Assignment 6

Harshit Rakesh Shiroiya,

November 2021

3. •Create successively larger sets of n randomly selected integers in the range [1,n]. You can do this using the following function and with the following experiment.[5]

```
create or replace function makeS (n integer)
returns void as
$$
begin
   drop table if exists S;
   create table S (x int);
   insert into S select * from SetOfIntegers(n,1,n);
end;
$$ language plpgsql;
```

This function generates a bag S of size n, with randomly select integers in the range [1, n]. Now consider the following SQL statements:

```
select makeS(10);
explain analyze select x from S;
explain analyze select x from S order by 1;
```

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.014 ms	0.392 ms
10^2	0.029 ms	0.094 ms
10^3	0.176 ms	0.492 ms
10^4	1.643 ms	4.889 ms
10^5	16.339 ms	73.582 ms
10^6	167.428 ms	856.306 ms
10^7	1770.416 ms	12048.246 ms
10^8	18281.931 ms	82083.887 ms

A)

Smaller values, such as 100 and 1000, don't make much of a difference, but the difference between scanning and sorting grows linearly as n grows. The explanation for this could be that for small data sets, all of the data is immediately stored in the main memory, making tasks like scanning and sorting extremely quick. However, when the file size grows, sorting and scanning must rely on external storage. Seek Time, Rotational Delay, and Transfer Time are three characteristics that affect access to external storage. Seek Time is the most expensive task. This could explain why massive input data takes so long to process.

B) The Execution time of the algorithm is linear, O(n).

C) While executing the algorithm and reading the explanatory analyze, we discovered that the QuickSort Algorithm was used, which takes O(n), confirming the time complexity of External Sort.

D)

set work_mem = '64KB';

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.016 ms	0.588 ms
10^2	0.031 ms	0.099 ms
10^3	0.179 ms	1.463 ms
10^4	1.635 ms	8.895 ms
10^5	16.458 ms	121.903 ms
10^6	163.005 ms	799.047 ms
10^7	1744.355 ms	9148.042 ms
10^8	18313.465 ms	128497.407 ms

set work_mem = ‘1GB’;

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.016 ms	0.053 ms
10^2	0.029 ms	0.099 ms
10^3	0.187 ms	0.525 ms
10^4	1.656 ms	4.912 ms
10^5	16.309 ms	54.484 ms
10^6	162.798 ms	576.719 ms
10^7	1868.105 ms	6776.755 ms
10^8	18564.314 ms	95082.045 ms

E)
According to the table above, the average Execution time for Running the Range Query is longer than the time it takes to establish an index. This confirms that SQL sorts smaller input S using the QuickSort Algorithm, whereas larger input S is sorted using the Merge Sort Algorithm.

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.031 ms	0.042 ms
10^2	0.073 ms	1.011 ms
10^3	1.224 ms	4.062 ms
10^4	3.724 ms	64.582 ms
10^5	33.817 ms	452.783 ms
10^6	284.634 ms	3529.724 ms

7.
Block size = 4096 bytes
Block address size = 9 bytes
Block access time (I/O operation) = 10 ms Record size = 150 bytes
Record primary key size = 8 bytes

(a).Specify (in ms) the minimum time to determine if there is a record with key k in the B^+-tree. (In this problem, you can not assume that a key value that appears in an non-leaf node has a corresponding record with that key value.)

Solution:

For calculating the minimum time to determine if there is a record with key k in the B^+tree,

n <= block size – (lblock-address size)/ (block-address size + record key size)

n <= 4096 – (9 / (9 + 8))

$n \leq 240$

New n = n +1
$\therefore$ n = 241

Minimum Time = $(\lceil \log_{241}(10^9) \rceil + 1) * 10 = (4 + 1) * 10 = 50$ms

(b) Specify (in ms) the maximum time to insert a record with key k in the B+-tree assuming that this record was not already in the data file.

Solution:

The max time will be taken when the branching factor is min i.e.,

$\therefore$ [240 / 2] + 1 = 121 / 2)

Therefore, the height will be, $\lceil \log_{121}(10^9 / 2) \rceil = 4$

In order to access record, we need additional block. Total Time = (4 + 1 + 1) * 10 = 60 ms

(c) How large must main memory be to hold the first two levels of the B+-tree? How about the first three levels?

