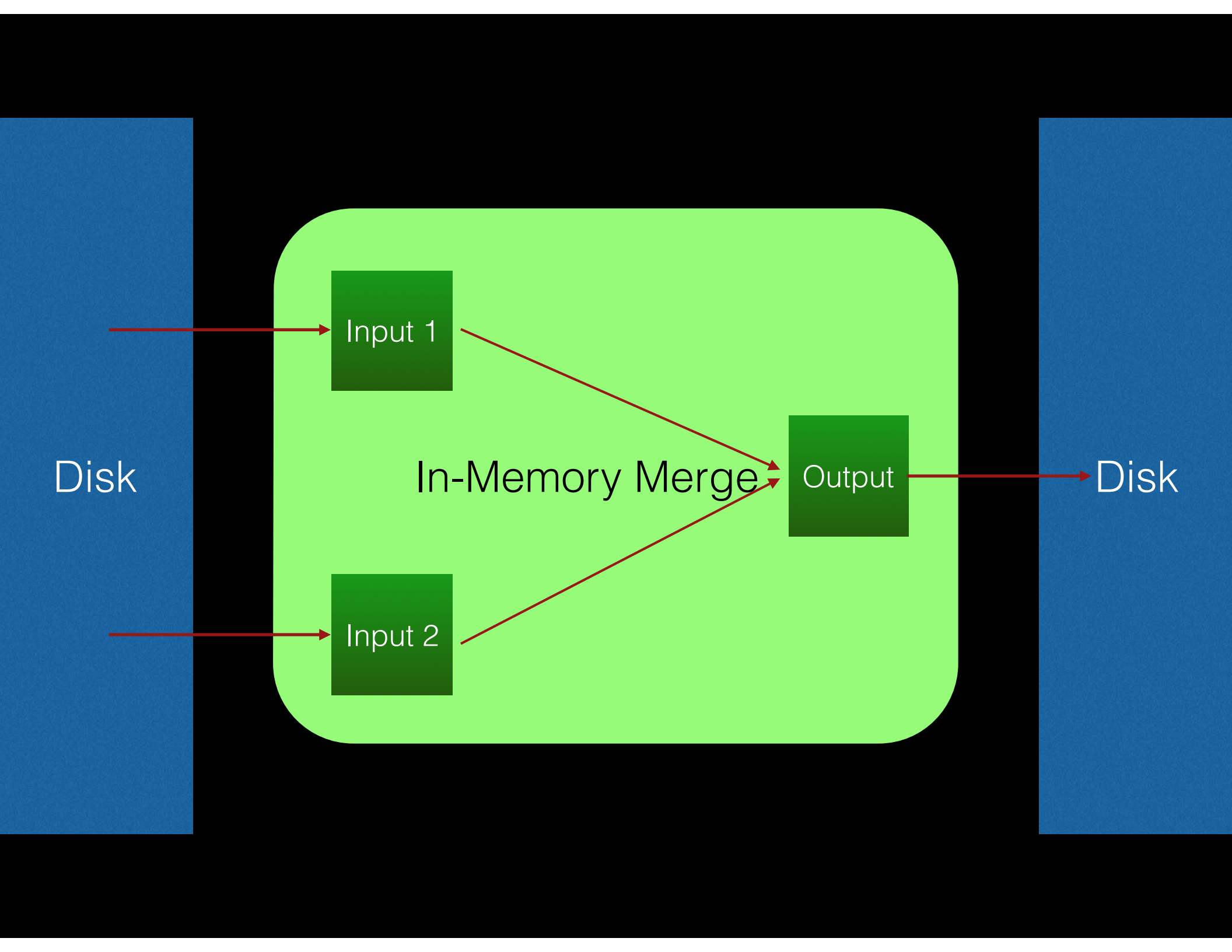
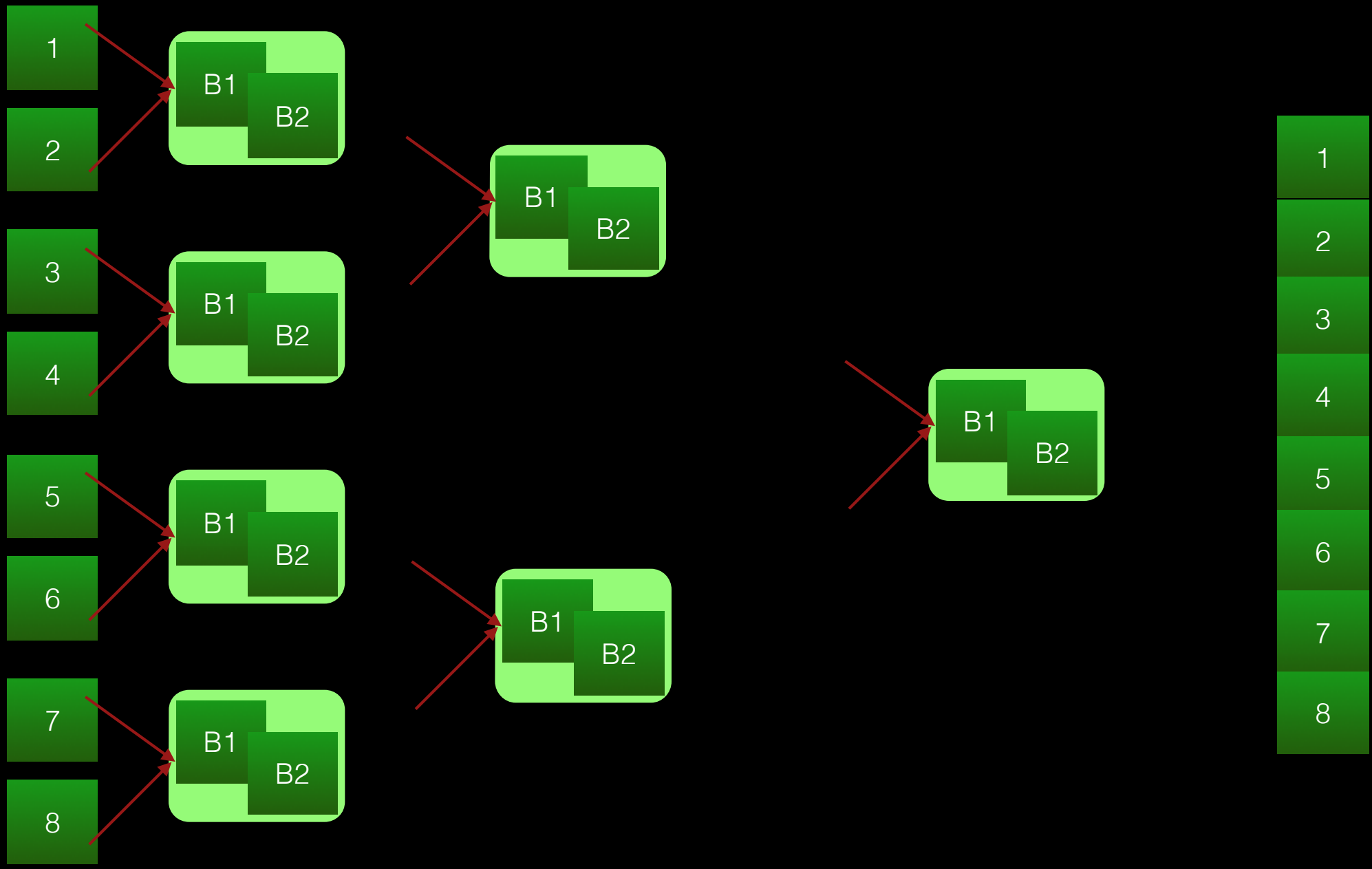
CS 186 Section 2: Out-of-Core algorithms and Single-Table SQL

