CSE 444 – Homework 2
Indexes and Operator Algorithms

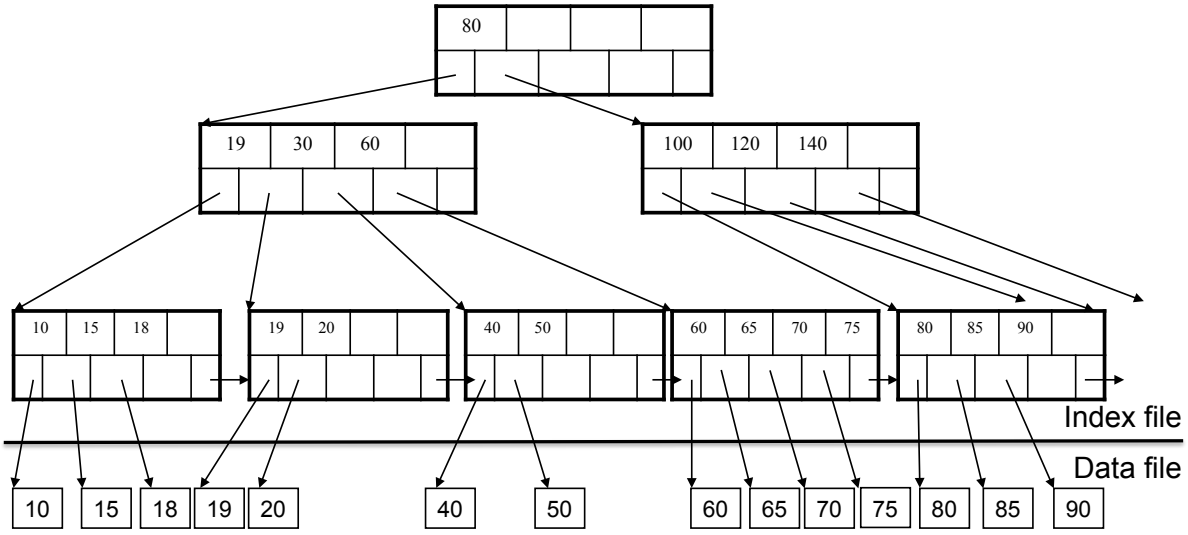
Name: Eric Boris

Question	Points	Score
1	15	
2	15	
3	10	
Total:	40	

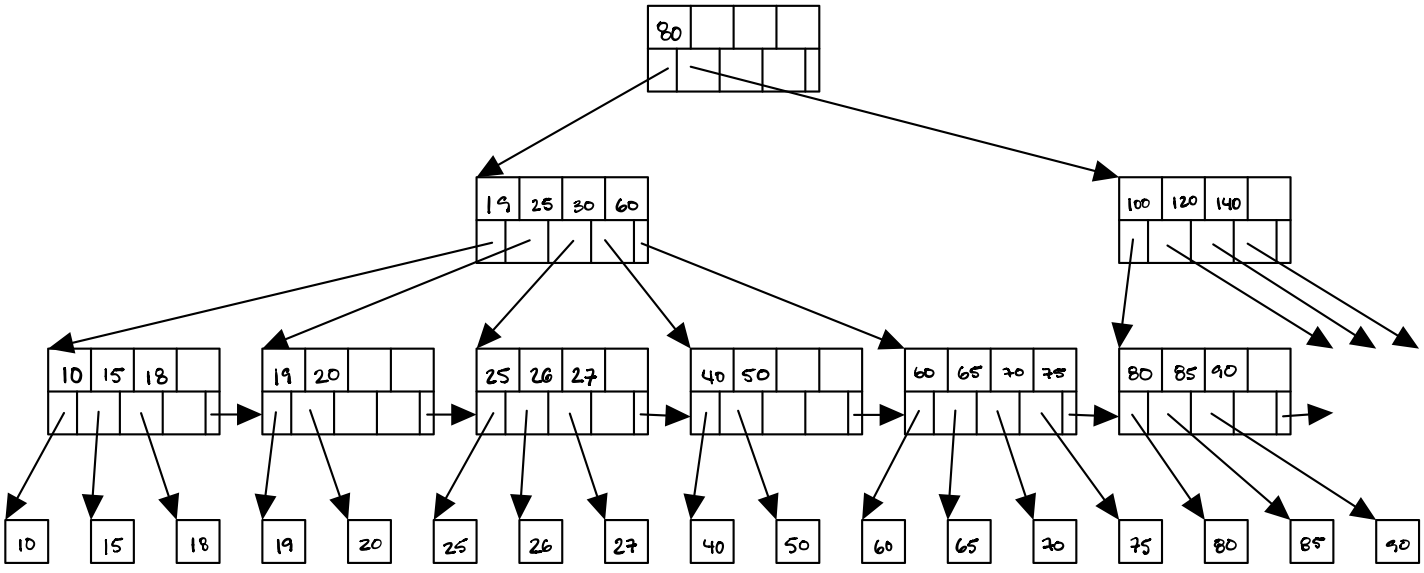
1 B+ Trees

1. (15 points)

