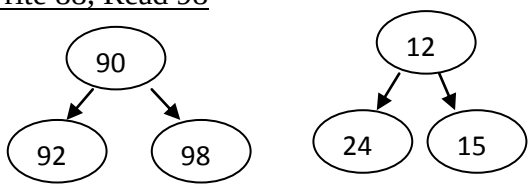
Output → [28,30,50,57]

Write 84, Read 12 → [98]



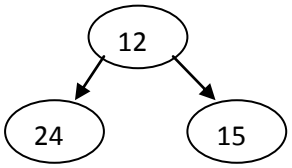
Output → [28,30,50,57,84]

Write 88, Read 98



Output → [28,30,50,57,84,88]

Write First Heap Records Sequentially



Output → [28,30,50,57,84,88,90,92,98]

Write Second Heap Records Sequentially

Output →first sorted segment [28,30,50,57,84,88,90,92,98]

second sorted segment [12,15,24]

QUESTION 3)

If the size of file is less or equal to the capacity available in the memory it is possible to use heap sort without merging and obtain a sorted file. If we have one sorted segment, there is no need to merge and file would become sorted.

For the replacement selection sort method, the case is similar. If the file size is less or equal to memory capacity it is possible to have a sorted file without merging. Additionally, while processing this method if we do not have to create an additional heap structure, it is also possible to have a sorted file without merging. This case is possible if only the current input is less than the current output (if the file is already sorted this will happen). In this case, the input will be inserted in the present heap and the same heap will be processed until the end of input file.

QUESTION 4)

a) 2000 / 10 = 200 is the number of sorted segments we are going to have in heap sort. For each of the segments we need a read and a write operation which causes 200*2 = **400** (s+r) in total.

b) In heap sort we read all of the blocks (b: no. of blocks) and we write all of the blocks.

$2*b*ebt = 2*(2000*10^6 / 2400)*(0.84) = 1400$ s = 23.3 minutes

c) In replacement selection sort, we overlap reading with writing so the overall process is equal to have a exhaustive reading which is half of the answer in part b.

$b*ebt = (2000*10^6 / 2400)*(0.84) = 700$ s = 11.6 minutes

QUESTION 5)

For this question following values are considered: (In the calculation of s+r we use estimated values as suggested by Salzberg)

R = 200 bytes, available memory = 20 MB

a) The whole file is $(4*10^6*200)$ = 800 MB. Each sorted segment should be 20MB, so there will be 800/20 = 40 sorted segments which causes 80 (s+r) operations to create sorted segments in heap sort part. Total time is:

$80*(24.3) + 2*b*ebt$ ==> (with approximation) $2*(4*10^6*200 / 2400)*0.84 = 560$ s = **9.36 min.**

b) 2-way merge: The heap sort part will cost 9.36 minutes. There are (800 / 20) = 40 sorted segments in file. The number of passes are ceiling($\log_2 40$)= 6. Since it is 2-way merge each sorted segments will be read in two parts. So in total there will be 80 read operations. The amount of write operations will be approximately the same as read operations. So there would be 80 write operations. Also reading and writing will cause 2*b*ebt in each pass. In total it is equal to exhaustive reading and writing in each

pass. For next passes, the same procedure will be followed. The number of sorted segments in each pass are 40, 20, 10, 5, 3, 2 respectively. Approximate total time is:

9.36 + 6*[2*2*80* (24.3)+(2*b*ebt)] ==> ~ **65.5** min. (see Salzberg p. 111)

c) 4-way merge: The heap sort part will cost 9.36 minutes. There are (800 / 20) = 40 sorted segments in file. The number of passes are ceiling($\log_4 40$)= 3. Since it is 4-way merge each sorted segments will be read in four pieces. So in total there will be 160 read operations. The amount of write operations will be approximately the same as read operations. So there would be 160 write operations. Also reading and writing will cause 2*b*ebt in each pass. In total it is equal to exhaustive reading and writing in each pass. For next passes, the same procedure will be followed. The number of sorted segments in each pass are 40, 10, 3 respectively. Approximate total time is:

9.36 + 3*[2*4*80*(24.3) + (2*b*ebt)] ==> ~ **37.5** min. (see Salzberg p. 111)

d) p-way merge: The heap sort part will cost 9.36 minutes. There are P sorted segments in file and P (P= 40)-way merge is done which means one pass is enough to complete the merging process. So the number of the required read operations are P*P. The number of write operations will be approximately the same as read operations. Since there is only one pass, one 2*b*ebt will be added. So the total time is

$$\text{Total time} = 9.36 + P*2*P*(s+r) + (2*b*ebt) = 18.7 + 2*P^2*(s+r)$$

Reasoning for no. of (s+r): P: we divide each sorted segment into P pieces, 2: we read all and write all, P: we have P number of sorted segments ==> P*2*P

$$2*P^2*(s+r)= 2*40*40*(16+8.3)= 3200*24.3= 77{,}760\text{ms} ==> \sim 78\text{sec}$$

e) The total time difference between 2-way merge and 4-way merge is (65.5 – 37.5) = 28 minutes. It is a significant difference. The reason is the number of ways (the value of p= 2, p= 4, 2-way, 4-way) affects the number of passes and in each pass we perform an exhaustive read and write. In 2-way merge we make 3 more exhaustive readings and writings than 4-way merge which creates the significant difference between these two methods.[1]

[1] Solutions are due to Övünç Sezer and Sefa Şahin Koç (with some editing).