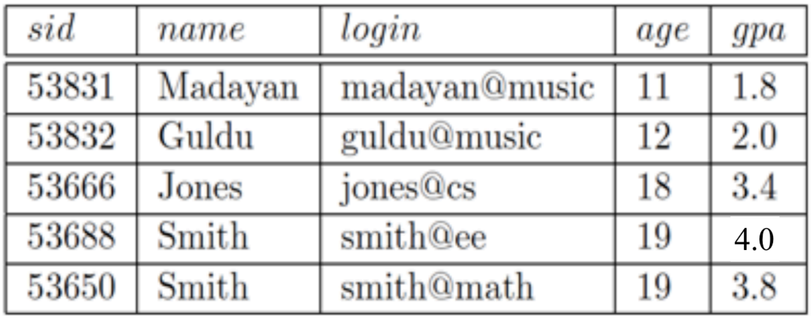
Homework3 Solution

CSC 675/CSC 775
Total points: 100

Q1. (8 points, 1 point each) Select either True or False for each of the following questions.

a) Each relation can have multiple clustered indexes. False
b) A relation can have a clustered index on several attributes. True
c) Each relation can have multiple indexes using Alternative1. False
d) Maintaining clustered indexes are more expensive than sorted files. False
e) Index-olny plans work more efficiently with clustered indexes than unclustered indexes. False
f) In any clustered index file the data records are sorted by the search keys. True
g) In any index file the data entries are sorted by the search keys. True
h) Adding an index to a relation makes all operations on the relation faster. False

Q2. (25 points, 5 points each) Consider the following instance of the Students relation, sorted by age. Assume that these tuples are stored in a sorted file in the order shown; the first tuple is on page 1, the second tuple is also on page 1; and so on. Each page can store up to three data records; so the fourth tuple is on page 2. Explain the data entries that each of the following indexes contain. If such an index cannot be constructed, provide reason.



sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	19	4.0
53650	Smith	smith@math	19	3.8

a) An unclustered index on <age, gpa> using Alternative (2).
b) A clustered index on age using Alternative (2).
c) A clustered index on <age, gpa> using Alternative (2).
d) A clustered index on age using Alternative (1).
e) An unclustered index on name using Alternative (2).

Solution

(5 points each)

a) <(11,1.8), (1,1) >, <(12,2.0), (1,2) >, <(18,3.4), (1,3) >, <(19,3.8), (2,2) >, <(19,4.0), (2,1) >.
b) <11, (1,1) >, <12, (1,2) >, < 18, (1,3) >, < 19, (2,1) >, <19, (2,2) >.
c) Such index cannot be ***clustered*** because the records in the file are sorted by age, therefore they can not be sorted by <age,gpa>.
d) <tuple 1>, <tuple 2>, <tuple 3>, <tuple 4>, <tuple 5>.
e) < Guldu, (1,2) >, < Jones, (1,3) >, < Madayen, (1,1) >, < Smith, (2,1) >, < Smith, (2,2) >.

Q3. (8 points, 4 points each) Consider these relations:

Emp(eid: integer, ename: varchar, sal: integer, age: integer, did: integer)
Dept(did: integer, budget: integer, floor: integer, mgr eid: integer)

Salaries range from $10,000 to $100,000, ages vary from 20 to 80, each department has about five employees on average, there are 10 floors, and budgets vary from $10,000 to $1 million. For each of the following queries, which of the listed index choices would you choose to speed up the query?

1. Query: Print ename, age, and sal for all employees.
(a) Clustered hash index on <ename, age, sal> fields of Emp.
(b) Clustered B+ tree index on <ename, age, sal> fields of Emp.
(c) Unclustered hash index on <ename, age, sal> fields of Emp.
(d) Unclustered hash index on <eid, did> fields of Emp.

2. Query: Find the dids of departments that are on the 10th floor and have a budget of less than $15,000.
(a) Clustered B+ tree index on <floor, budget> fields of Dept.
(b) Clustered hash index on the floor field of Dept.
(c) Unclustered hash index on the floor field of Dept.
(d) Clustered B+ tree index on the budget field of Dept.

Solution

1. If our system include index-only plans then **c** is correct: an unclustered hash index on <ename, age, sal> fields of Emp.

2. **a** is correct: a clustered B+ tree index on <floor, budget> fields of Dept

Q4. (5 points) Consider this schema:

Student (username , firstname, lastname, year)

Which index would be most useful in reducing processing time of this query?

