**ComS 363 Spring 2022**

**Class Participation**

**Learning objective:**

1. Gain a deeper understanding of how indexes and relations are related.
2. Gain a deeper understanding of the search algorithm for B+Tree.
3. Explore join algorithms and disk I/O costs.

**Instruction:**

Answer all the sub-questions for Question 1 and Question 2.a-c to get your full credits for course participation. There is no partial credit for class participation.

**Questions:**

1. Suppose the below diagram is a dense B+tree index created on the attribute id of the relation R(id, name) where id is the primary key of R and id is an integer. Suppose data entry format 2 (search key value, record id) is used. A dense index means that there is one data entry per tuple. Recall that the record id has information about which page in a file (storing the relation) and the slot number where the tuple is located in that page. Recall that a dense index means that there is one data entry in a leaf page per tuple.

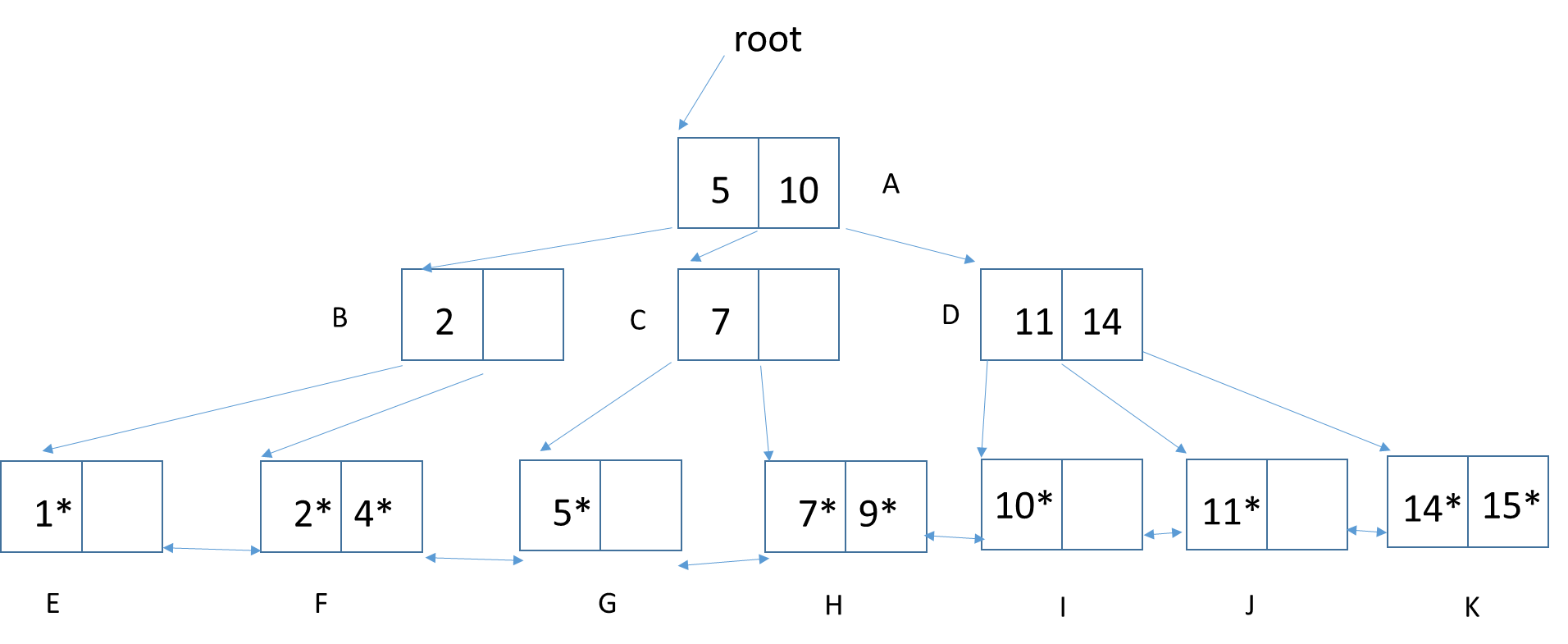
**Notation**: The notation i\* on a leaf page represents the search key value of i followed by the record id to the tuple in the relation R. Answer the following questions. You can use the labels A, B, C, …, K to represent the tree nodes in your answers for 1.c)-1.e)

Figure 1. Dense B+Tree index

1. How many tuples are in the relation R?

Answer: 10 tuples since there are 10 data entries (in the leaf pages).

