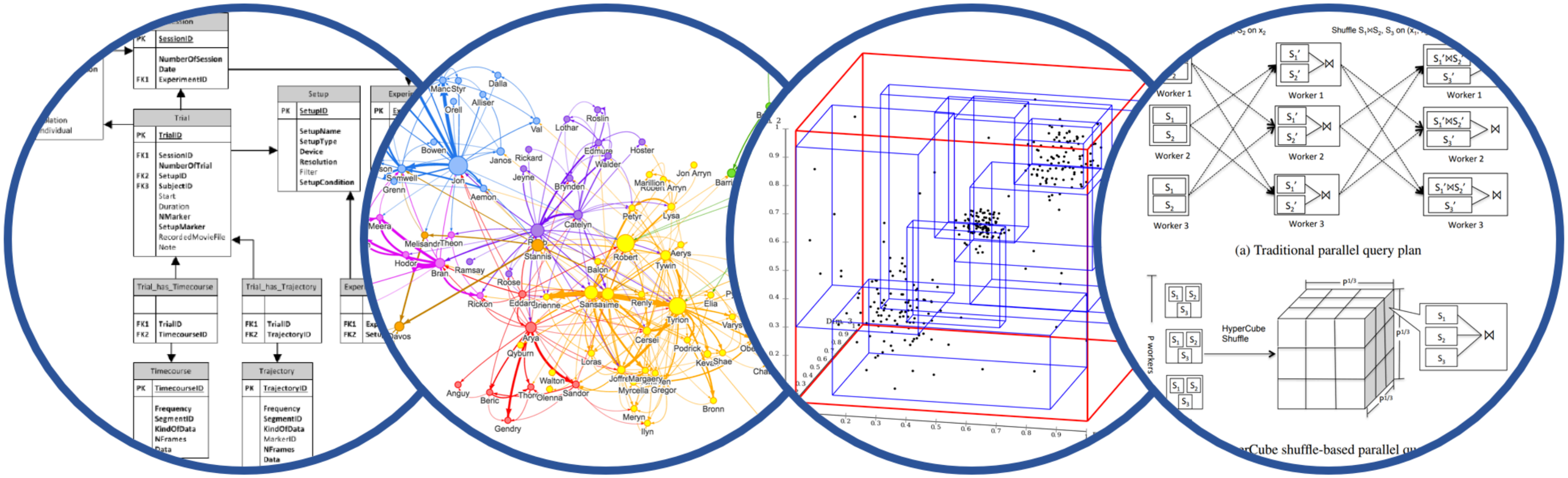


Database System Internals

Operator Algorithms (part 2)

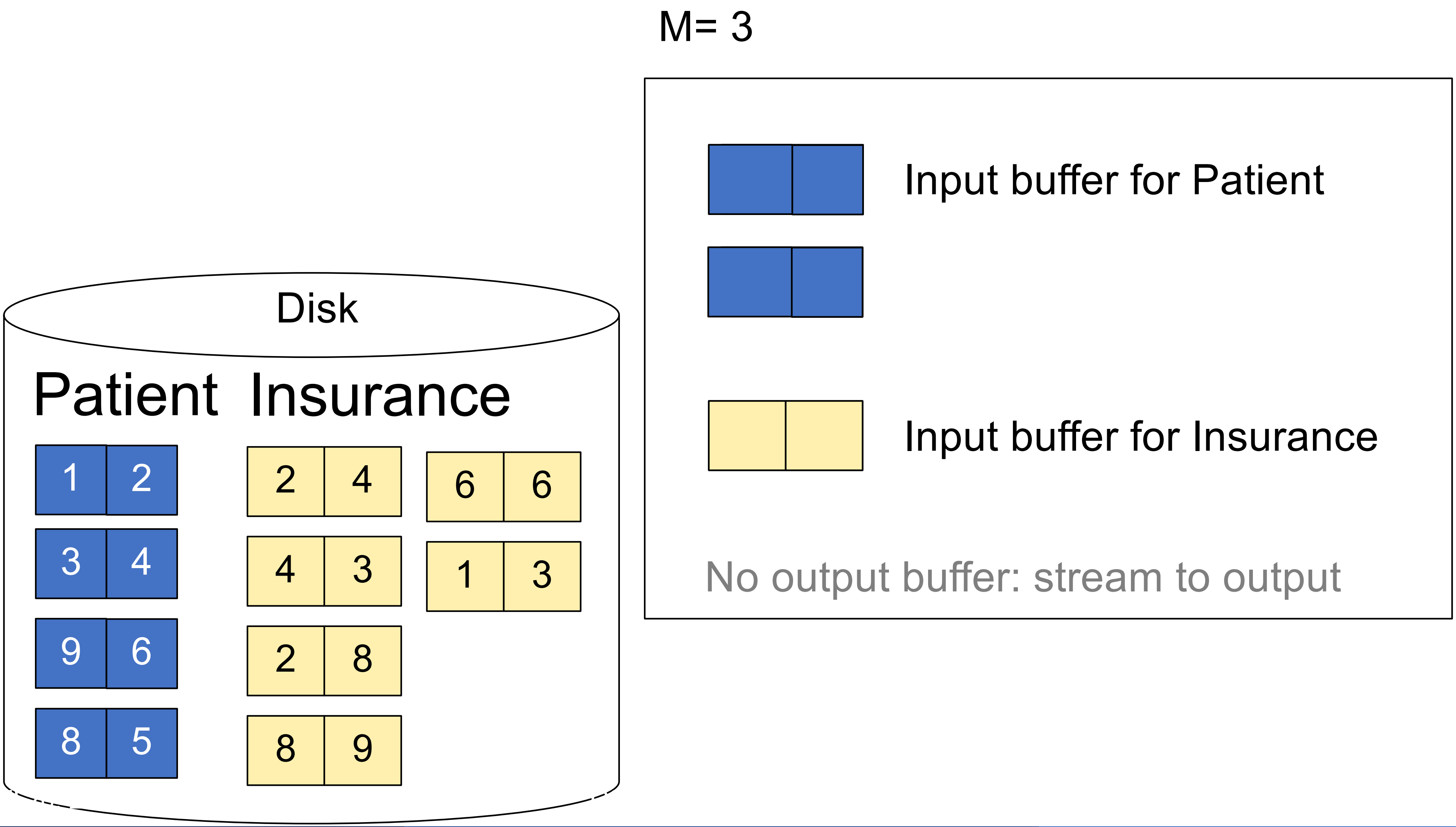
Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Block-Memory Refinement

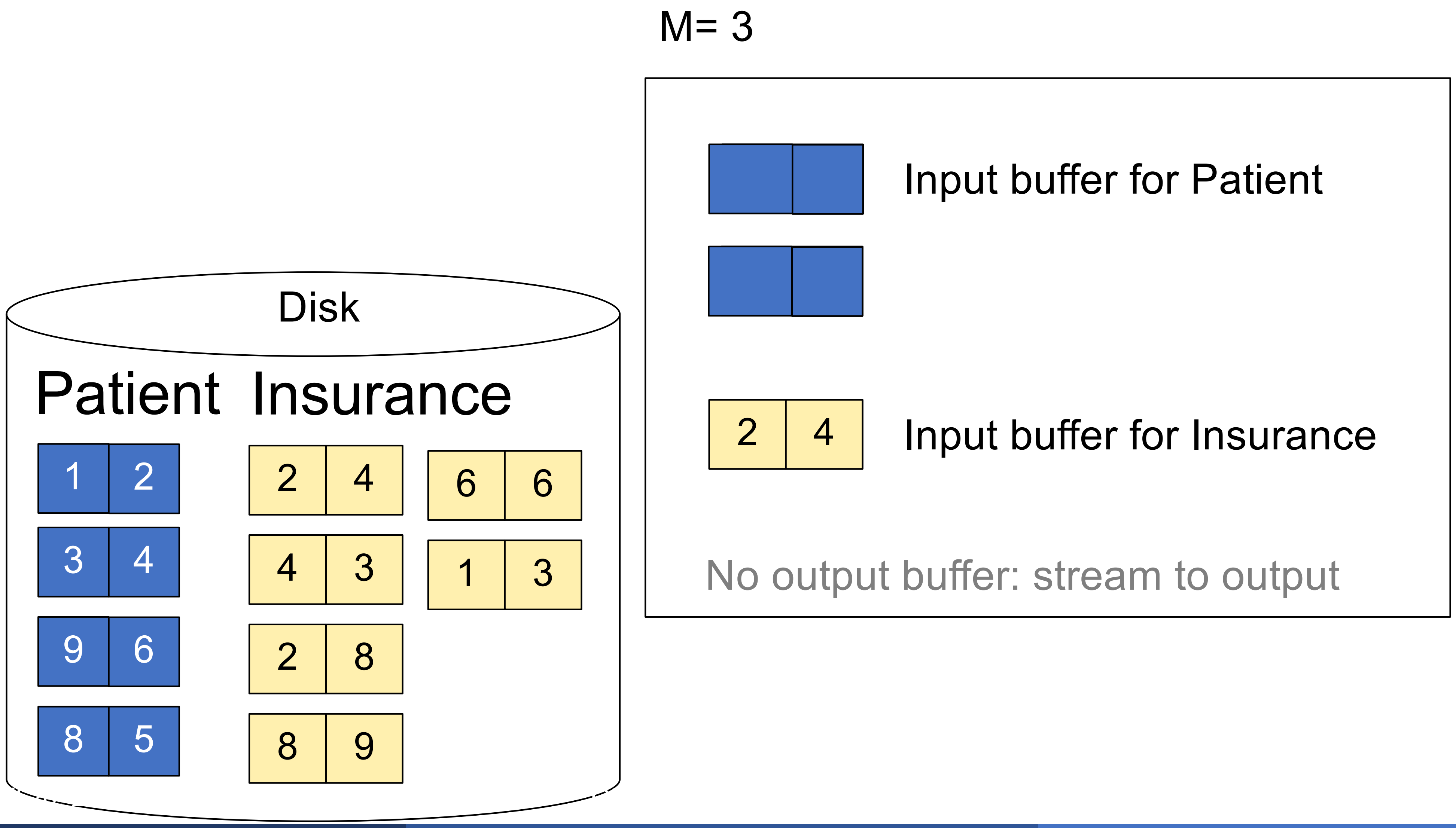
for each group of M-1 pages r in R do
  for each page of tuples s in S do
    for all pairs of tuples t_1 in r, t_2 in s
      if t_1 and t_2 join then output (t_1,t_2)

What is the Cost?

