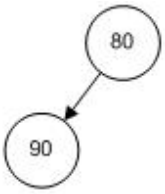
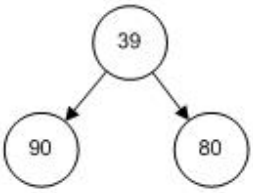
S. 1.

1. Insert 80

2. Insert 90

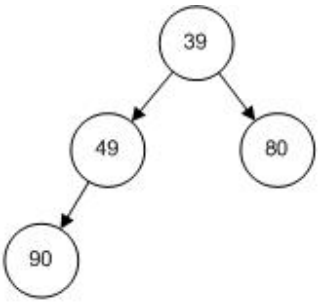


3. Insert 39

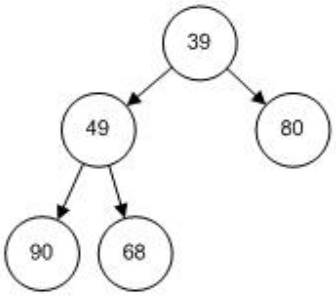


---

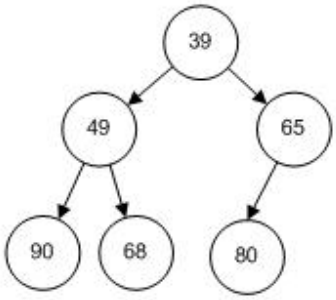
4. Insert 49



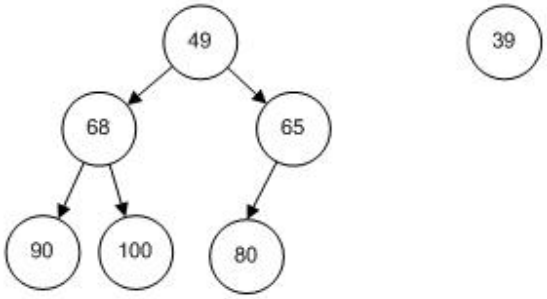
5. Insert 68



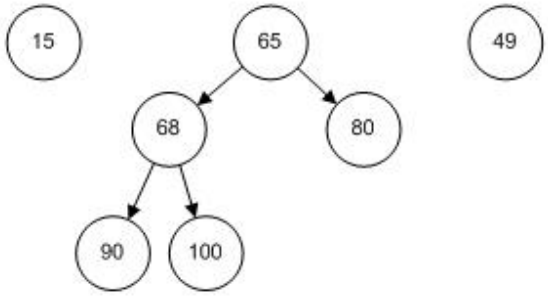
6. Insert 65



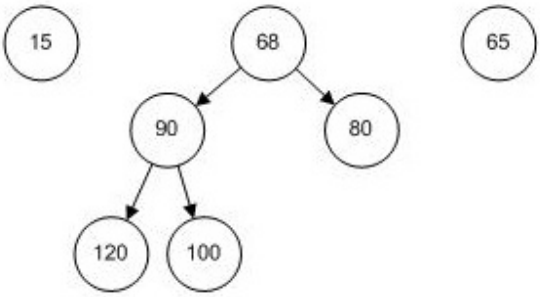
7. Input 100, Output 39



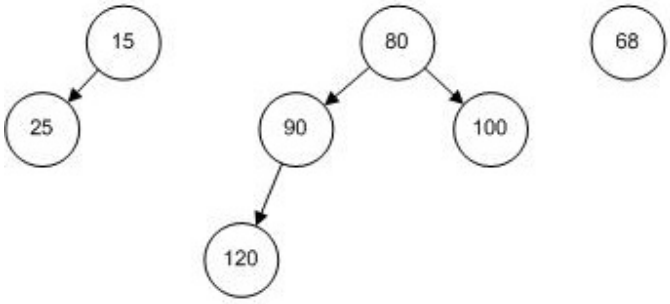
8. Input 15, Output 49



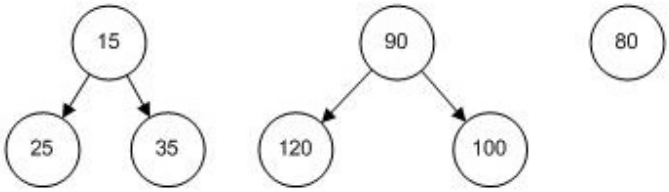
9. Input 120, Output 65



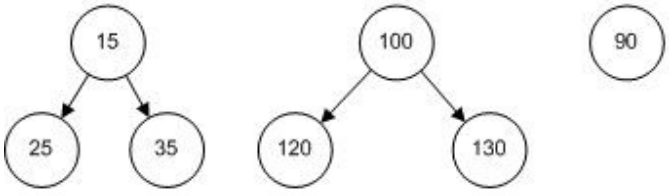
10. Input 25, Output 68



11. Input 35, Output 80



12. Input 130, Output 90



After all, we will merge the newly created queue with the old one by outputting the top records one after another. In this way, we will have these records in sorted order.

S. 2.

Without merging the file, the cost will only be reading and writing the file in 10 MB segments.

Let *b* be the number of blocks in the file.

$$2 \times \mathrm{b} \times \mathrm{ebt} = 2 \times \frac{10MB}{6} \times 0.84 = 2800\,\mathrm{sec}$$

S. 3.

Assuming 10 MB is about 10,000,000 bytes, the number of segments in the file;

$$\mathrm{nsg} = \frac{10MB \times 400}{10MB} = 400 segments$$

We already know that 2800 seconds passed in heapsort. So, we are only expected to calculate the merging time.