Solution:

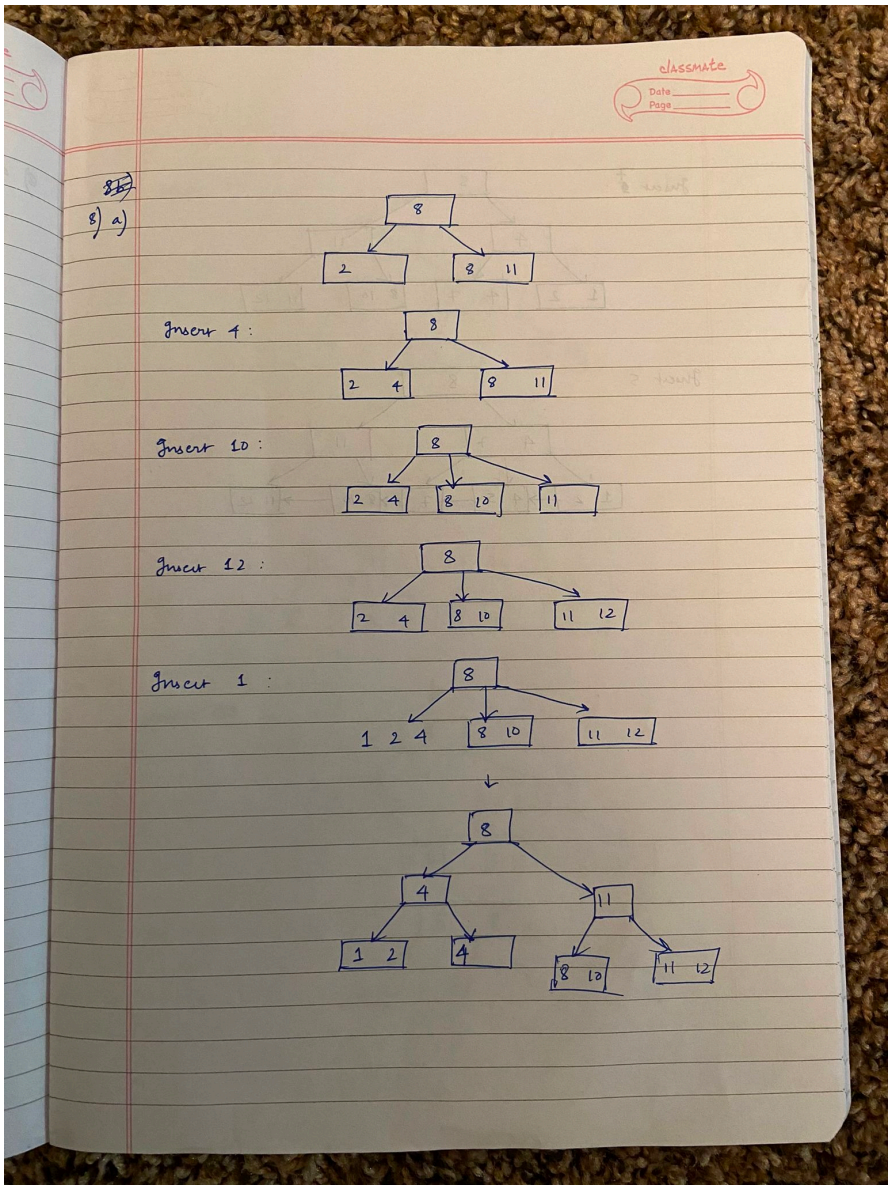
In Main memory,
Two levels = 1 + 240 = 241 blocks

Each block has 241 pointers 240 keys.

Each block size = 241(block size) + block size

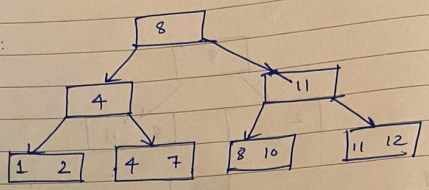
For height 1, 241 * 4096 + 4096 = 991232 bytes ≈ 1MB
For height 2, 241 * 241 * 4096 + 4096 = 237903872 bytes ≈ 238MB

8.