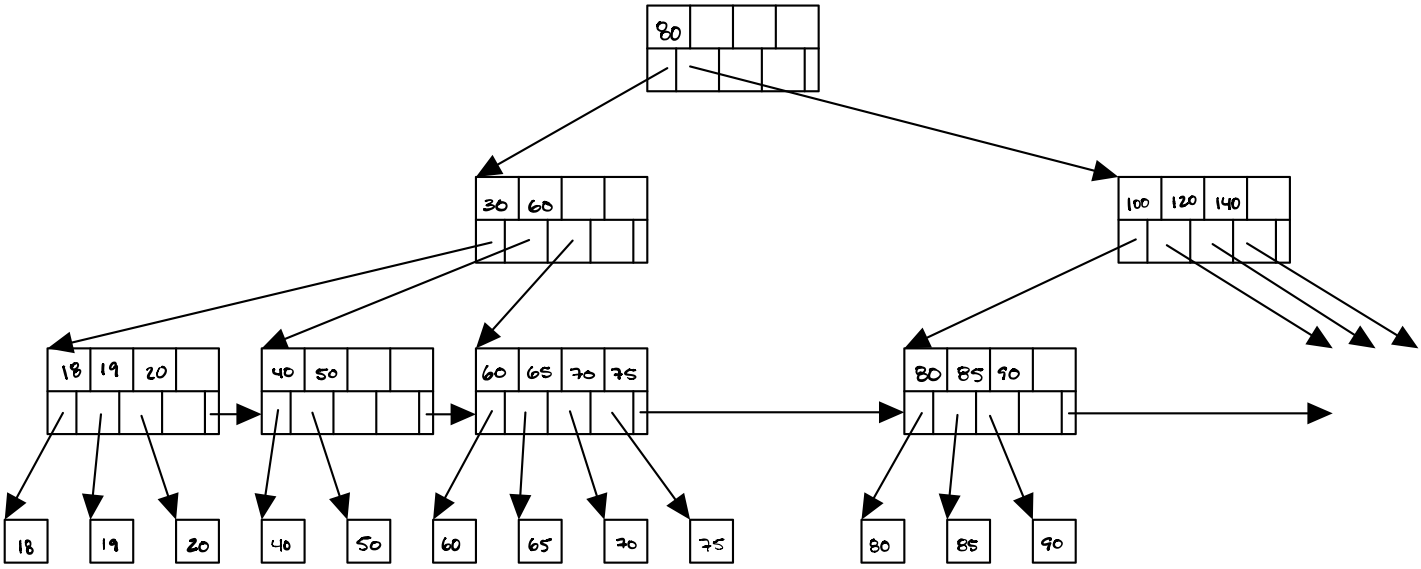
   Consider the following B+ tree index:



   (a) (5 points) Draw the modified tree after the insertion of tuples with search key values 25, 26, and 27 into the base relation and index. Draw the final tree only.