As we know, the number of passes needed is calculated by;

$$\lfloor \log_P (nsg) \rfloor$$

a. 2-way merge

Number of passes = $\lfloor \log_2 (400) \rfloor = 9$

Passing time = $9 \times \left[2 \times P \times nsg \times (s + r) + 2 \times b \times ebt \right]$

= $9 \times \left[2 \times 2 \times 400 \times (24.3) + 2 \times \frac{10MB}{6} \times 0.84 \right]$ = 25, 550 seconds

So, total time for2-way merging is; 25, 550 seconds + 2800 seconds = 28,350 seconds

b. 4-way merge

Number of passes = $\lfloor \log_4 (400) \rfloor = 5$

Passing time = $5 \times \left[2 \times P \times nsg \times (s + r) + 2 \times b \times ebt \right]$

= $5 \times \left[2 \times 4 \times 400 \times (24.3) + 2 \times \frac{10MB}{6} \times 0.84 \right]$ = 14, 389 seconds

So, total time for2-way merging is; 14, 389 seconds + 2800 seconds = 17, 189 seconds

c. P-way merge

We have 400 segments, so P = 400 in this part.

Number of passes = $\lfloor \log_{400}(400) \rfloor = 1$

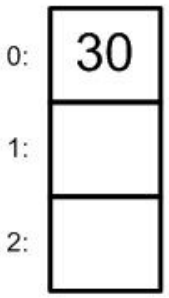
Passing time = $1 \times [2 \times P \times nsg \times (s + r) + 2 \times b \times ebt]$

$$= 1 \times \left[2 \times 400 \times 400 \times (24.3) + 2 \times \frac{10MB}{6} \times 0.84 \right] = 10,576 \text{ seconds}$$

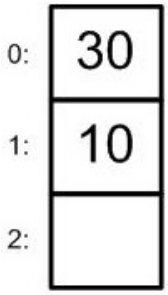
So, total time for2-way merging is; 14, 389 seconds + 2800 seconds = 13, 376 seconds