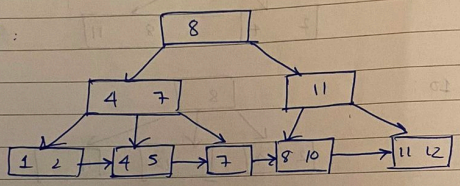


8) b)

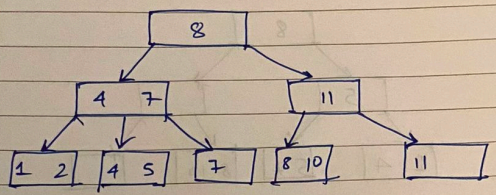
Inserar 7 :



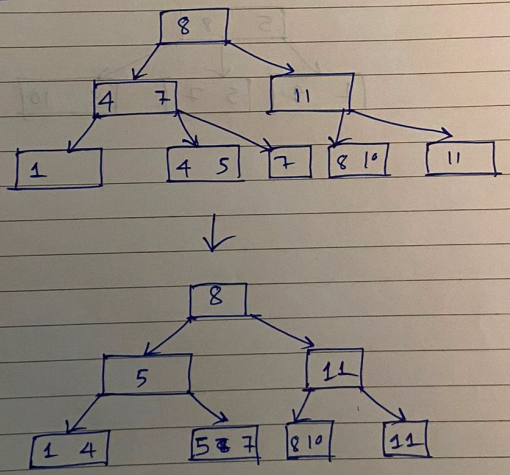
Inserar 5 :



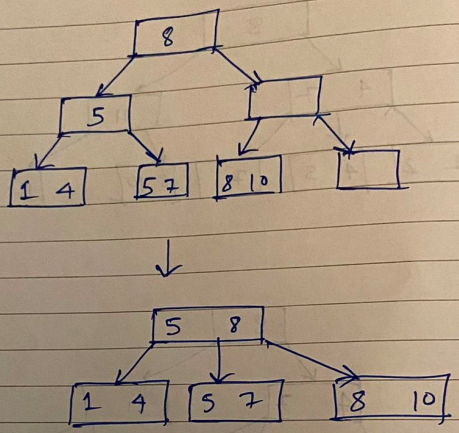
8) b) Delete 12



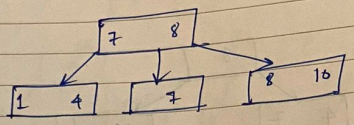
Delete 2



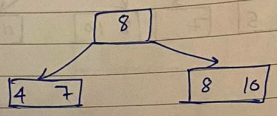
Delete 11 :



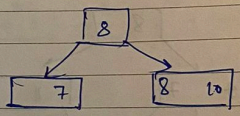
8(c) Delete 5



Delete 1 :