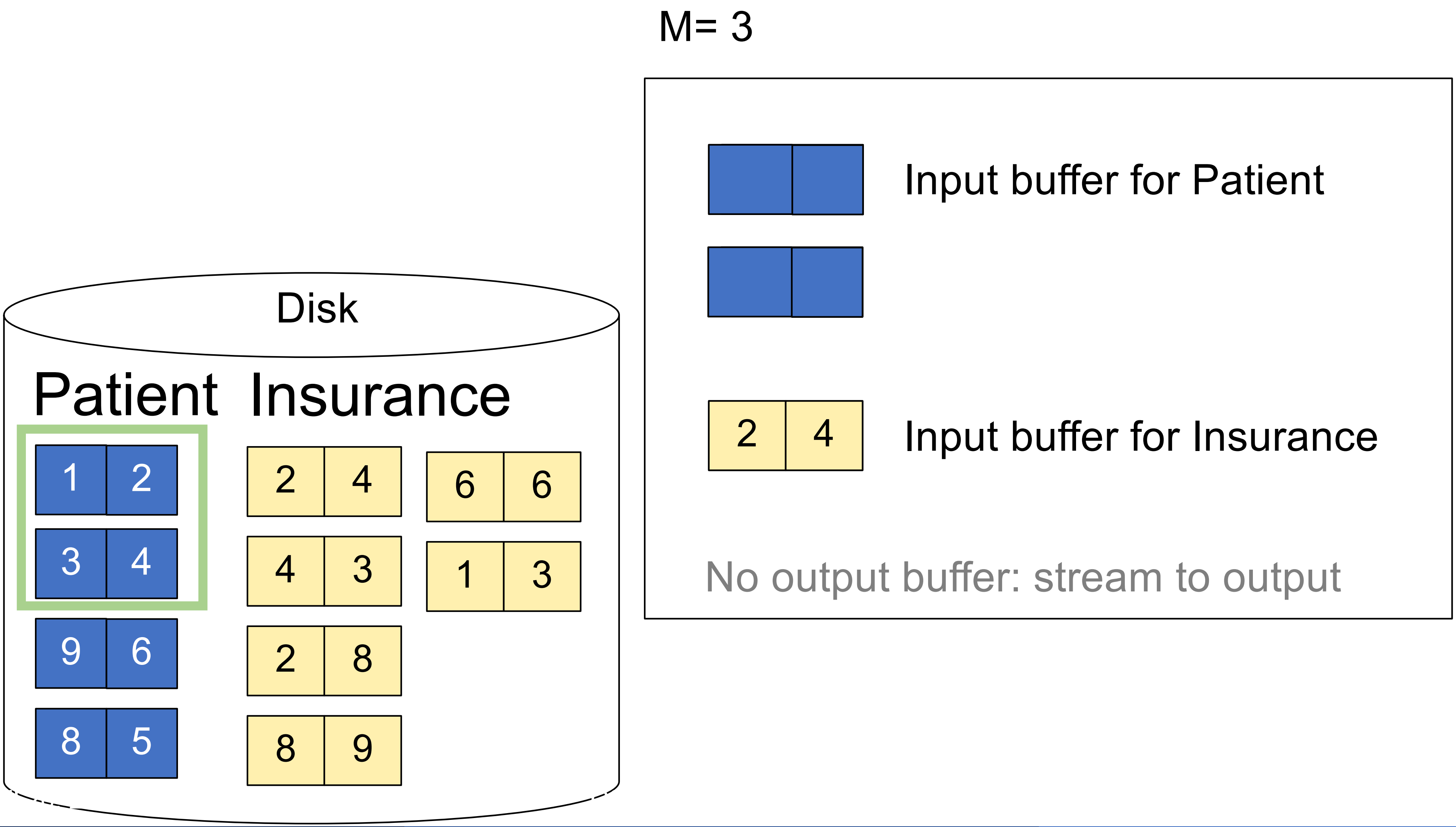
Block Memory Refinement



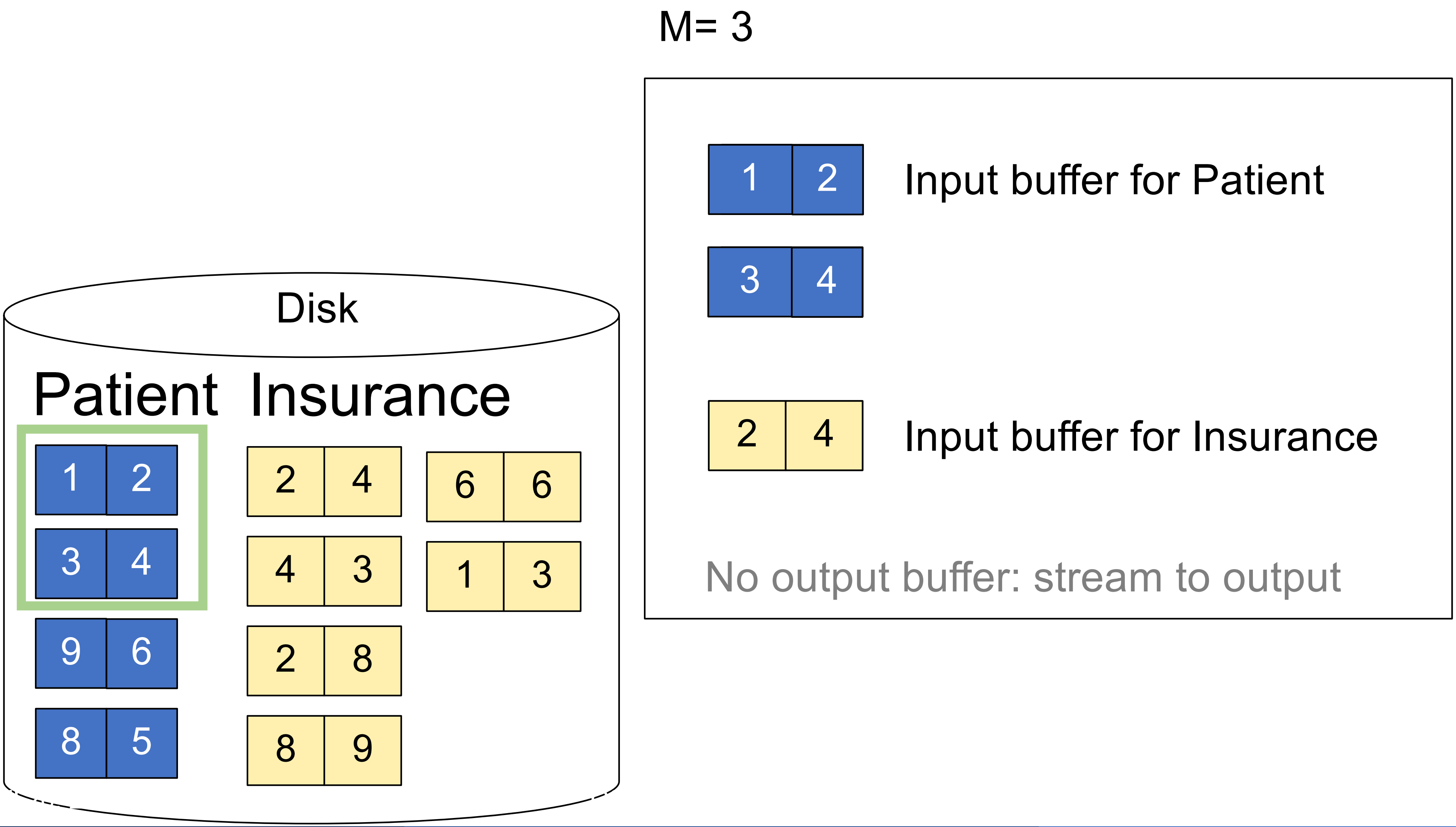
Block Memory Refinement



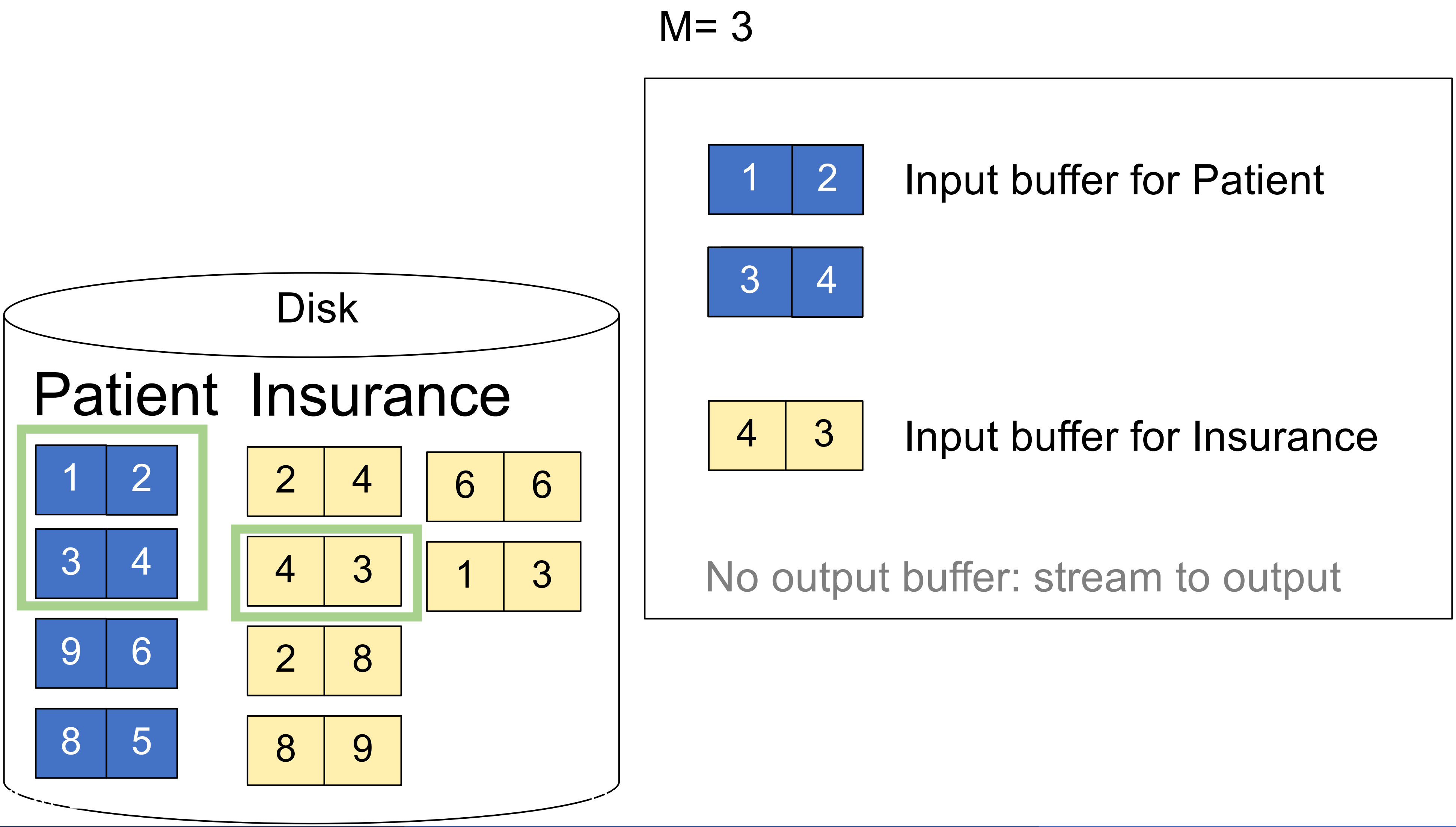
Block Memory Refinement



Block Memory Refinement

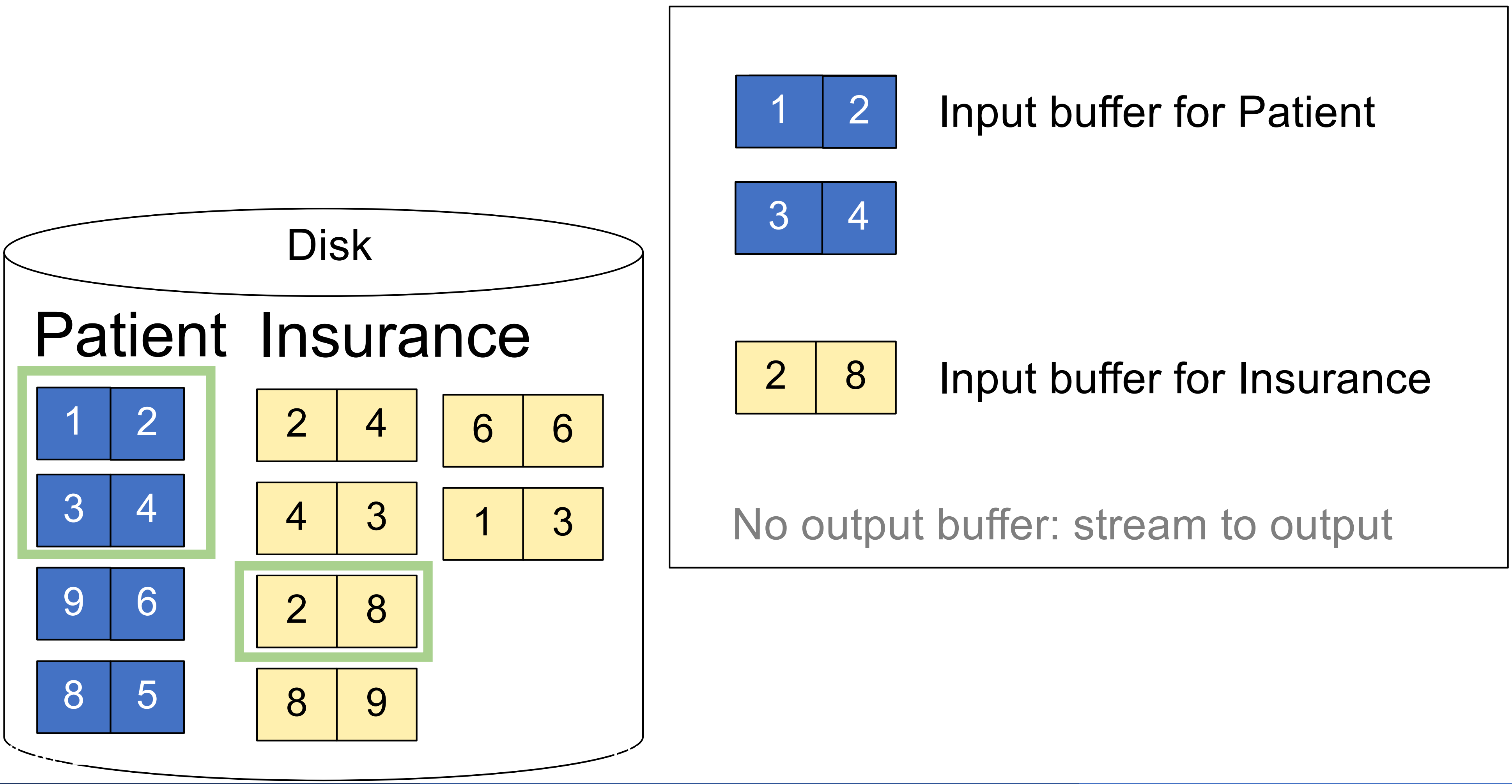


Block Memory Refinement

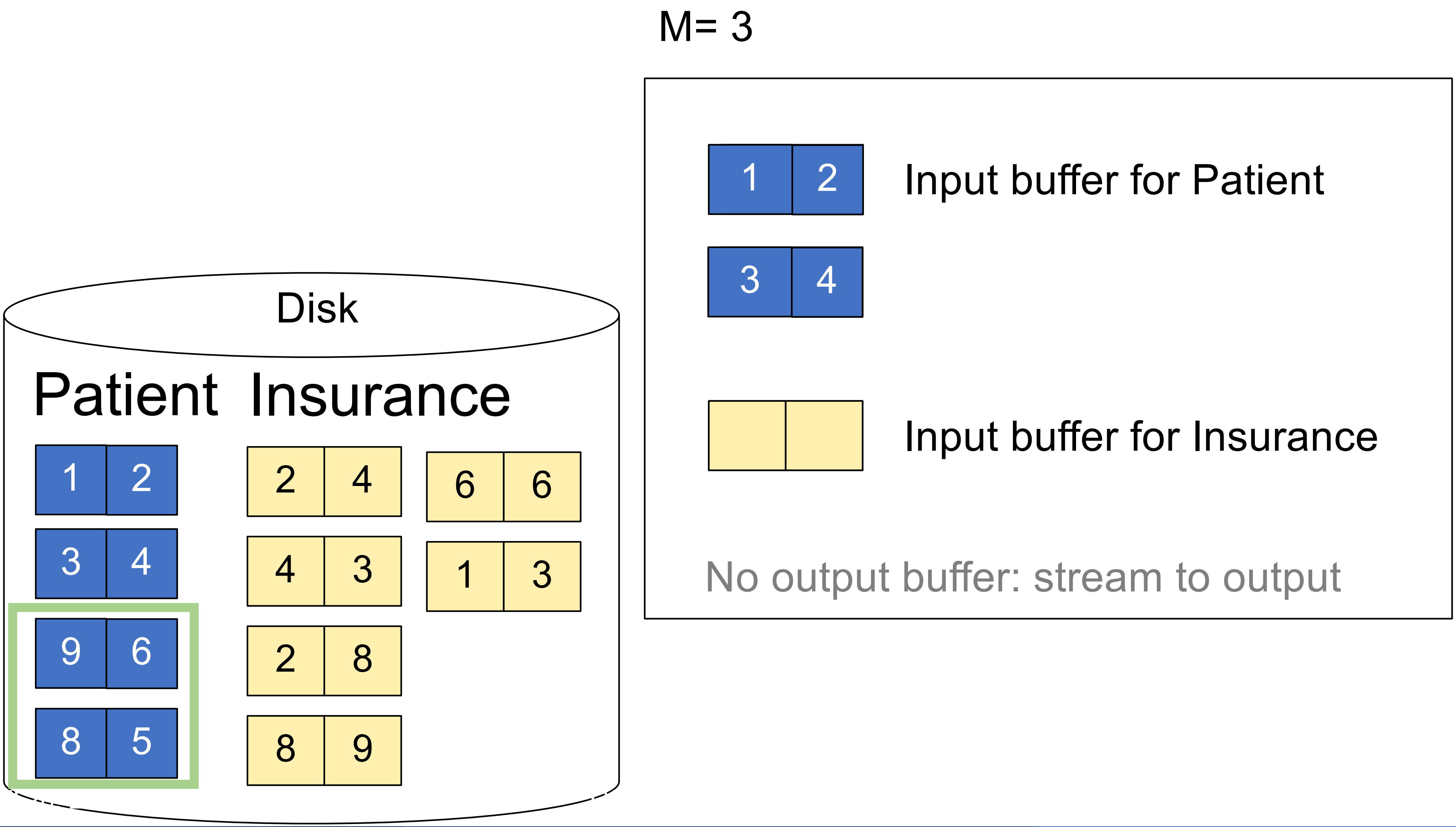


Block Memory Refinement

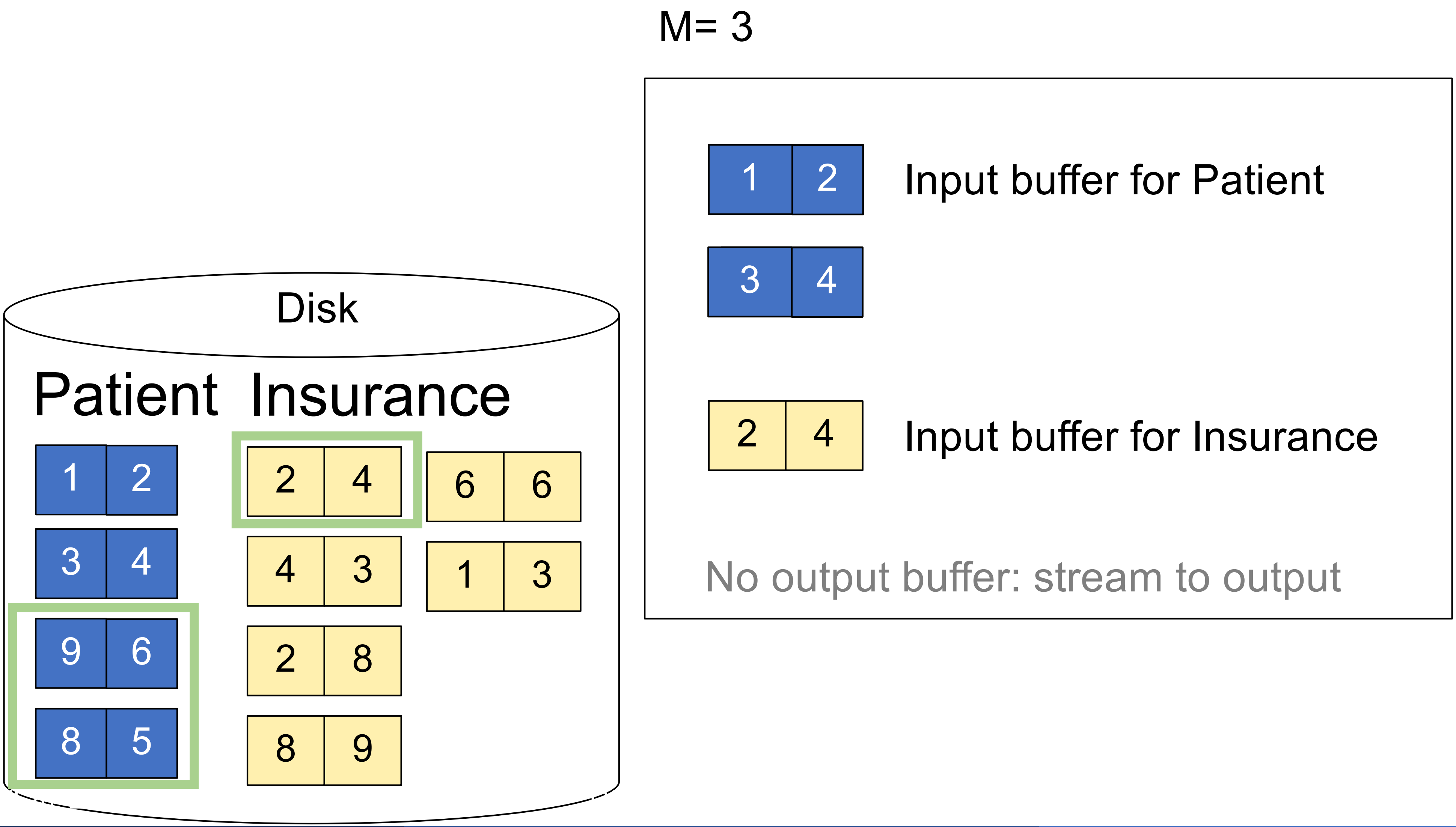
M= 3



Block Memory Refinement



Block Memory Refinement



Block Memory Refinement

```
for each group of M-1 pages r in R do
   for each page of tuples s in S do
      for all pairs of tuples t1 in r, t2 in s
         if t1 and t2 join then output (t1,t2)
```

What is the Cost?

Block Memory Refinement

```
for each group of M-1 pages r in R do
   for each page of tuples s in S do
      for all pairs of tuples t1 in r, t2 in s
         if t1 and t2 join then output (t1,t2)
```

- Cost: $B(R) + B(R)B(S)/(M-1)$

What is the Cost?