S. 4.

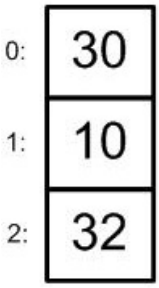
1. Insert 30



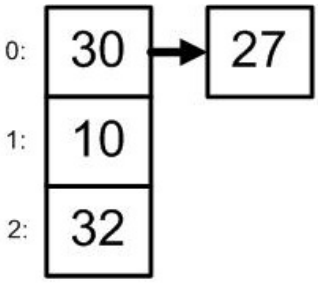
2. Insert 10



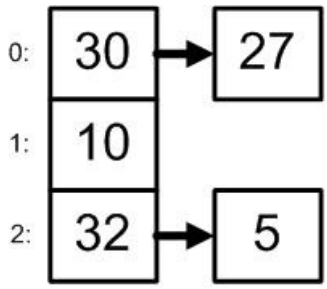
3. Insert 32



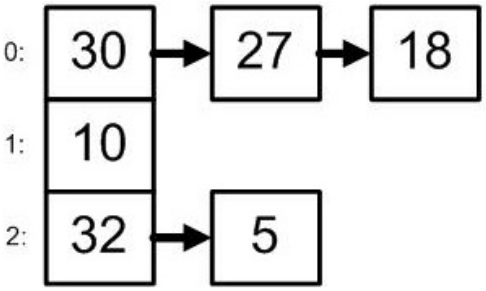
4. Insert 27



5. Insert 5



6. Insert 18



a. 3 disk accesses are needed. We will first access 30, then 28, and then 18.

b. We will divide the total number of accesses required for each record with the total number of records to find the average.

Let n be the number of records in the hash table.

The average number of disk access required for successful search is;

$$\frac{\sum_{i=0}^{n}(numberOfDiskAccessToRetrieve)_i}{n} = \frac{1+2+3+1+1+2}{6} = 1.\overline{6}$$

c. Using the same ideology, the number of accesses for an unsuccessful search;

$$\frac{3+1+2}{3} = 2$$

S. 5.

We are expected to calculate the place of the records using mod (key, 31) hash function, so the pseudo keys and binary values of the given records will be the same because they are less than 2^31 value.

First thing we are going to do is to calculate the binary values of the given records.

10	= 1010	7	= 111	17	= 10001	16	= 10000
20	= 10100	13	= 1101	21	= 10101	22	= 10110
3	= 11	14	= 1110	25	= 11001	30	= 11110

Then we will insert these records into hash table.

1. **Insert 10**

0:	10		
1:			

2. **Insert 20**

0:	10	20	
1:			