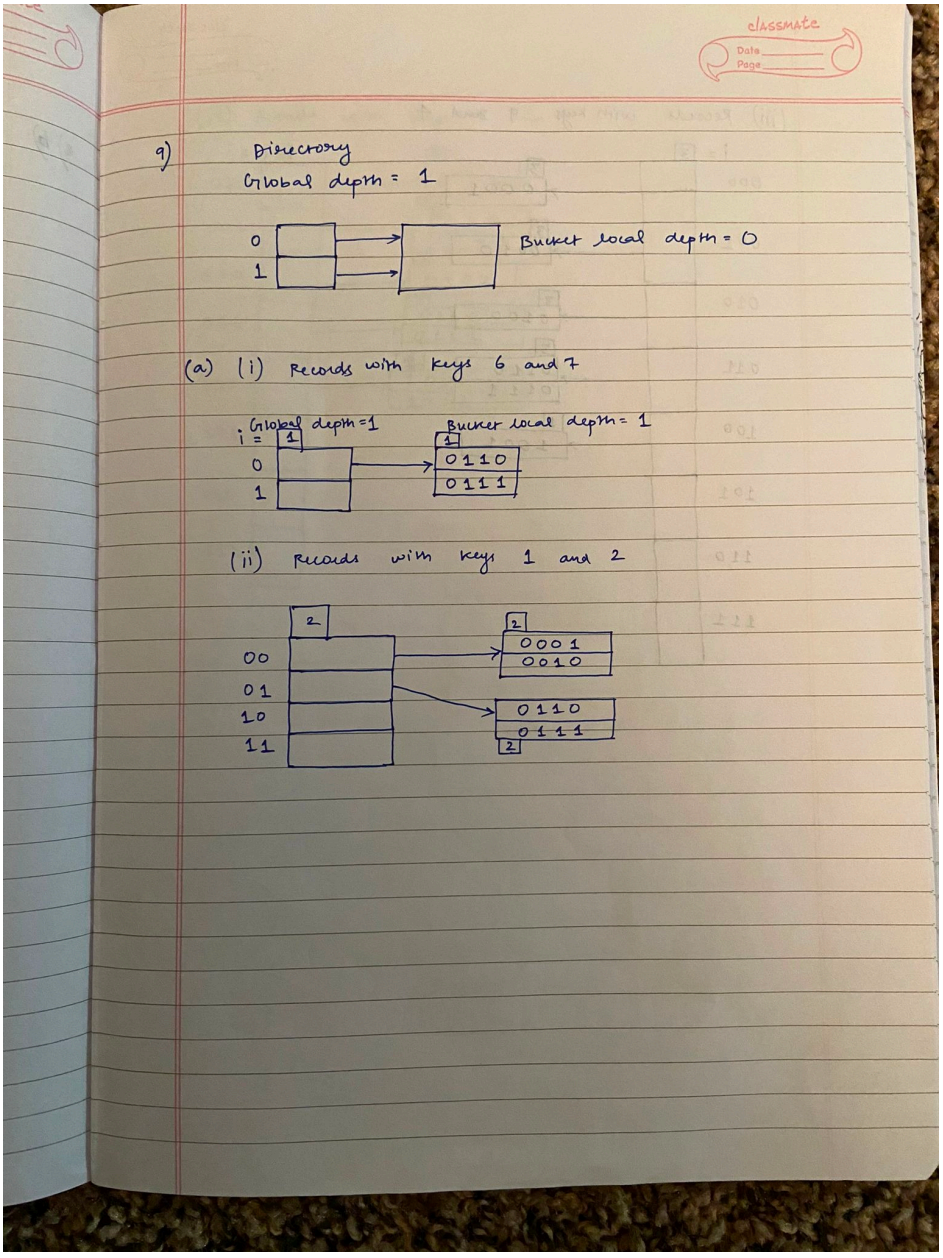


Delete 4 :

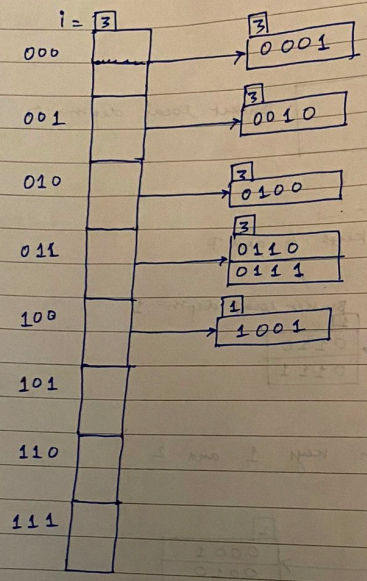


9)

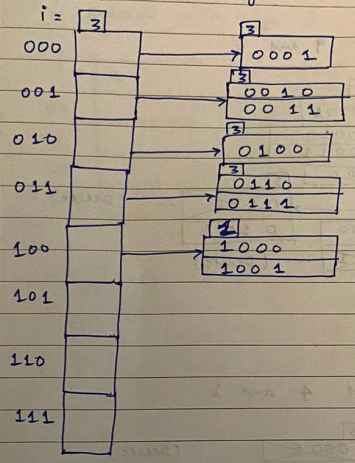
9.


classmate
Date
Page
9)
Directory
Global depth = 1
0
1
Bucket local depth = 0
(a) (i) Records with keys 6 and 7
Global depth = 1
i =
1
0
1
Bucket local depth = 1
1
0 1 1 0
0 1 1 1
(ii) Records with keys 1 and 2
2
00
01
10
11
2
0 0 0 1
0 0 1 0
0 1 1 0
0 1 1 1
2