Vikram Sreekanti

External Sorting


Disk
Input 1
Input 2
In-Memory Merge
Output
Disk

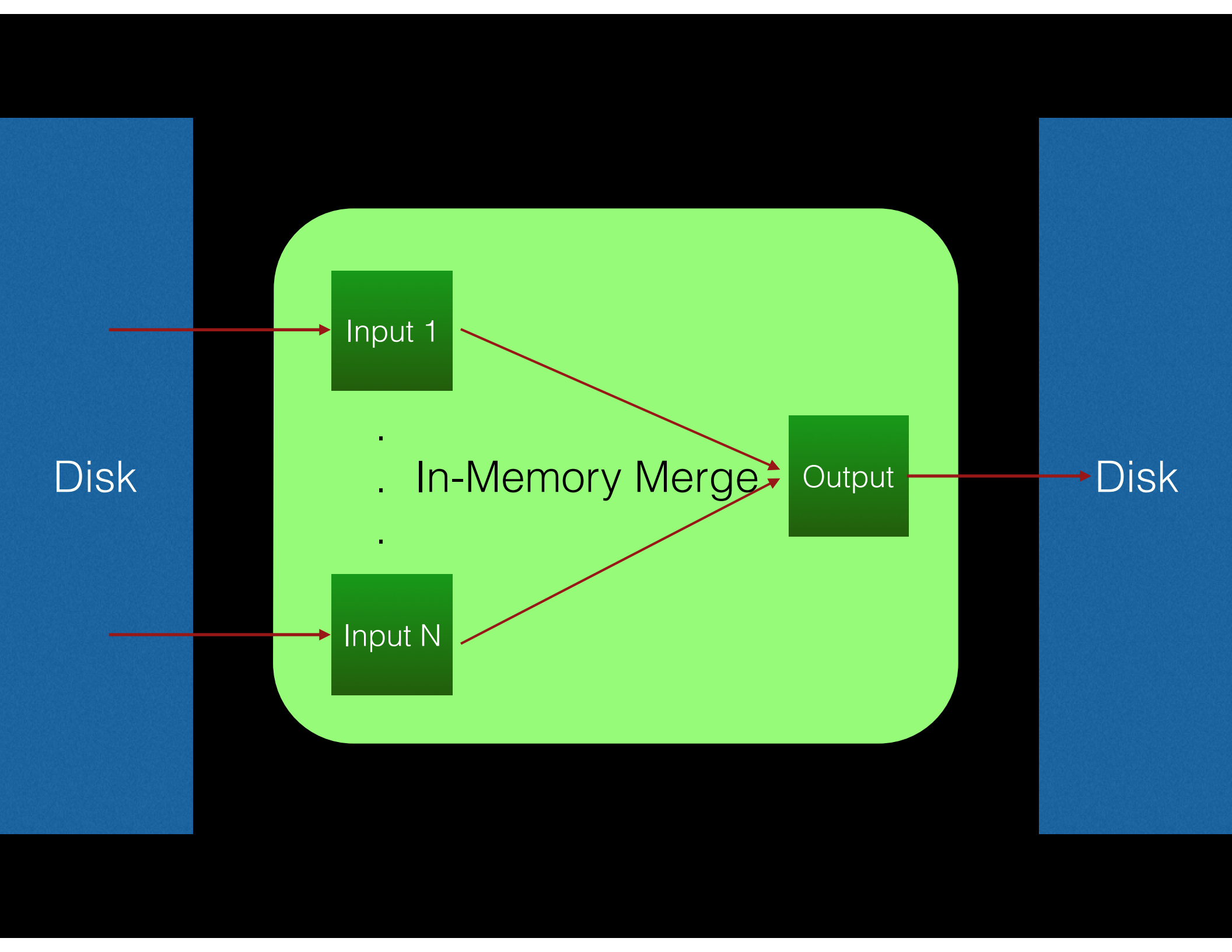

1
2
3
4
5
6
7
8
B1
B2
B1
B2
B1
B2
B1
B2
B1
B2
B1
B2
1
2
3
4
5
6
7
8

2-Way Merge Sort Cost

`N` is the number of pages in the file

Number of passes: `ceil(log_2(N)) + 1`

Number of IOs: `2N (ceil(log_2(N)) + 1)`


Disk
Input 1
.
.
.
In-Memory Merge
Output
Disk
Input N

N-Way Merge Sort Cost

`N` is the number of pages in the file

`B` is the number of pages in memory

Number of passes: `ceil(log_{B-1}(N/B)) + 1`

Number of IOs: `2N (ceil(log_{B-1}(N/B)) + 1)`

External Sorting Exercises

1. List the differences between 2-way external merge sort and general external merge sort:

    - 2-way only utilizes 2 input buffers, general utilizes B-1