3. **Insert 3**

0:	10	20	
1:	3		

4. **Insert 7**

0:	10	20	
1:	3	7	

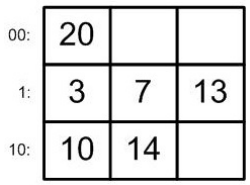
5. **Insert 13**

0:	10	20	14
1:	3	7	

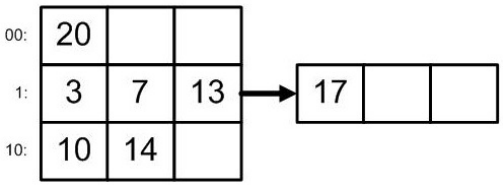
6. **Insert 14**

0:	10	20	14
1:	3	7	13

7. Update table

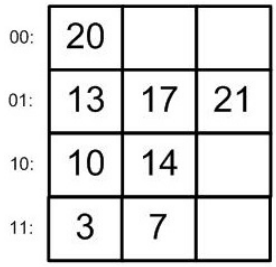


8. Insert 17

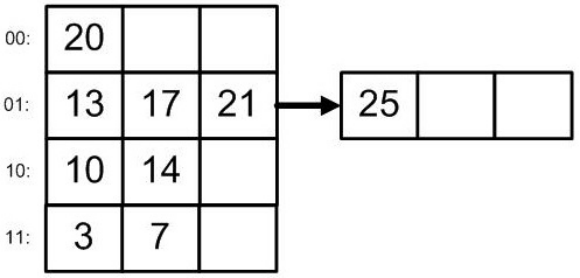


9. Insert 21

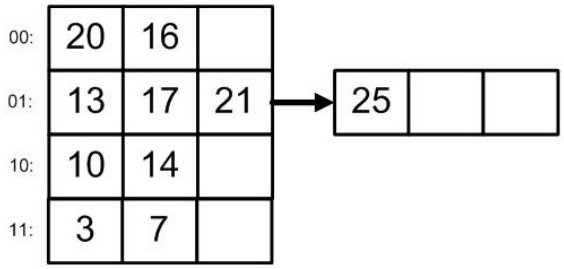
10. Update Table



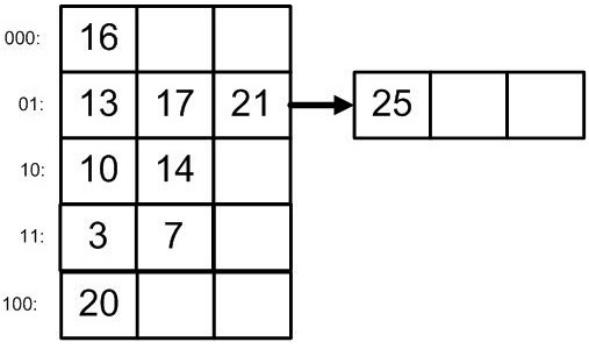
11. Insert 25



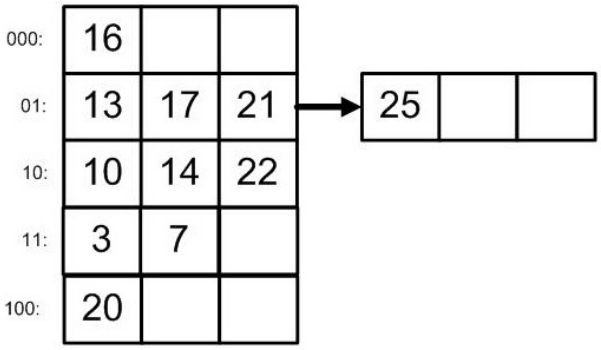
12. Insert 16



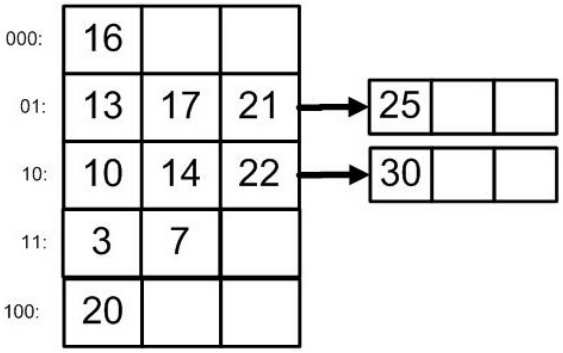
13. Update table



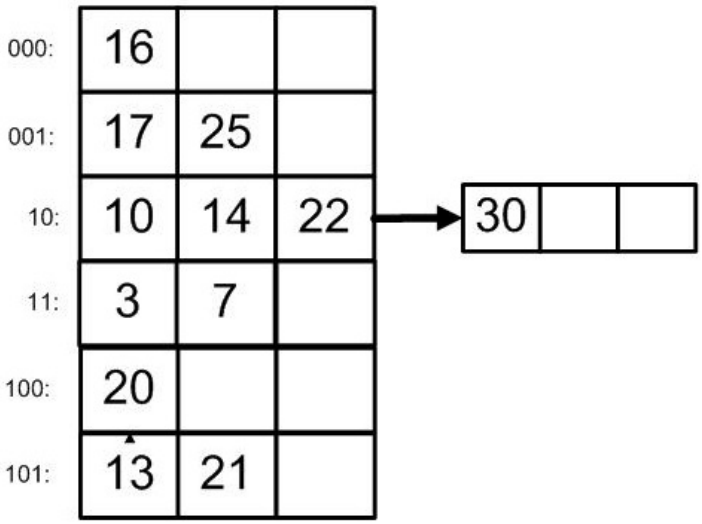
14. Insert 22



15. Insert 30



After updating for the last time, the latest table becomes;



S. 6.

a. When bv = 5 and h = 5;

Number of blocks hashed at level h	= 27
Number of blocks hashed at level h +1	= 10
The binary address of the last bucket of the file	= 100100
The binary address of the last bucket of the file hashed at level 5	= 11111

b. When bv = 0 and h = 5;

Number of blocks hashed at level h	= 32
Number of blocks hashed at level h +1	= 0
The binary address of the last bucket of the file	= 11111
The binary address of the last bucket of the file hashed at level 5	= 11111

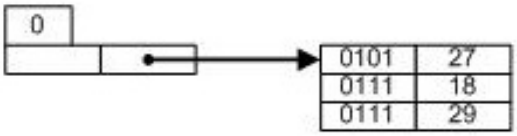
S. 7.

We should first find the binary keys of records including their pseudo keys.