1. What is the order of this tree?

Answer: 2/2 = 1

1. List all the nodes that must be examined to find R.id=6.

Answer: A, C, G

1. List all the nodes that must be examined to find R.id < 10.

Answer: A, C, H, G, F, E

1. List all the nodes that must be examined to find R.id R.id > 2 Answer:A, B, F, G, H, I, J, K

**Question 2:** The instances of the food relation and the recipe relation are below. The underlined attribute(s) indicate the primary key.

Recipe. Recipe.fid is a foreign key to food.fid.

| fid | iid | amount |
| --- | --- | --- |
| 1 | 1 | 50g |
| 1 | 2 | 60g |
| 2 | 1 | 20g |
| 3 | 1 | 10g |
| 4 | 3 | 100g |

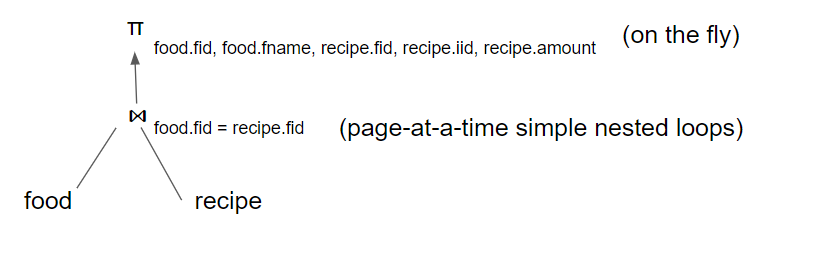
food

| fid | fname |
| --- | --- |
| 1 | Pizza |
| 2 | Hummus |
| 3 | BBQ |
| 4 | Noodle |

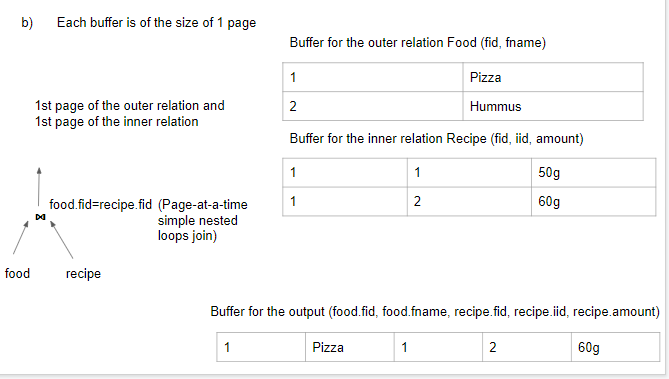
**Assumption:** Each disk page is large enough to store at most two rows of the food table at a time or two rows of the recipe table at a time. However, a page can only store at most one row of the join result of the above query at a time. Assume that each page stores as many rows as it can.

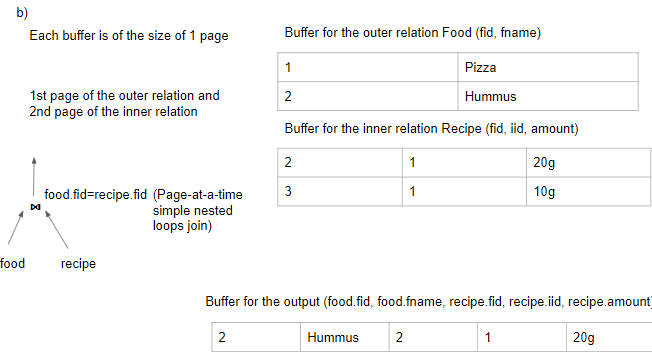
select \* from food inner join recipe on food.fid=recipe.fid