(b) (5 points) Consider the *original* tree again. Draw the modified tree after the deletion of tuples with search key values 10 and 15. Draw the final tree only.



(c) (5 points) Consider the original tree again. How many pages will be read from disk to answer each of the following queries. Assume that the buffer pool is empty before each query executes, and that it is large enough to hold the entire database. Assume also that 4 tuples from the base relation fit on one page in the data file and that all pages are full:

- Look up all tuples with search key value of 40 if the index is *clustered*.
- Look up all tuples with search key value of 40 if the index is *unclustered*.
- Look up all tuples with search key value in the range $[60, 90]$ if the index is *clustered*.
- Look up all tuples with search key value in the range $[60, 90]$ if the index is *unclustered*. Assume the worst-case scenario in terms of how the keys are stored in pages.

- 4
- 4
- 7
- 11

2 Operator Algorithms

2. (15 points)

Consider two relations R and S with the following sizes:

Relation	# Blocks	# Tuples
R	100	1000
S	80	800

(a) (5 points) Compute the cost of joining these two relations in each of the following two scenarios:

- Using a tuple-at-a-time nested loop join algorithm with R as the outer relation.
- Using a page-at-a-time nested loop join algorithm with R as the outer relation.

Tuple at a time

$B(R) + T(R)B(S) = 100 + 1000 \cdot 80$

$= 80,100$

Page at a time

$B(R) + B(R)B(S) = 100 + 100 \cdot 80$

$= 8,100$

Relation	# Blocks	# Tuples
R	100	1000
S	80	800

(b) (5 points) Assume the join operator is allowed to use $M = 10$ pages of memory. What is the least-cost method that we could use to join the two relations with a nested-loop type of join algorithm? Describe the method for the R and S relations and compute its cost.

Method:

for each group of M-1 pages s in S do:
  for each page of tuples r in R do:
    for all pairs of tuples t_1 in s, t_2 in r:
      if t_1 and t_2 join then output (t_1, t_2)

Describe:

1. Load M-1 pages of S at a time (leaving 1 page free for R).
2. For each M-1 page segment of S, load each page of R.
3. Check against the join conditions.

Cost:

$$B(S) + \frac{B(S)B(R)}{M-1} = 80 + \frac{80 \cdot 100}{10-1}$$

$$= 968.8$$

$$= 969$$

(c) (5 points) Now consider that the join is a primary-key to foreign-key join and that S is indexed on the join attribute but R is not. Describe how R and S can be joined using an index-based nested loop join algorithm. Assume that all index pages are in memory. Indicate the cost assuming each tuple of R joins with exactly one tuple in S.

$B(R) = 100, \quad T(R) = 1000$

$B(S) = 80, \quad T(S) = 800, \quad V(S, a) = 800$ ← since indexed on primary key

$$B(R) + \frac{T(R)T(S)}{V(S,a)} = 100 + \frac{1000 \cdot 800}{800}$$

$$= 1100$$

We know S is indexed on the join attribute.

Join Attribute is primary key so $V(S,a) = T(S) = 800$

Iterate over R.

For each tuple of R, fetch corresponding tuple of S.

Each Tuple of R joins to exactly one Tuple of S.

Hence cost is that of iterating over each block of R and each Tuple of R.

3 Multi-Pass Algorithms

3. (10 points)

Consider again the relations R and S from the previous question but this time **assume they have 10 times as many pages. That is, assume** $B(R) = 1000$ **and** $B(S) = 800$. Explain how a DBMS could efficiently join these two relations given that only **11 pages** can fit in main memory at a time. Your explanation should be detailed: specify how many pages are allocated in memory and what they are used for; specify what exactly is written to disk and when. Compute the cost of the join operation.

(a) (5 points) Present a solution that uses a hash-based algorithm.