Sort-Merge Join

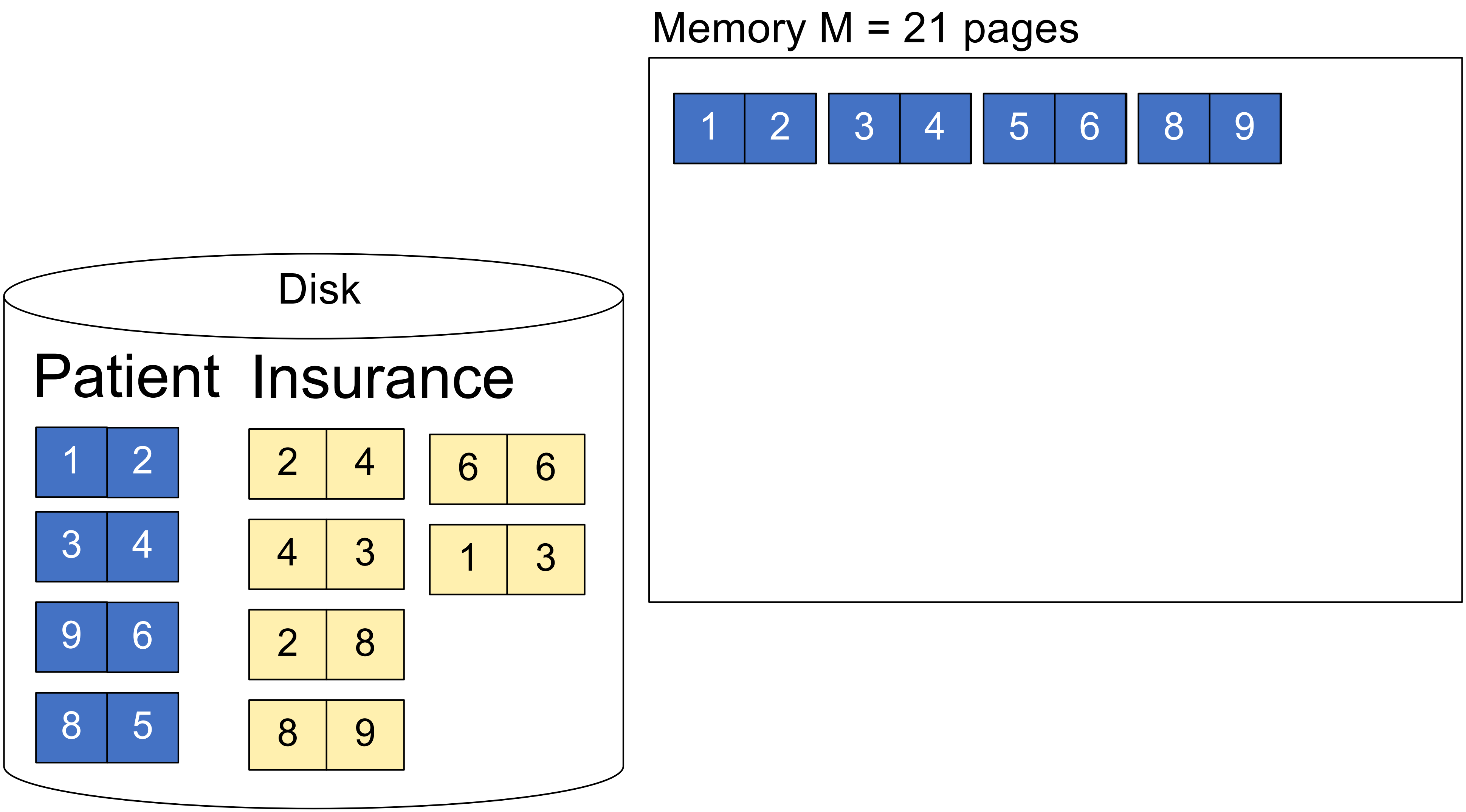
Sort-merge join: $R \bowtie S$

- Scan R and sort in main memory
- Scan S and sort in main memory
- Merge R and S

- Cost: $B(R) + B(S)$
- One pass algorithm when $B(S) + B(R) \le M$
- Typically, this is NOT a one pass algorithm,
  - We'll see the multi-pass version next lecture