    - During pass 0, 2-way only uses 1 page to sort files. notice how applying B=3 to the general equation doesn't yield the correct # of passes. The general external merge sort uses all B pages in its buffer to sort the initial runs in pass 0.

External Sorting Exercises

2. Your system has 640 KB of memory allocated for the buffer for external sorting and you have infinite space for scratch disks. Each page holds 64 KB of data.

    1. How many pages can you buffer hold?

        640KB * (1 page/64KB) = 10 pages

    2. How many pages are in a 4MB file?

        4 MB * (1024 KB/1 MB) * (1 page/64KB) = 64 pages

    3. How many passes would it take to N-way sort a 4 MB file?

        1 + ceil (log_9(7)) = 2 passes

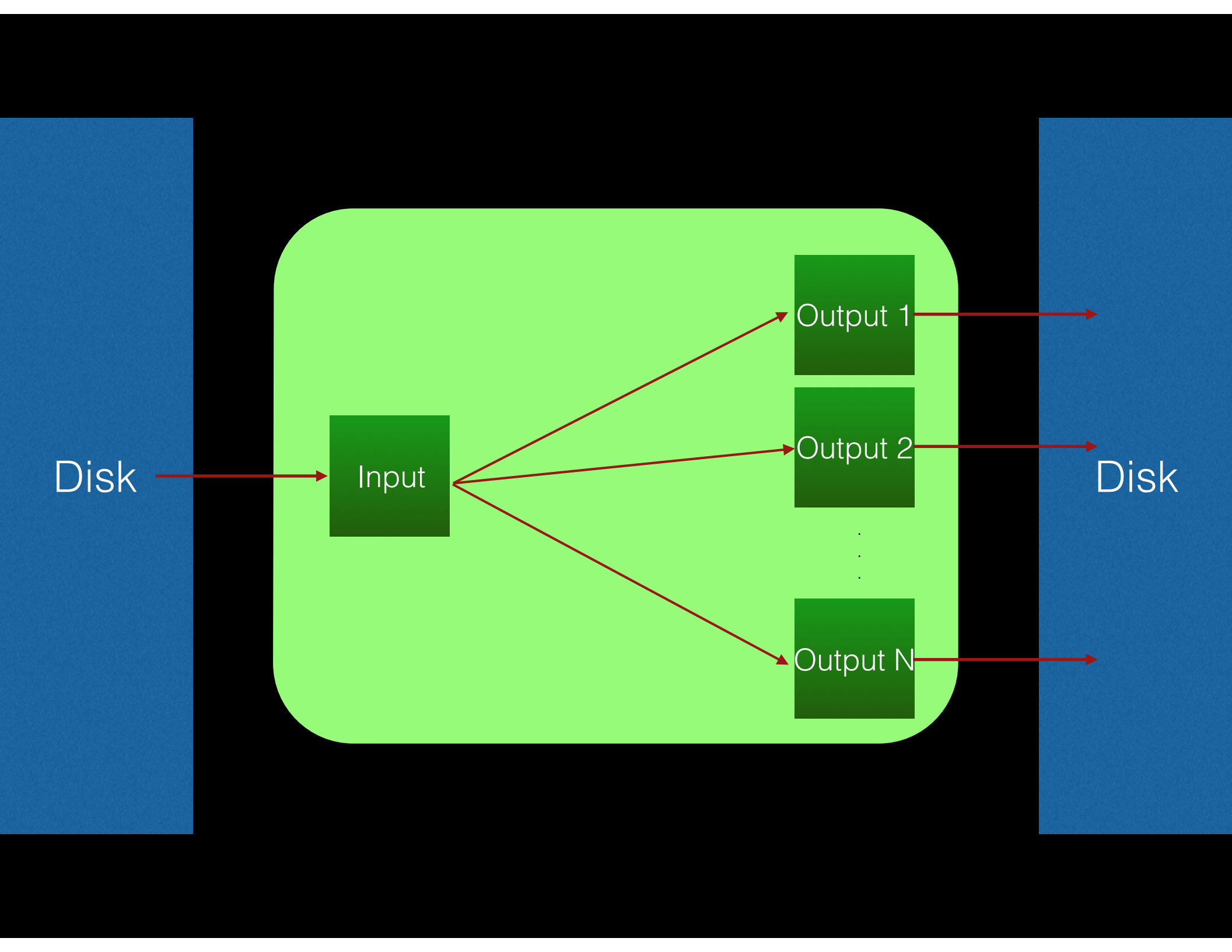
    4. How many IOs are needed for 3?