SELECT S. lastname FROM Student S WHERE 3 > year AND year > 1

a) Hashtable on Student(year)
b) Hashtable on Student(year, lastname)
c) Hashtable on Student(lastname, year)
d) B+ tree on Student(year)
e) B+ tree on Student(year, lastname)
f) B+ tree on Student(lastname, year)

Solution:
e is correct, B+ tree on Student(year, lastname)

Q5. (18 points, 6 points each part) Suppose we are building a B-Tree. Pages are 4008 bytes. A key value takes 8 bytes, and a pointer to a tree node or a record takes 8 bytes.

P_0	K_1	P_1	K_2	P_2	⋄ ⋄ ⋄	K_m	P_m

a) (6 points) What is the maximum number of keys in each node?

Solution:

$8(m+1)+ 8m <=4008$

$16m<=4000$

$m=250$

Grading guide:

-3 points: wrong method

-1 point: wrong number

b) (6 points) Calculate the maximum number of records N, which can be indexed with an index file with height=2 and m keys (m is calculated in part a)?

Solution:

#Indexed Records = #Leaf nodes * F

#Leaf nodes = $F^h = (m+1)^2 = 251^2 = 63{,}001$ leaf pages

#Records can be indexed = $63{,}001 * (m+1) = 15{,}813{,}251$

Grading guide:

-3 points: wrong method

-1 point: wrong number

c) (6 points) How many I/O operations are required for searching through N records using a sorted file (N is calculated in part b)?.

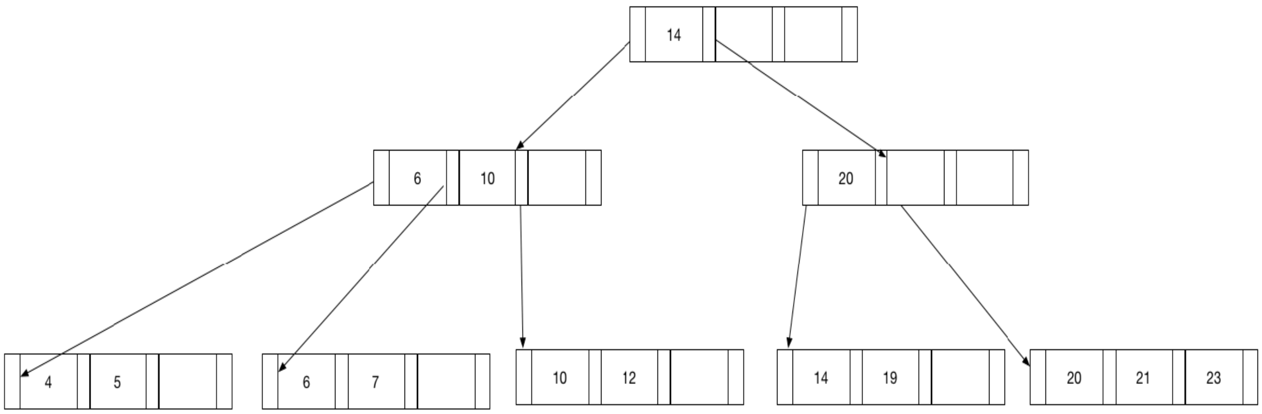
Solution: $\text{Log}_2 15{,}813{,}251 \sim 24$ I/O

Grading guide:

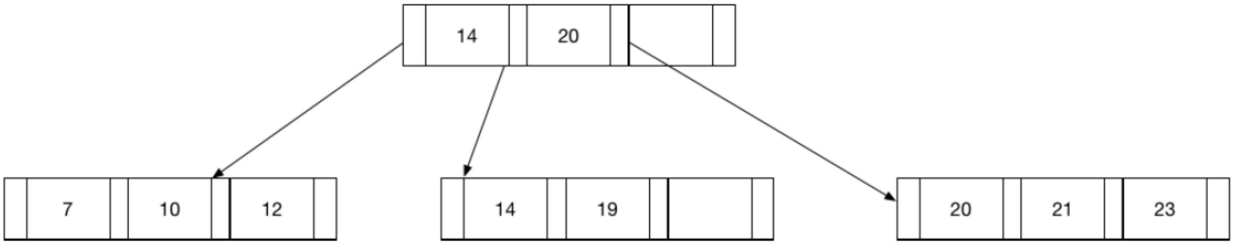
-3 points: wrong method

-1 point: wrong number

Q6. (12 points) Consider a B+ tree with nodes up to 3 keys (4 pointers). Assume the left order is for < and the right order is for >=. If you can borrow from both siblings, choose the right sibling. Show the B+ tree after deleting values 4, 5, and 6 from the B+ tree.