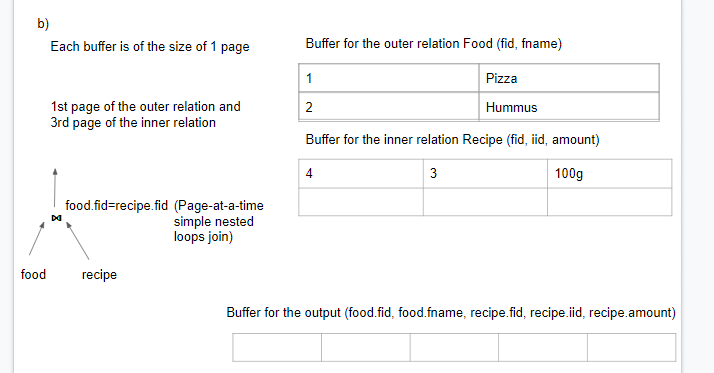
1. Draw a query execution plan for the above query using the page-at-a-time simple-nested loops join algorithm. Let food be the outer relation and recipe be the inner relation.

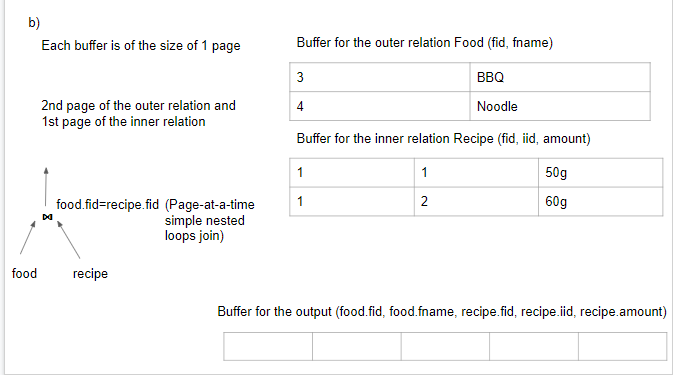


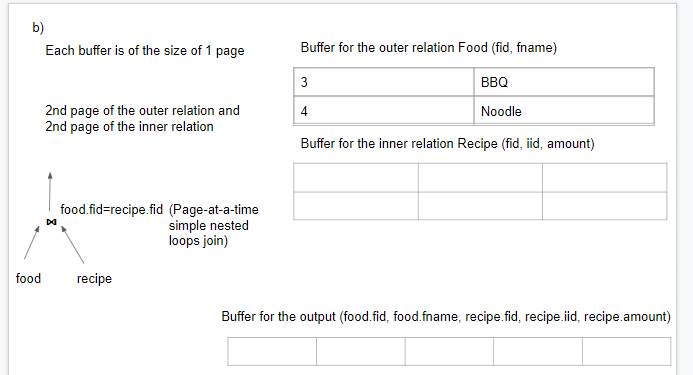
1. Work through the page-at-a-time simple-nested loops join algorithm for the query using the above instances of food and recipe relations when food is the outer relation.











1. Calculate the disk I/O cost when using the page-at-a-time simple nested loops join algorithm when food is the outer relation and the recipe relation is the inner relation.

Disk I/O cost = |Food| + |Food|\*|Recipe| = 2 +2\*3 = **8**

1. Estimate the disk I/O cost when using the block nested loops join algorithm and the food relation is the outer relation and the recipe relation is the inner relation. Suppose the total database memory buffer to perform the join operation is 4 pages.

Disk I/O cost = 2+3=5 = |Food| + ||Recipe|\*ceiling( 2/(4-2))

1. Draw a query execution plan for the above query using the indexed nested loops join algorithm. Let the food relation be the outer relation and the recipe relation be the inner relation. There are indexes on the primary key and each foreign key whose attribute is not the first attribute of the primary key.