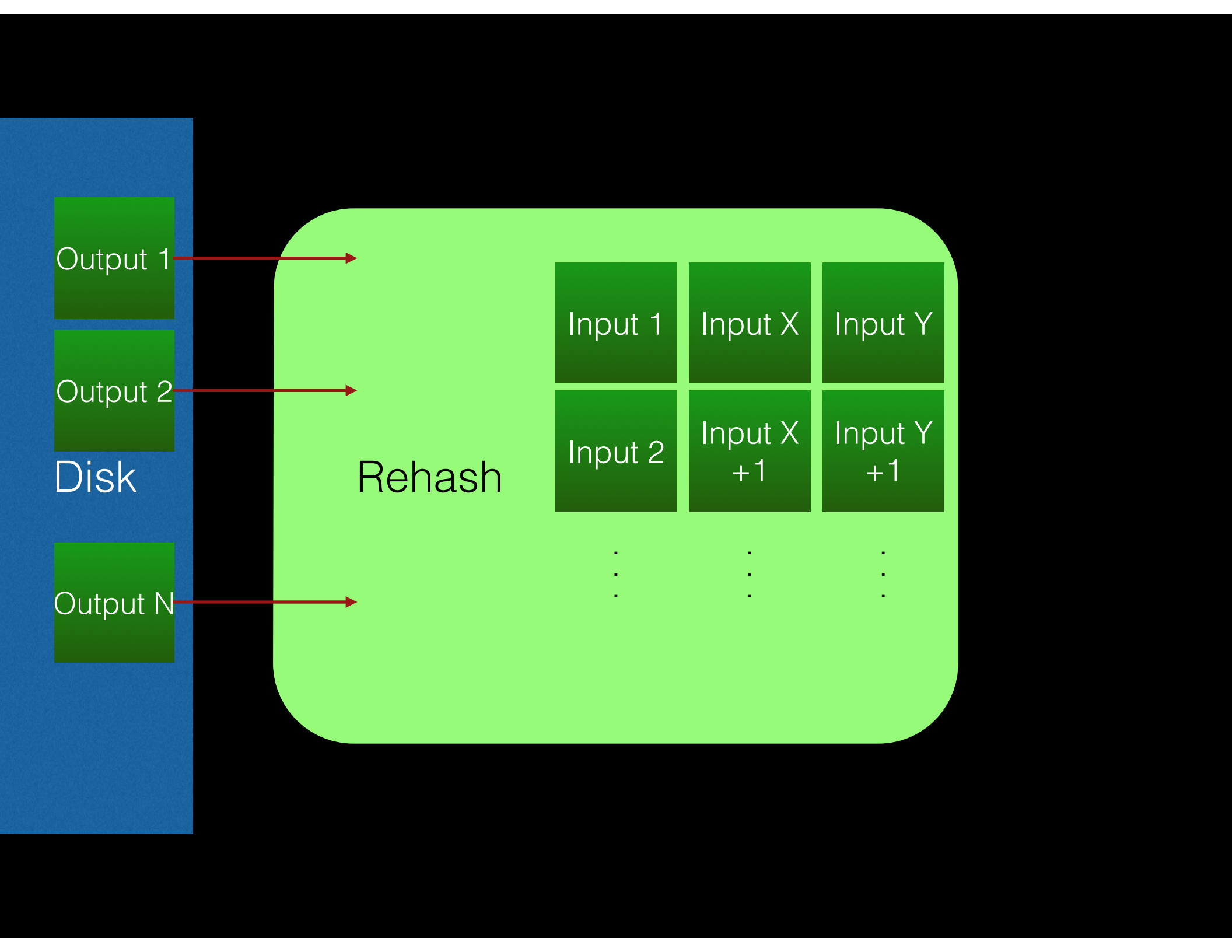
        2 * (num_passes) * (num_pages_in_file) = 256 IOs