Proceed in 2 phases:
- **Partition phase:**
  - **Given M=11 and 1000 pages, reserve 1 page for input buffer with M=10 remaining for partitioning.**
  - **M=10 is insufficient to hold 1000 pages, execute Partition phase in 2 Passes:**
    - **Pass 1: Hash 1000 pages into 100 partitions via M=10, 1000/10=100.**
    - **Pass 2: Hash each 100 partitions via M=10, 100/10=10.**
  - **Pass 1:**
    - **Iterate over each page of R.**
      - **Load each page into memory.**
      - **Hash each tuple in the page using a mod function.**
      - **Put each tuple in the resultant buffer bucket.**
      - **Output to disk when the bucket is full.**
    - **Repeat the above for S using the same mod function.**
    - **Cost:**
      - **B(R): 1 read and 1 write.**
      - **B(S): 1 read and 1 write.**
      - **2B(R) + 2B(S)**
  - **Pass 2:**
    - **Iterate over each partition of R.**
      - **Load each page of partition into memory.**
      - **Hash each tuple in the page using a different mod function from Pass 1.**
      - **Put each tuple in the resultant buffer bucket.**
      - **Output to disk when the bucket is full.**
      - **Output remaining pages to disk when finished with pages in partition.**
    - **Repeat the above for S using the same mod function.**
    - **Cost:**
      - **B(R): 1 read and 1 write.**
      - **B(S): 1 read and 1 write.**
      - **2B(R) + 2B(S)**
- **Probe phase:**
  - **Iterate over each partition of R.**
    - **Sort the partition using a unique mod function.**
    - **Use the result to get the corresponding partition of S.**
    - **Join.**
    - **Output the join to disk or stream.**
  - **Cost:**
    - **B(R): 1 read.**
    - **B(S): 1 read.**
    - **B(R) + B(S)**

Total Cost:
- **5B(R) + 5B(S) = 5(1000) + 5(800) = 9000**

(b) (5 points) Present a solution that uses a sort-merge-based algorithm.

Proceed in 4 passes:
- **Pass 1:**
  - **Iterate over R.**
    - **Load 11 pages into memory.**
    - **Sort into a run.**
    - **Write to disk.**
  - **Results in 91 runs.**
  - **Repeat the above for S.**
  - **Results in 73 runs.**
  - **Total 164 runs.**
    - **Insufficient memory to hold all runs.**
    - **Must merge runs into larger runs to fit into memory.**
  - **Cost:**
    - **B(R): 1 read and 1 write.**
    - **B(S): 1 read and 1 write.**
    - **2B(R) + 2B(S)**
- **Pass 2:**
  - **Iterate over runs of R.**
    - **Load M-1 pages of each run into memory.**
    - **Sort into larger runs.**
    - **Write to disk.**
  - **Results in 10 larger runs.**
  - **Repeat the above for S.**
  - **Results in 8 larger runs.**
  - **Total 18 larger runs.**
    - **Insufficient memory to hold all runs.**
    - **Must merge larger runs into largest runs to fit into memory.**
  - **Cost:**
    - **B(R): 1 read and 1 write.**
    - **B(S): 1 read and 1 write.**
    - **2B(R) + 2B(S)**
- **Pass 3:**
  - **Iterate over larger runs of R.**
    - **Load M-1 larger runs into memory.**
    - **Sort into largest run.**
    - **Write to disk.**
  - **Results in 1 run.**
  - **Repeat the above for S.**
  - **Results in 1 run.**
  - **Total 2 runs.**
    - **Sufficient memory to hold all runs.**
  - **Cost:**
    - **B(R): 1 read and 1 write.**
    - **B(S): 1 read and 1 write.**
    - **2B(R) + 2B(S)**
- **Pass 4:**
  - **Iterate over the largest runs of R.**
    - **Iterate over the largest runs of S.**
      - **Join.**
      - **Output the join to disk or stream.**
  - **Cost:**
    - **B(R): 1 read.**
    - **B(S): 1 read.**
    - **B(R) + B(S)**

Total Cost:
- **7B(R) + 7B(S) = 7(1000) + 7(800) = 12600**