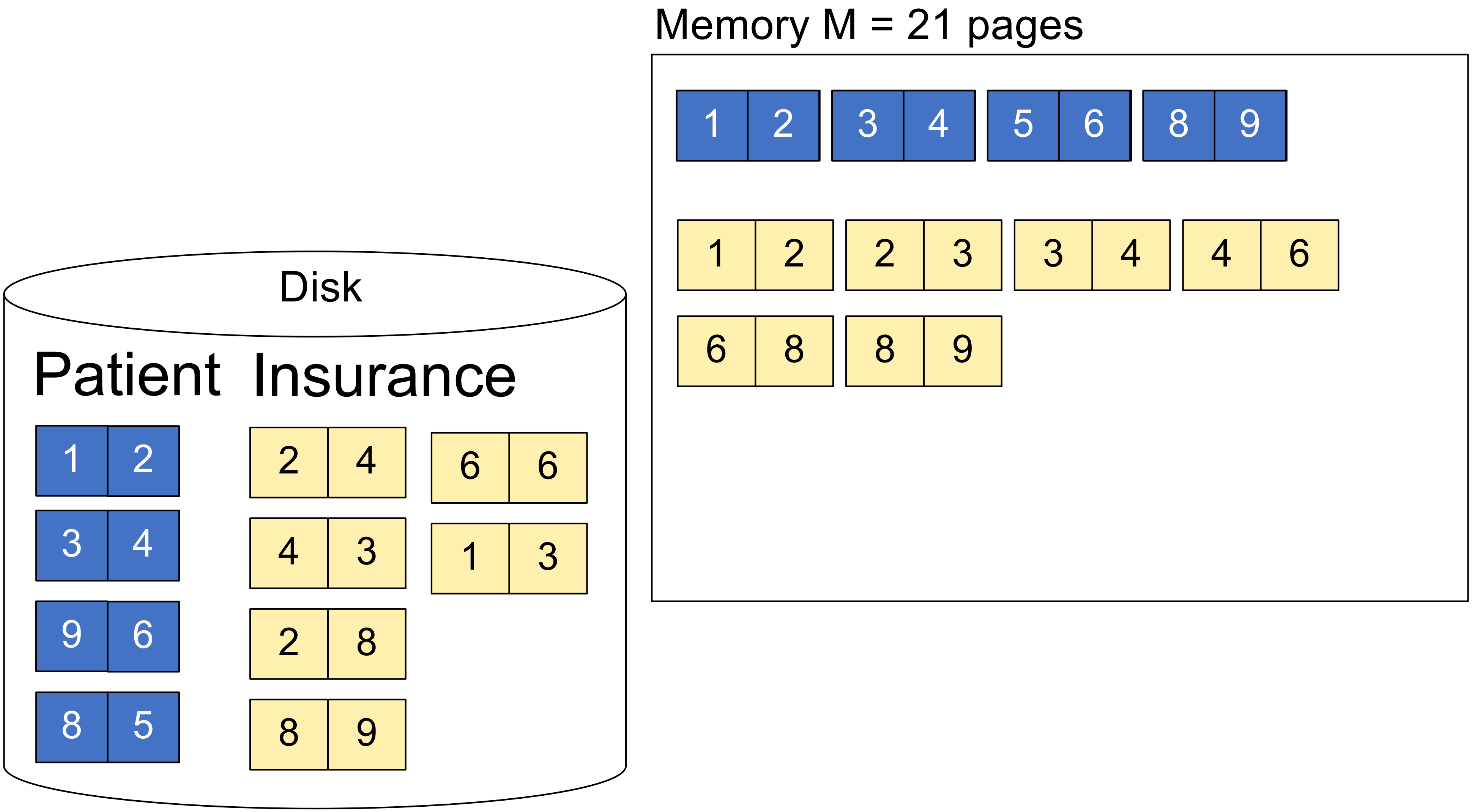
Sort-Merge Join Example

Step 1: Scan Patient and sort in memory



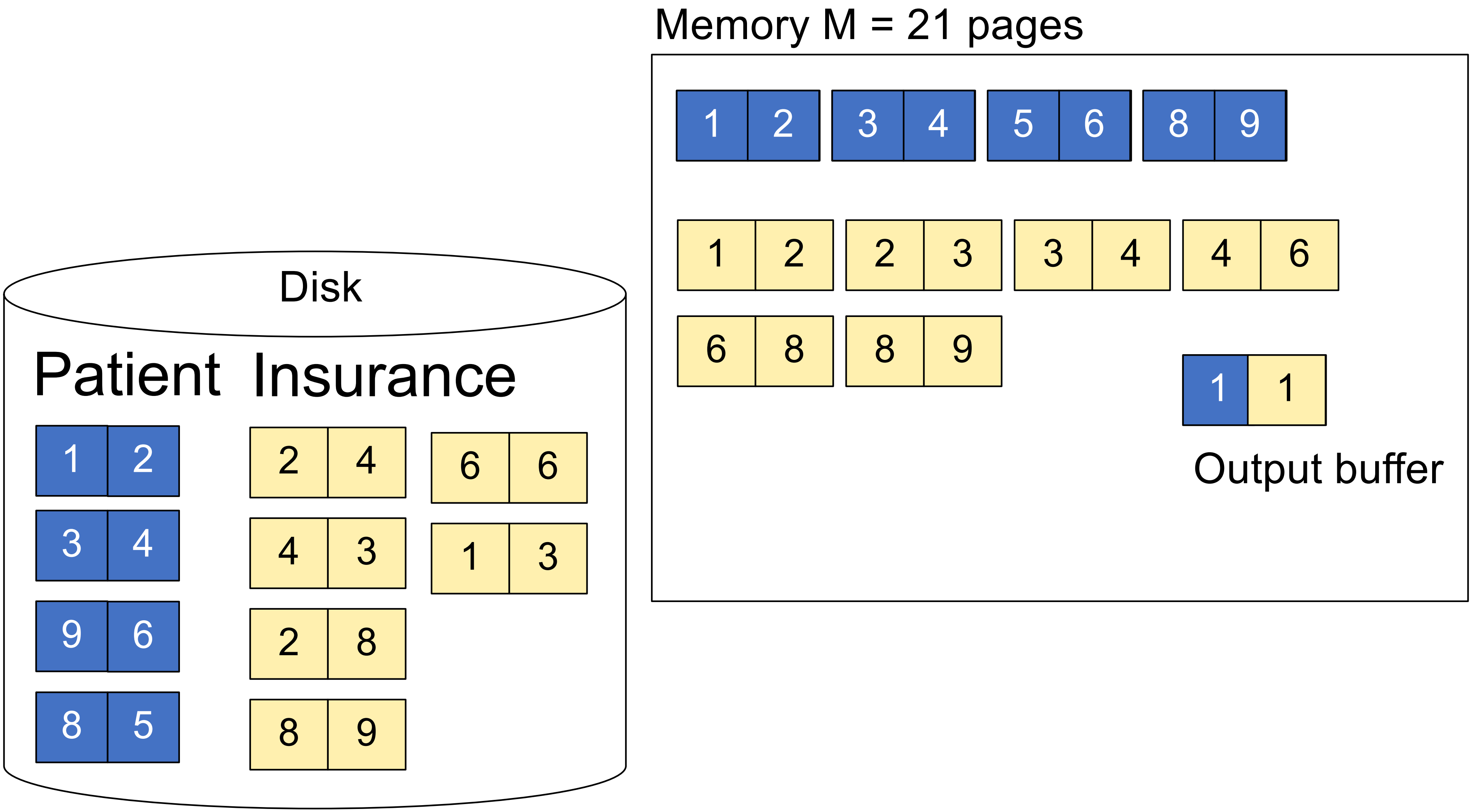
Sort-Merge Join Example

Step 2: Scan Insurance and sort in memory



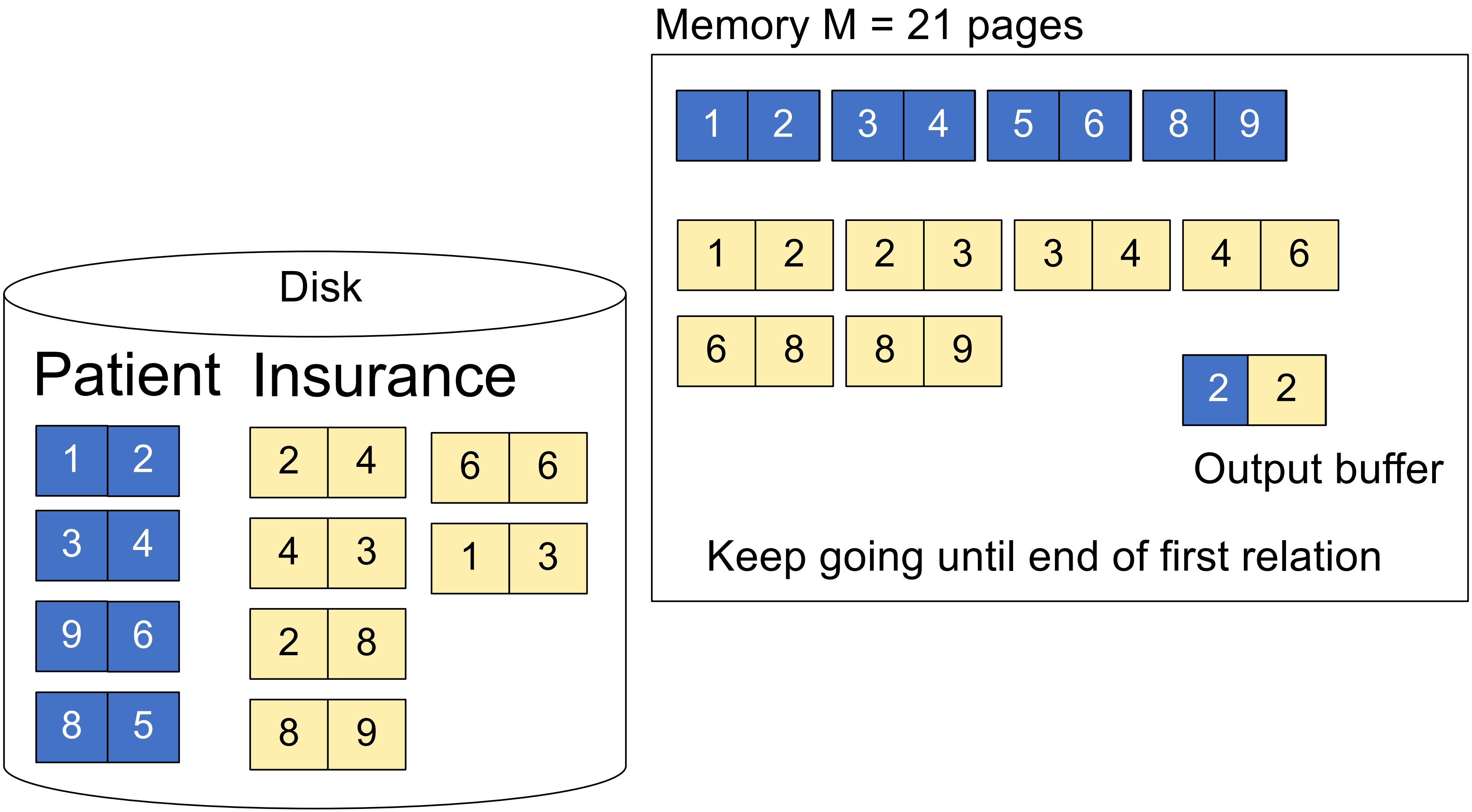
Sort-Merge Join Example

Step 3: Merge Patient and Insurance



Sort-Merge Join Example

Step 3: Merge Patient and Insurance



Outline

- **Join operator algorithms**
  - One-pass algorithms (Sec. 15.2 and 15.3)
  - Index-based algorithms (Sec 15.6)
  - Two-pass algorithms (Sec 15.4 and 15.5)

Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- B(R)= size of R in blocks
- T(R) = number of tuples in R
- V(R, a) = # of distinct values of attribute a

Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- B(R)= size of R in blocks
- T(R) = number of tuples in R
- V(R, a) = # of distinct values of attribute a

What is the cost in each case?

- Clustered index on a:
- Unclustered index on a:

Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- B(R)= size of R in blocks
- T(R) = number of tuples in R
- V(R, a) = # of distinct values of attribute a

What is the cost in each case?

- Clustered index on a: B(R)/V(R,a)
- Unclustered index on a:

Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- B(R)= size of R in blocks
- T(R) = number of tuples in R
- V(R, a) = # of distinct values of attribute a

What is the cost in each case?

- Clustered index on a: B(R)/V(R,a)
- Unclustered index on a: T(R)/V(R,a)

Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- B(R)= size of R in blocks
- T(R) = number of tuples in R
- V(R, a) = # of distinct values of attribute a

What is the cost in each case?

- Clustered index on a: B(R)/V(R,a)
- Unclustered index on a: T(R)/V(R,a)

Note: we ignore I/O cost for index pages

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R)$ = ?

- Table scan:
- Index based selection:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R)$ = ?

- Table scan: B(R) = 2,000 I/Os
- Index based selection:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered:
  - If index is unclustered:

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered: B(R)/V(R,a) = 100 I/Os
  - If index is unclustered:

Index Based Selection

- Example:

  $B(R) = 2000$
  $T(R) = 100{,}000$
  $V(R, a) = 20$

  cost of $\sigma_{a=v}(R) = ?$

- Table scan: $B(R) = 2{,}000$ I/Os
- Index based selection:
  - If index is clustered: $B(R)/V(R,a) = 100$ I/Os
  - If index is unclustered: $T(R)/V(R,a) = 5{,}000$ I/Os

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R)$ = ?

- Table scan: B(R) = 2,000 I/Os !
- Index based selection:
  - If index is clustered: B(R)/V(R,a) = 100 I/Os
  - If index is unclustered: T(R)/V(R,a) = 5,000 I/Os !

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os !
- Index based selection:
  - If index is clustered: B(R)/V(R,a) = 100 I/Os
  - If index is unclustered: T(R)/V(R,a) = 5,000 I/Os !

Lesson: Don't build unclustered indexes when V(R,a) is small !

Index Based Selection

- Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R) = ?$

- Table scan: B(R) = 2,000 I/Os
- Index based selection:
  - If index is clustered: B(R)/V(R,a) = 100 I/Os
  - If index is unclustered: T(R)/V(R,a) = 5,000 I/Os

Lesson: Don't build unclustered indexes when V(R,a) is small !

Index Nested Loop Join

R ⋈ S

- Assume S has an index on the join attribute
- Iterate over R, for each tuple fetch corresponding tuple(s) from S

- Previous nested loop join: cost
  - B(R) + T(R)*B(S)
- Index Nested Loop Join Cost:
  - If index on S is clustered: B(R) + T(R)B(S)/V(S,a)
  - If index on S is unclustered: B(R) + T(R)T(S)/V(S,a)

Outline

- **Join operator algorithms**
  - One-pass algorithms (Sec. 15.2 and 15.3)
  - Index-based algorithms (Sec 15.6)
  - Two-pass algorithms (Sec 15.4 and 15.5)

Two-Pass Algorithms

- Fastest algorithm seen so far is one-pass hash join
  What if data does not fit in memory?
- Need to process it in multiple passes
- Two key techniques
  - Sorting
  - Hashing

Basic Terminology

- A run in a sequence is an increasing subsequence

- What are the runs?

  2, 4, 99, 103, 88, 77, 3, 79, 100, 2, 50

Basic Terminology

- A run in a sequence is an increasing subsequence

- What are the runs?

2, 4, 99, 103, | 88, | 77, 3 | , 79, 100, 2, | 50

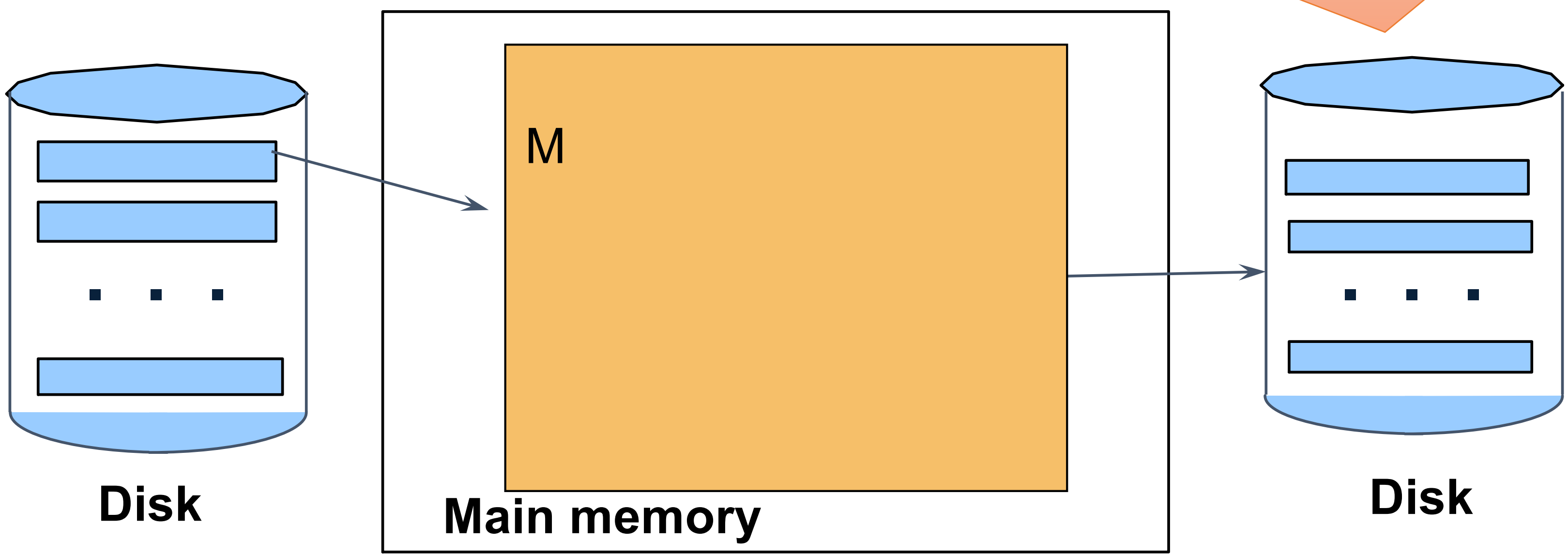
External Merge-Sort: Step 1

Phase one: load M blocks in memory, sort, send to disk, repeat

External Merge-Sort: Step 1

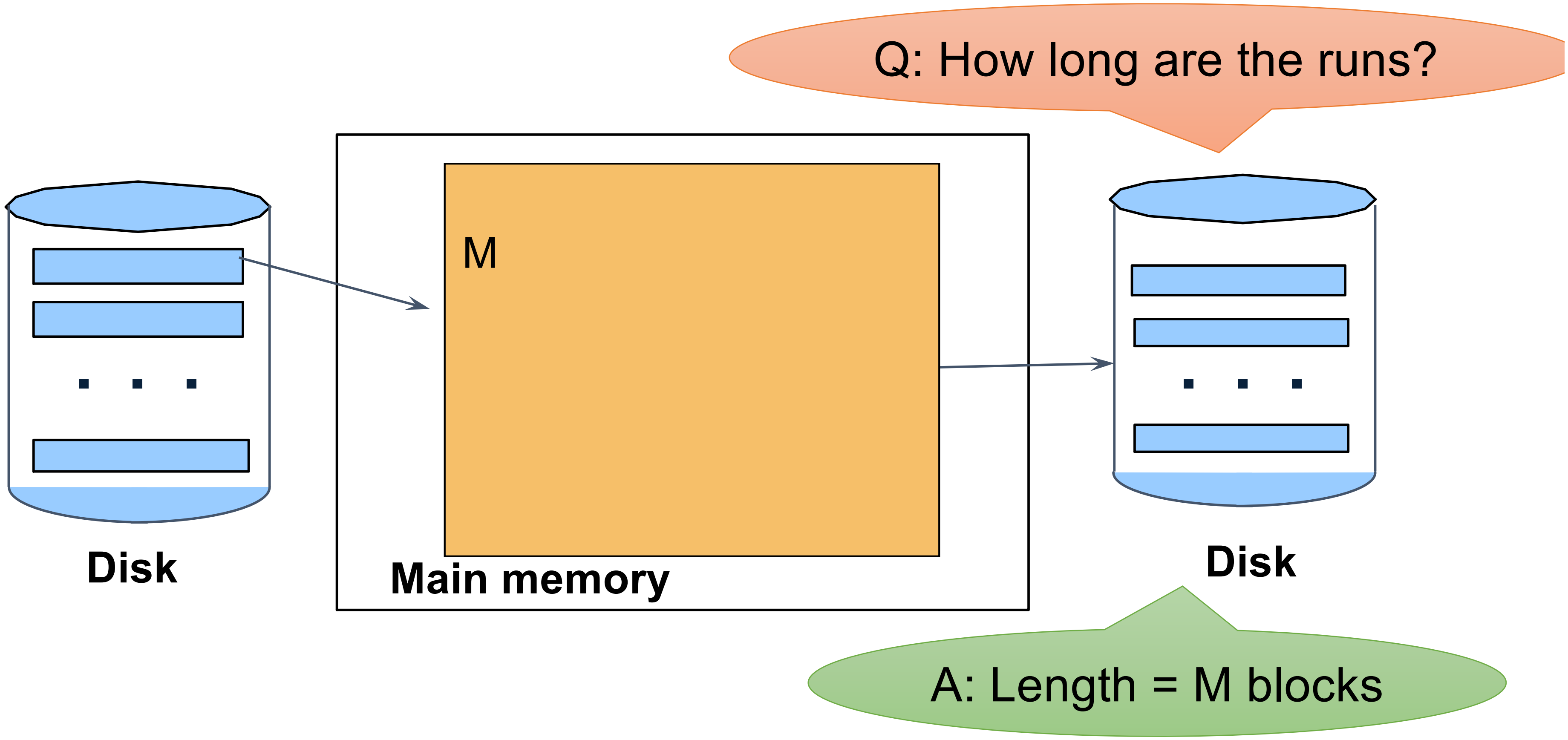
Phase one: load M blocks in memory, sort, send to disk, repeat

Q: How long are the runs?