    5. What is the maximum file size for 2 passes?

        B * (B-1) = 90 pages; 90 pages * 64 KB/page ~ 5.6MB

External Hashing


Disk
Input
Output 1
Output 2
Output N
Disk


Output 1
Output 2
Disk
Output N
Rehash
Input 1
Input X
Input Y
Input 2
Input X +1
Input Y +1

What happens if a partition has more than B pages?

RECURSIVE PARTITIONING!

External Hashing Exercises

1. Why can we process B * (B - 1) pages of data with external hashing in just two passes (divide and conquer phases)?

   Our main limitation is how big the partitions can be after the partition hashing. Since we need to be able to read in the whole partition into memory, each partition can be at most B pages big.

External Hashing Exercises

2. If you're processing exactly B * (B - 1) pages of data, is it likely that you'll have to perform recursive external hashing? Why?

   You would have to have an absolutely perfect hash function that evenly distributes any record into the B-1 partitions. This is almost impossible in practice. Rather, we should expect that some partitions may be larger than B after partition hashing.

External Hashing Exercises

3. While you recursively perform external hashing, you reuse the same hash functions for partitioning. What's the problem with this?

   The partition that is too big to fit in memory will still be too big to fit in memory if we maintain the same partition hashing strategy.

Single-Table SQL

```
SELECT [DISTINCT] <column names>
FROM <table>
[WHERE <predicate>]
[GROUP BY <columns> [HAVING <column predicate>]]
[ORDER BY <column list>];
```

SQL Exercises

```
Songs  (song_id,   song_name,
album_num,  weeks_in_top_40)

Artists(artist_id, artist_name,
first_year_active)

Album  (album_id,  album_name,
artist_num, year_released, genre)
```

SQL Exercises

```
Songs  (song_id,   song_name,   album_num,  weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Album  (album_id,  album_name,  artist_num, year_released, genre)
```

Find the name of every song that spent more than 10 weeks in the top 40.

```
SELECT song_name
FROM Songs
WHERE weeks_in_top_40 > 10;
```

SQL Exercises

```
Songs  (song_id,   song_name,   album_num,  weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Album  (album_id,  album_name,  artist_num, year_released, genre)
```

Find the name and first year active of every artist whose name starts with the letter ‘A’.

```
SELECT artist_name, first_year_active
FROM Artists
WHERE artistname LIKE ‘A%’;
```

SQL Exercises

```
Songs  (song_id,   song_name,   album_num,  weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Album  (album_id,  album_name,  artist_num, year_released, genre)
```

Find the number of “Rap” albums released each year.

```
SELECT year_released, COUNT(*)
FROM Album WHERE genre = ‘Rap’
GROUP BY year_released;
```

SQL Exercises

```
Songs  (song_id,   song_name,   album_num,  weeks_in_top_40)

Artists(artist_id, artist_name, first_year_active)

Album  (album_id,  album_name,  artist_num, year_released, genre)
```

Find the number of albums released in each genre; don't include genres that have a count of less than 10.

```
SELECT genre, COUNT(*)
FROM Album
GROUP BY genre
HAVING COUNT(*) >= 10;
```