(iii) Records with keys 9 and 4

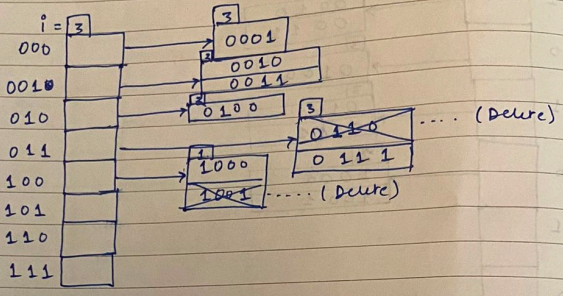


(iv) Records with keys 8 and 3

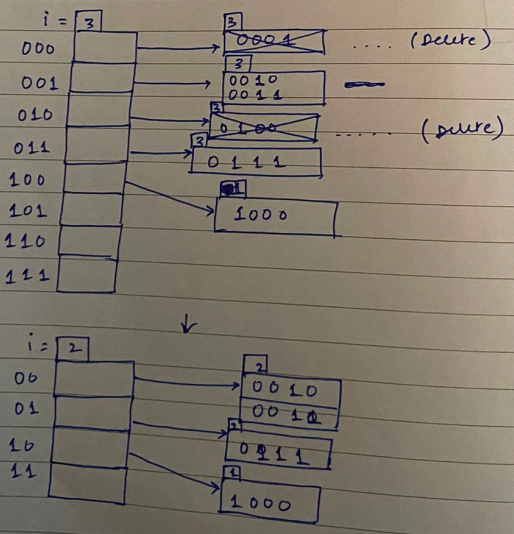


a)

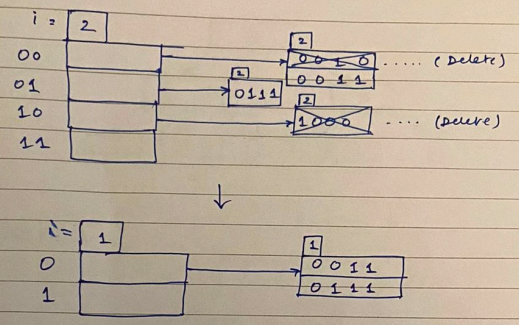
b) (i) Records with keys 9 and 6



(ii) Records with keys 4 and 1



(iii) Records with keys 2 and 8



10. create index cname_index on worksFor using btree (cname);

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.012 ms	0.025 ms
10^2	0.031 ms	0.025 ms
10^3	0.202 ms	0.172 ms
10^4	1.965 ms	1.637 ms
10^5	19.731 ms	16.604 ms
10^6	120.178 ms	90.185 ms

11. create index salary_index on worksFor using btree (salary);

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.014 ms	0.010 ms
10^2	0.033 ms	0.026 ms
10^3	0.233 ms	0.458 ms
10^4	2.253 ms	1.079 ms
10^5	22.568 ms	16.604 ms
10^6	150.384 ms	122.818 ms

12. create index cname_salary_index on worksFor using btree (cname,salary);

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.230 ms	0.026 ms
10^2	0.025 ms	0.022 ms
10^3	0.185 ms	0.029 ms
10^4	1.729 ms	0.075 ms
10^5	17.148 ms	0.856 ms
10^6	79.222 ms	10.076 ms

13create index pid_index on Person using btree (pid);
create index cname_index on worksFor using btree (cname);

A) NOT IN

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.230 ms	0.026 ms
10^2	0.123 ms	0.090 ms
10^3	0.791 ms	0.626 ms
10^4	7.487 ms	5.577 ms
10^5	77.511 ms	58.839 ms
10^6	622.562 ms	568.291 ms

B) IN

size n of relation S	avg execution time to scan S (in ms)	avg execution time to sort S (in ms)
10^1	0.023 ms	0.379 ms
10^2	0.025 ms	0.022 ms
10^3	0.393 ms	0.375 ms
10^4	3.675 ms	3.664 ms
10^5	37.943 ms	38.611 ms
10^6	201.227 ms	195.577 ms