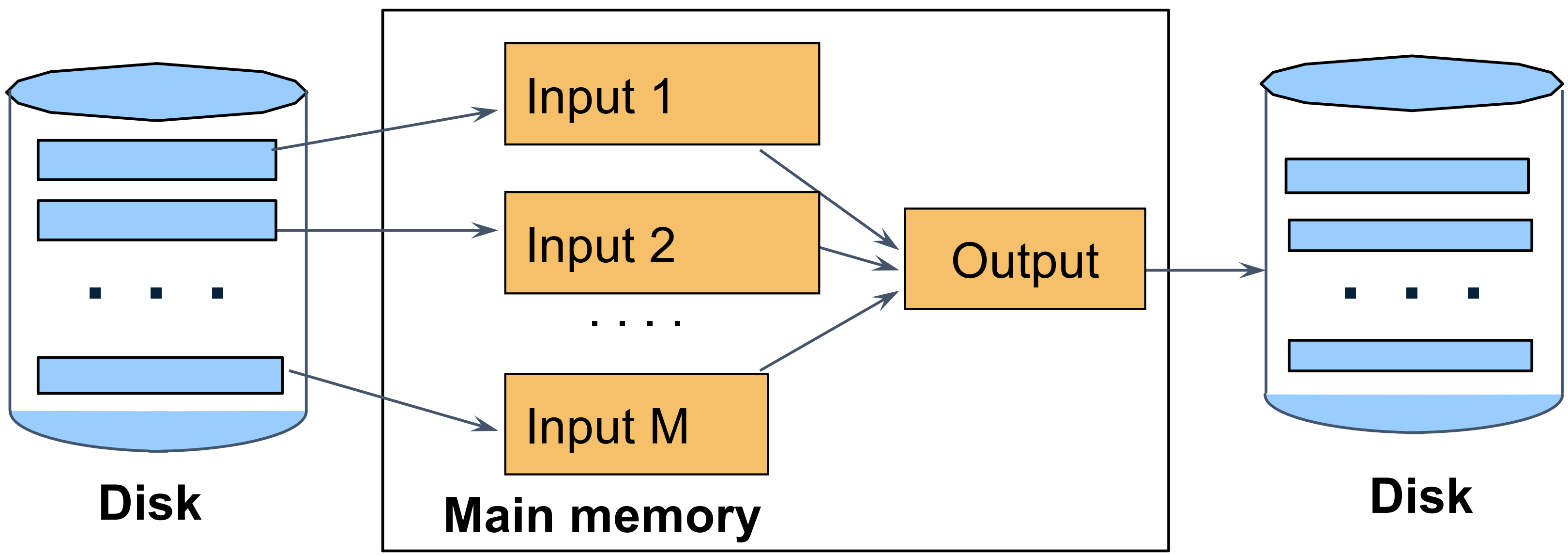
External Merge-Sort: Step 1

Phase one: load M blocks in memory, sort, send to disk, repeat



Phase two: merge M runs into a bigger run

- Merge M – 1 runs into a new run
- Result: runs of length M (M – 1) ≈ M^2



Example

- Merging three runs to produce a longer run:

0, 14, 33, 88, 92, 192, 322
2, 4, 7, 43, 78, 103, 523
1, 6, 9, 12, 33, 52, 88, 320

Output:
0

Example

- Merging three runs to produce a longer run:

0, **14, 33, 88, 92, 192, 322**
2, 4, 7, 43, 78, 103, 523
1, 6, 9, 12, 33, 52, 88, 320

Output:
0, ?

Example

- Merging three runs to produce a longer run:

0, **14, 33, 88, 92, 192, 322**
2, 4, 7, 43, 78, 103, 523
1, **6, 9, 12, 33, 52, 88, 320**

Output:
0, 1, ?

Example

- Merging three runs to produce a longer run:

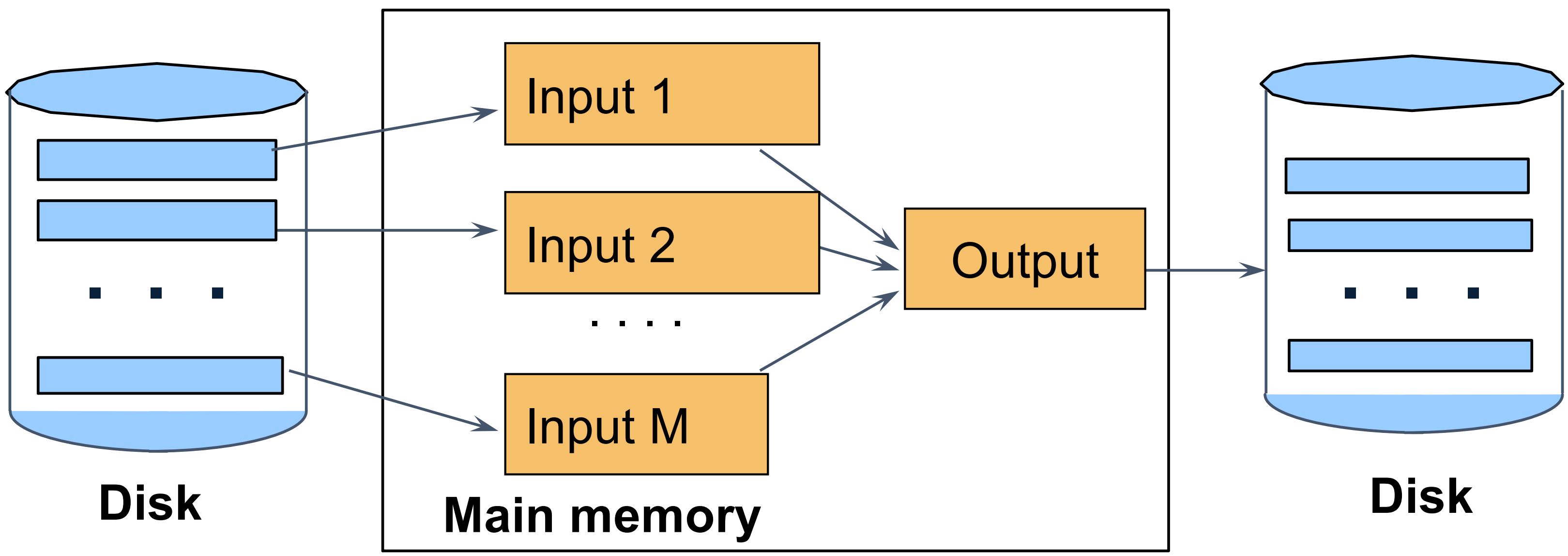
0, **14, 33, 88, 92, 192, 322**
2, 4, 7, **43, 78, 103, 523**
1, 6, **9, 12, 33, 52, 88, 320**

Output:
0, 1, 2, 4, 6, 7, ?

External Merge-Sort: Step 2

Phase two: merge M runs into a bigger run

- Merge M – 1 runs into a new run
- Result: runs of length M (M – 1) ≈ M^2



If approx. B <= M^2 then we are done

Cost of External Merge Sort

- Assumption: $B(R) <= M^2$
- Read+write+read = $3B(R)$

Discussion

- What does $B(R) <= M^2$ mean?
- How large can R be?

Discussion

- What does $B(R) <= M^2$ mean?
- How large can R be?

- Example:
  - Page size = 32KB
  - Memory size 32GB: $M = 10^6$-pages

Discussion

- What does $B(R) <= M^2$ mean?
- How large can R be?

- Example:
  - Page size = 32KB
  - Memory size 32GB: $M = 10^6$ pages

- R can be as large as 10^{12} pages
  - 32×10^{15} Bytes = 32 PB

Merge-Join

Join $R \bowtie S$

- How?....

Merge-Join

Join R ⋈ S

- Step 1a: generate initial runs for R
- Step 1b: generate initial runs for S
- Step 2: merge and join
  - Either merge first and then join
  - Or merge & join at the same time