Solution:

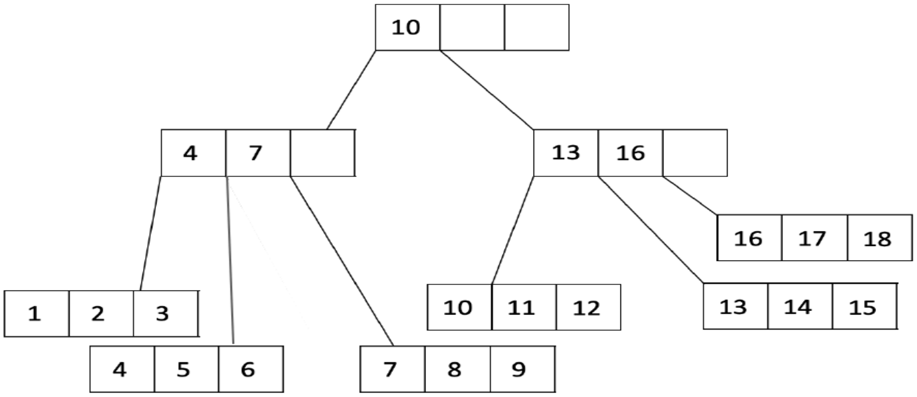


Grading guide:

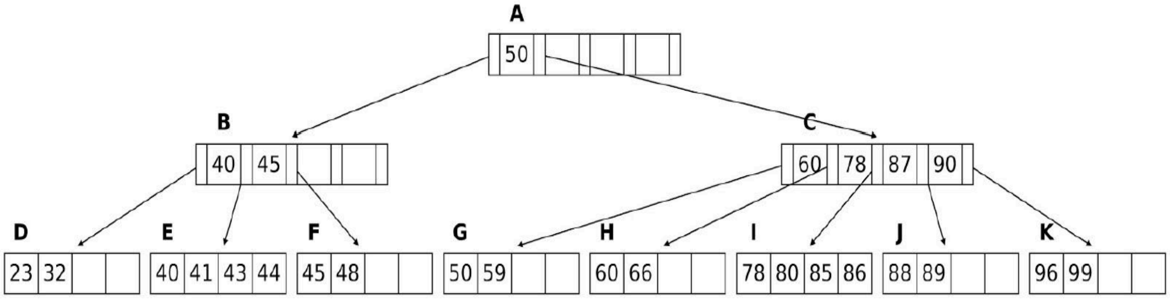
-1 points: for each wrong/missing number in the tree nodes,
-2 points for a wrong/missing split operation,
-2 points for a wrong/missing merge operation,
-2 points: for wrong number of leaf pages,
-2 points for wrong number of levels/height,

Q7. (12 points) Consider a B+ tree with nodes up to 3 keys (4 pointers). Assume the left order is for < and the right order is for >=. Use Bulk-loading to create a B+ tree with values 1 to 18. Show the final tree.

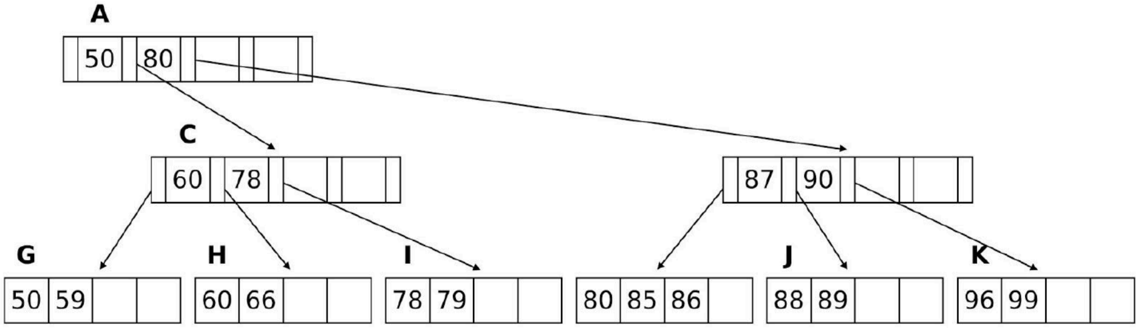
Solution:



Q8. (12 points) Consider the following B+ tree. Draw the new B+ tree after inserting a data entry with key 79.



***Solution:** The following tree shows the nodes that are changed.*



The other nodes of the tree have not been changed.

Grading guide:

-1 points: for each wrong/missing number in the tree nodes,
-2 points for a wrong/missing split operation,
-2 points for a wrong/missing merge operation,
-2 points: for wrong number of leaf pages,
-2 points for wrong number of levels/height,