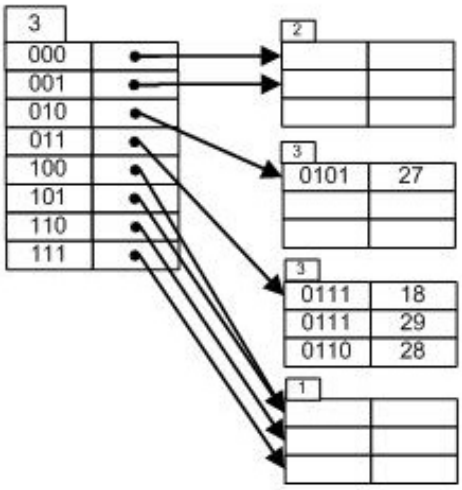
Binary Keys	**Pseudo Keys**
27 = 11011	27 = 0101
18 = 10010	18 = 0111
29 = 11101	29 = 0111
28 = 11100	28 = 0110
42 = 101010	42 = 1001
13 = 1101	13 = 0010
16 = 10000	16 = 0101

Then we will insert the records using these pseudo key values.

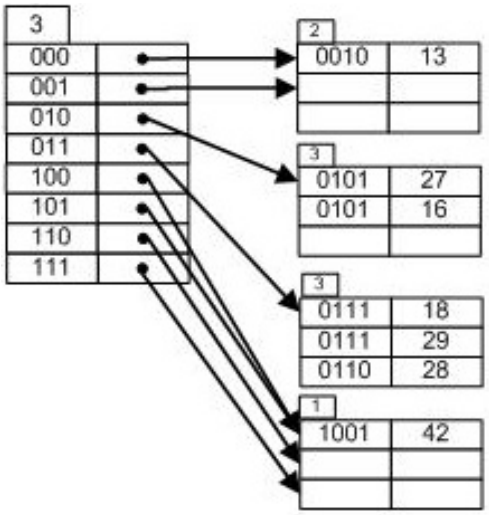
1. **Insert 27, 18 and 29**



2. **Update the table to have more spaces for the coming records.**



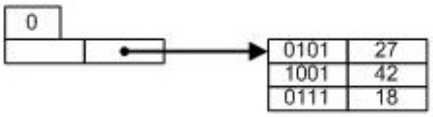
3. **Insert 28, 42, 13 and 16 successively**



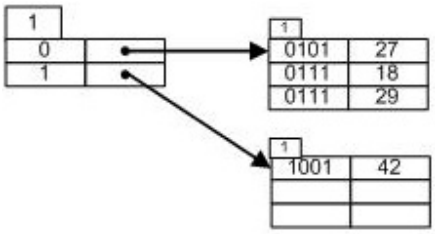
S. 8.

We have already found the binary values in Q. 7. What we will do here is to insert the record with the given order using those key values.

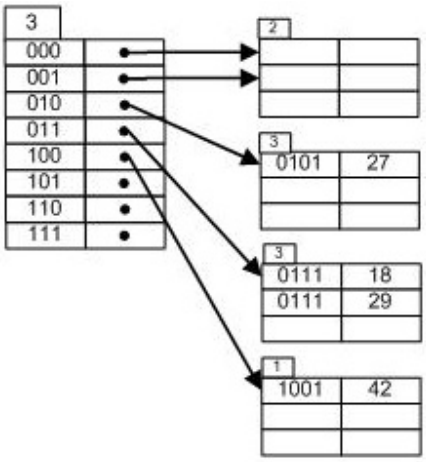
1. **Insert 27, 42, 18**



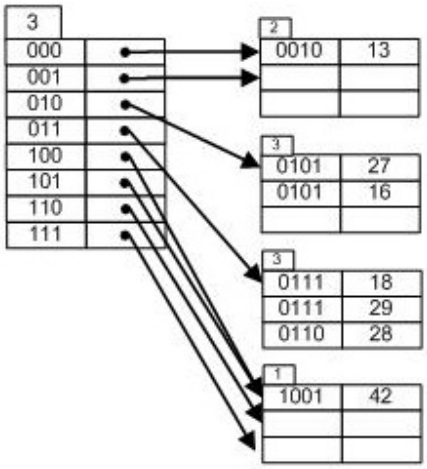
2. **Update table and insert 29**



3. **Update table**



4. **Insert 28, 13, 16**



As it is seen, the resulting table is not changed. However, we need some extra space in memory for 2nd step of the 8th question.