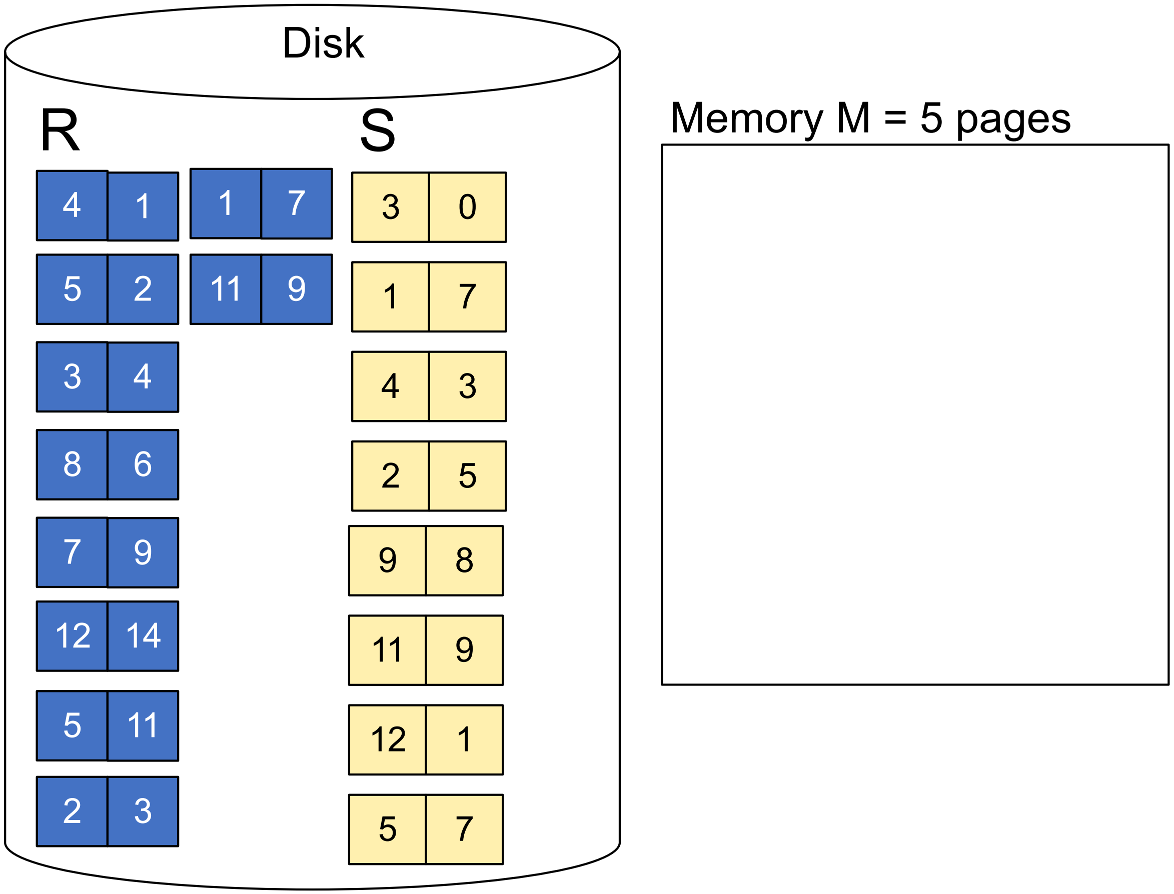
Merge-Join Example

Setup: Want to join R and S

Relation R has 10 pages with 2 tuples per page

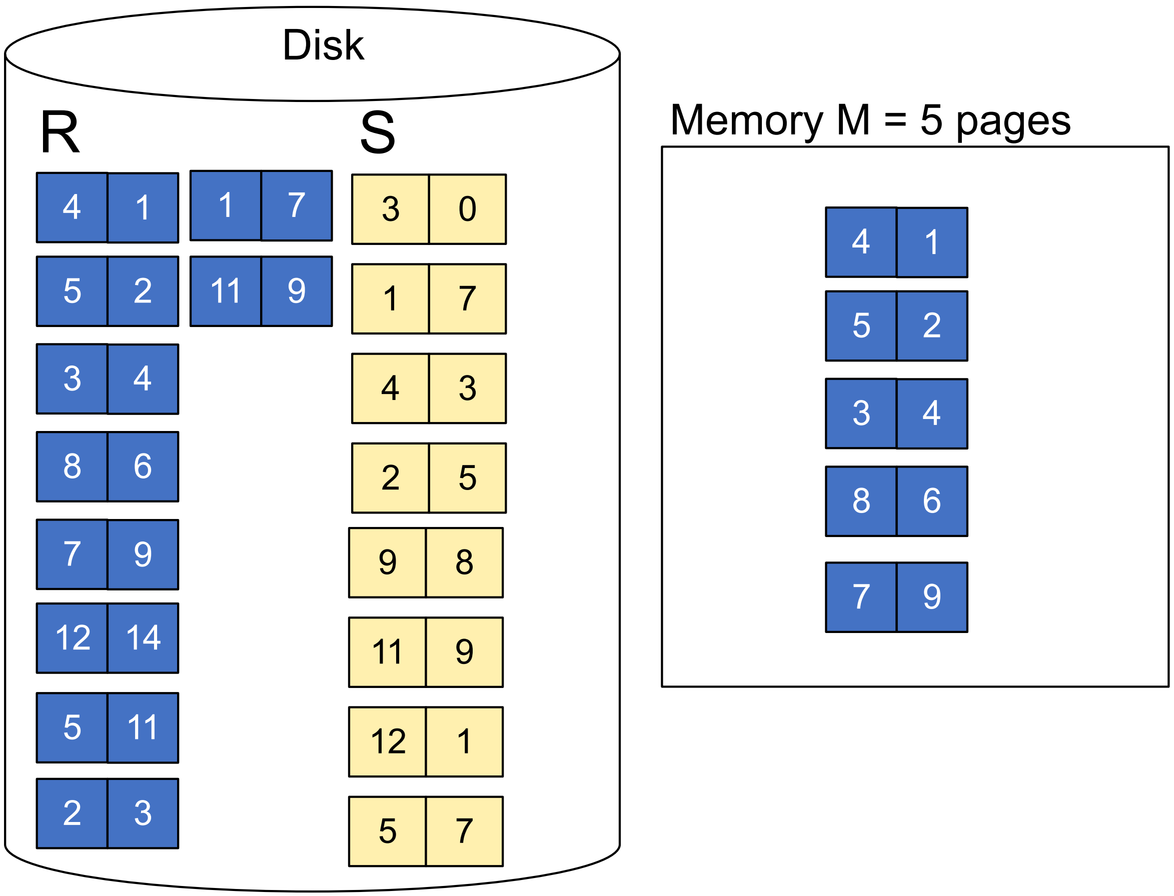
Relation S has 8 pages with 2 tuples per page

Values shown are values of join attribute for each given tuple



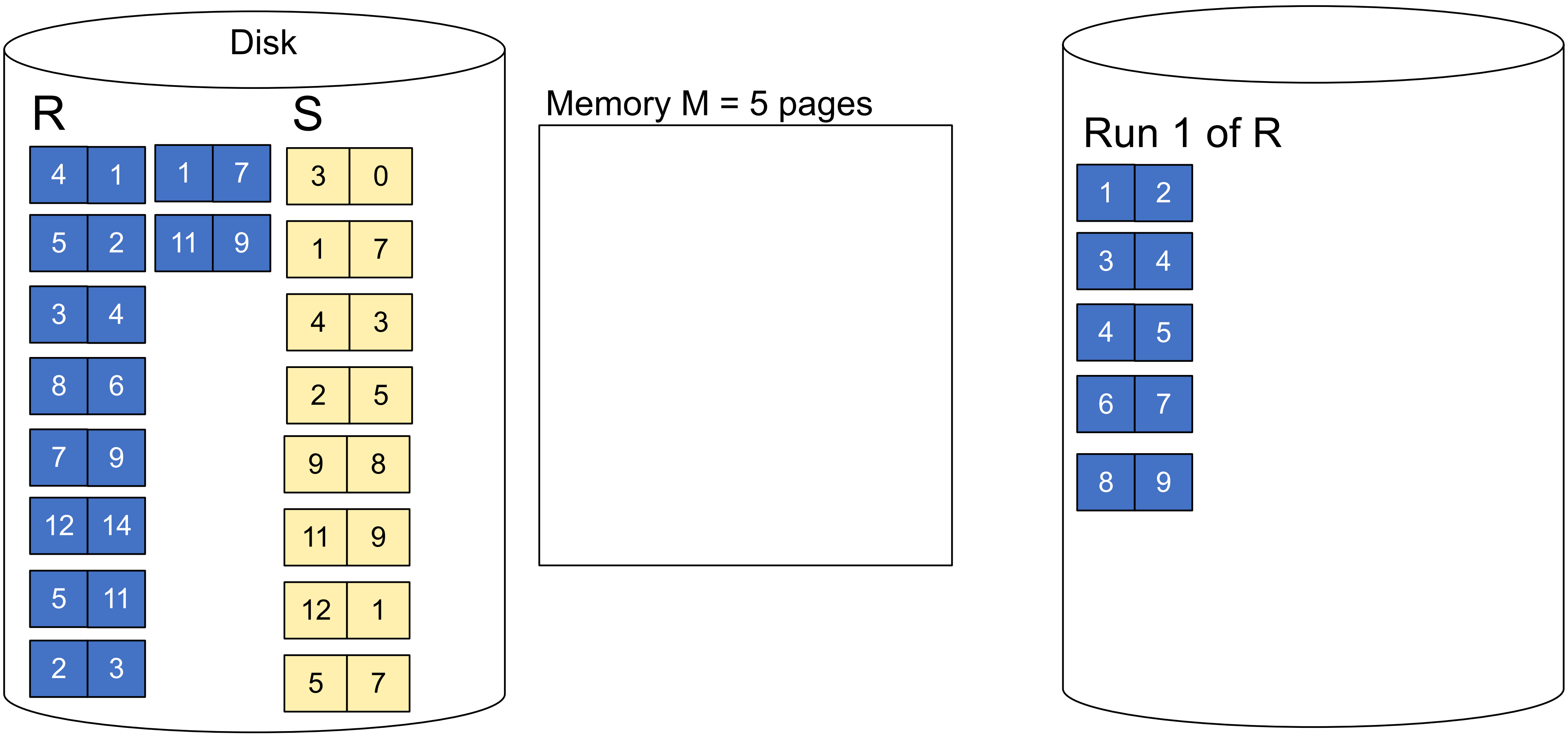
Merge-Join Example

Step 1: Read M pages of R and sort in memory



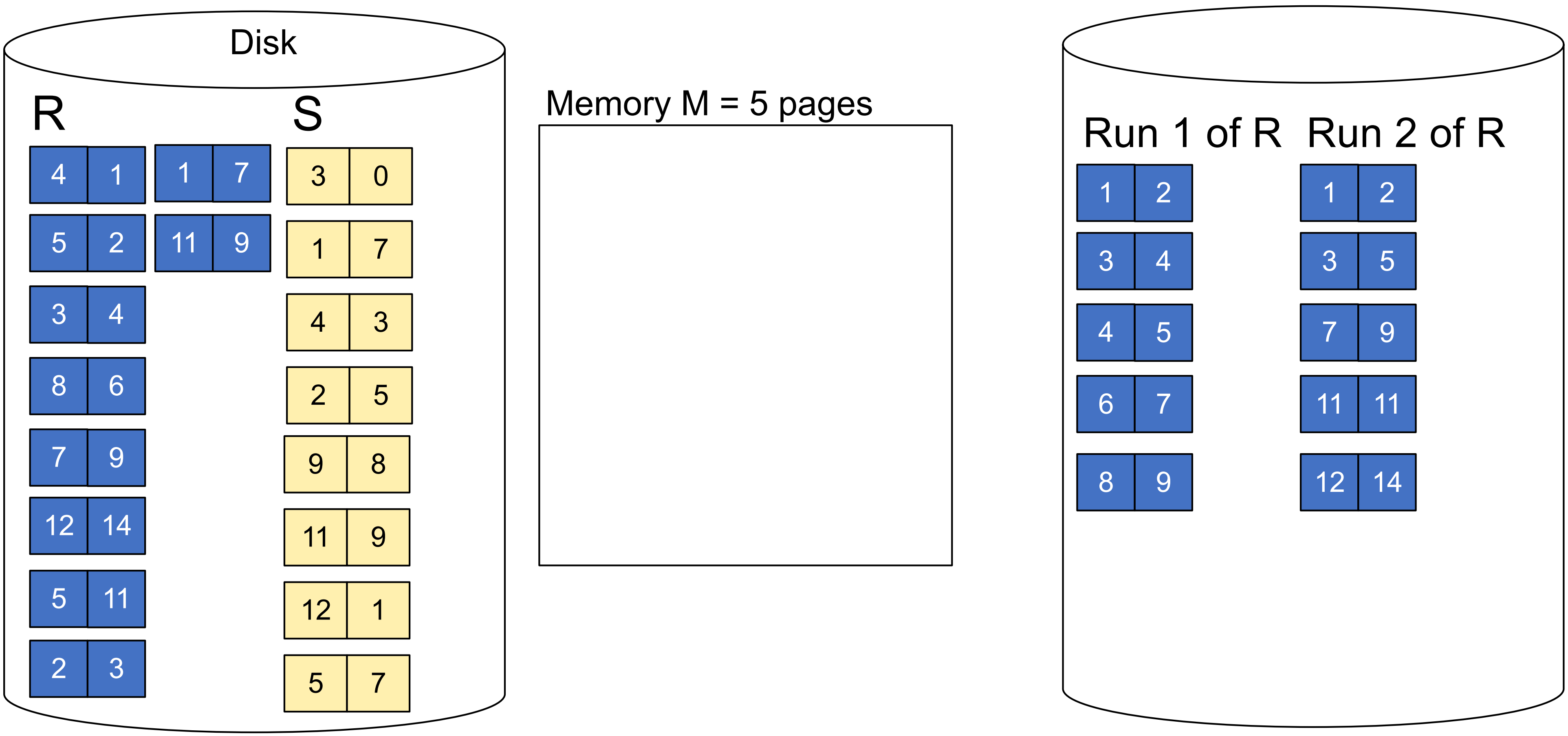
Merge-Join Example

Step 1: Read M pages of R and sort in memory, then write to disk



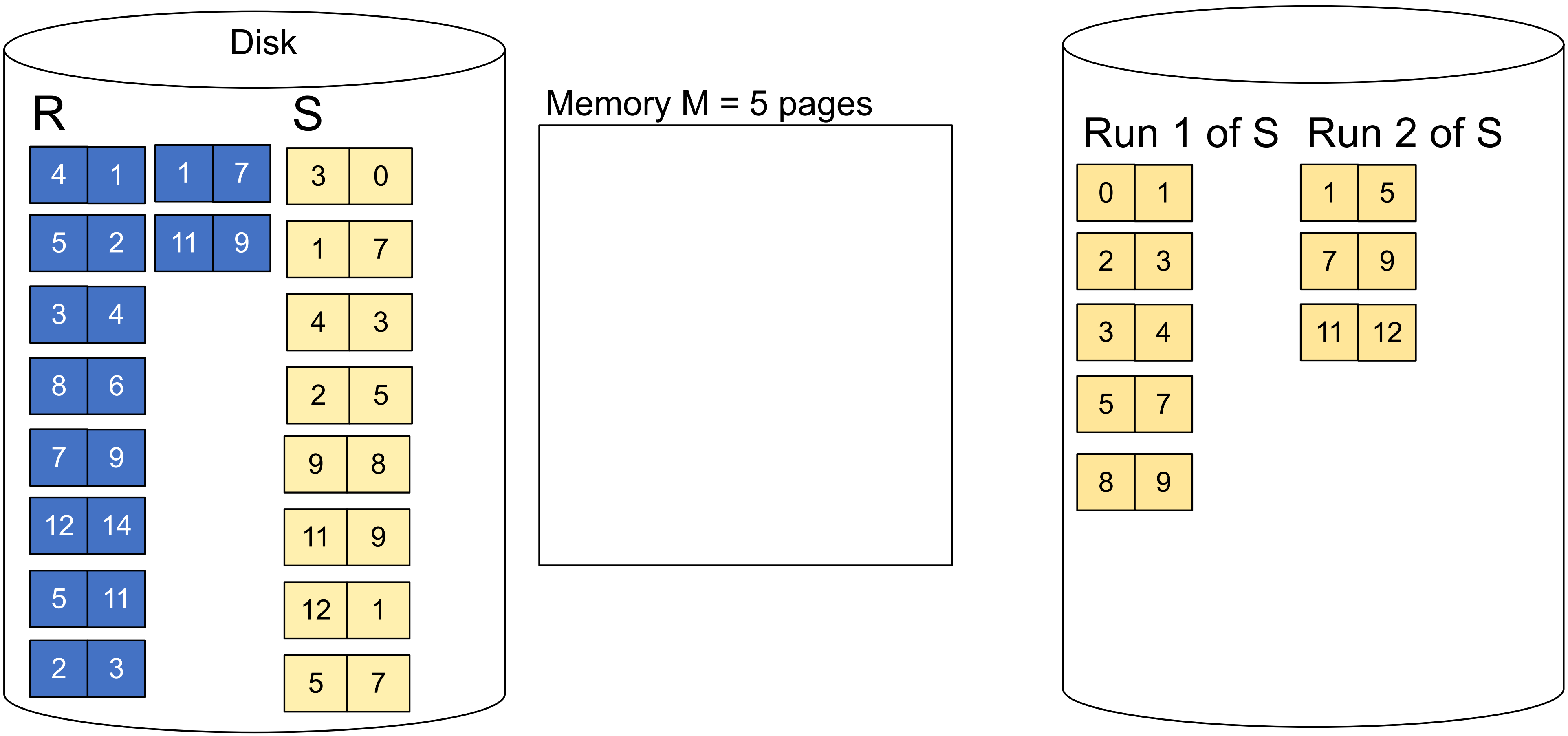
Merge-Join Example

Step 1: Repeat for next M pages until all R is processed



Merge-Join Example

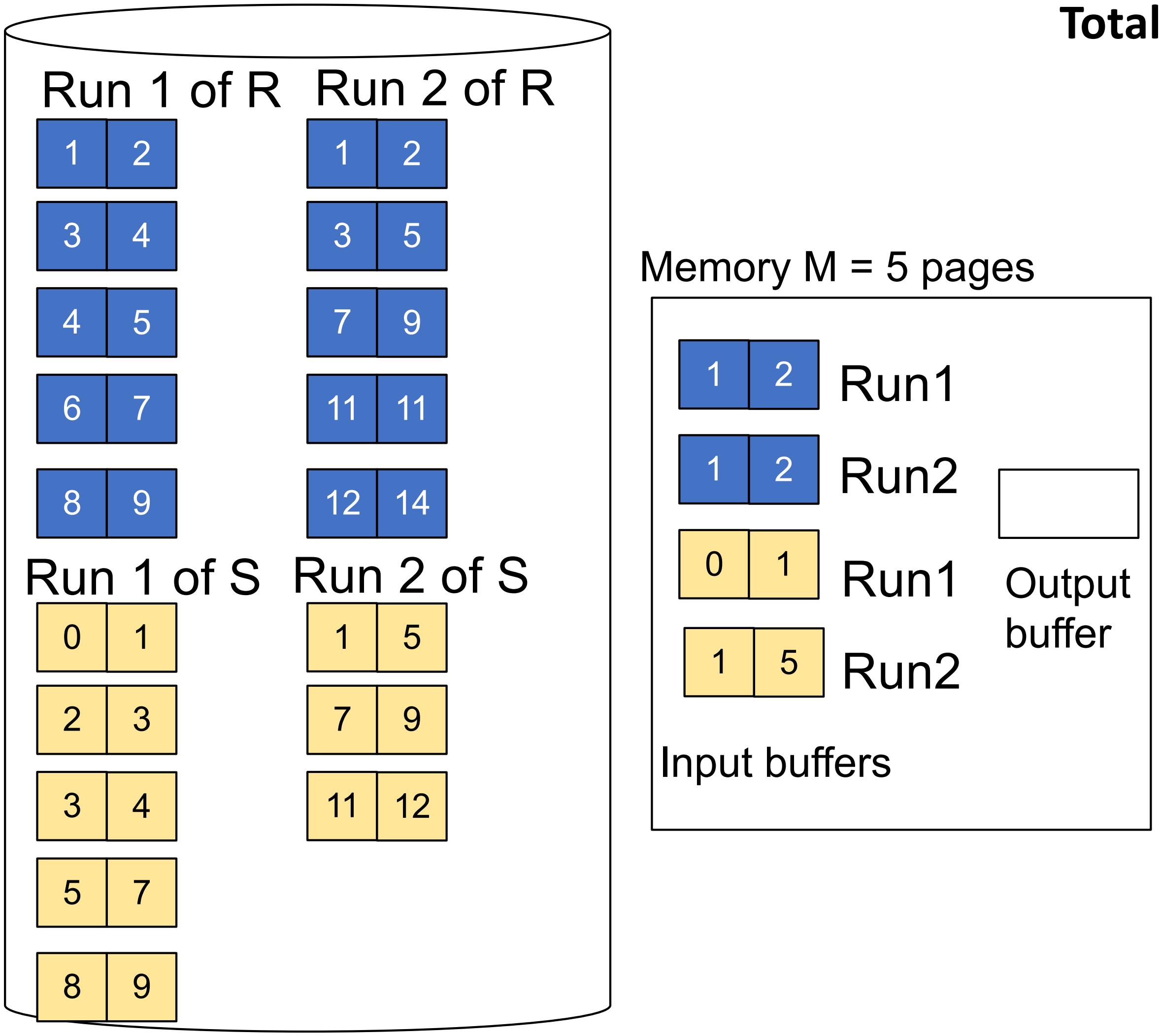
Step 1: Do the same with S



Merge-Join Example

Step 2: Join while merging sorted runs

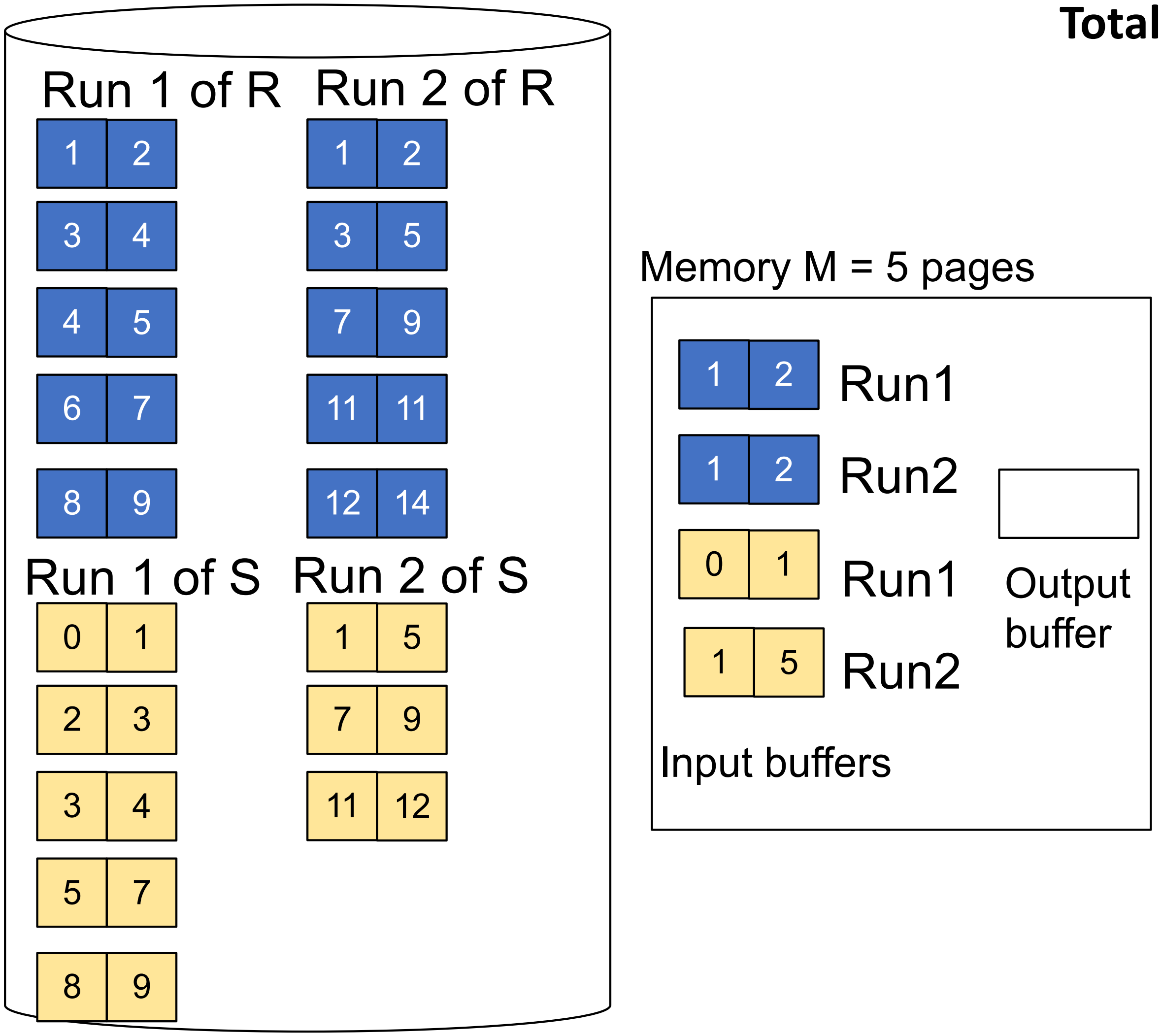
Total cost: 3B(R) + 3B(S)



Step 2: Join while merging
Output tuples

Merge-Join Example

Step 2: Join while merging sorted runs



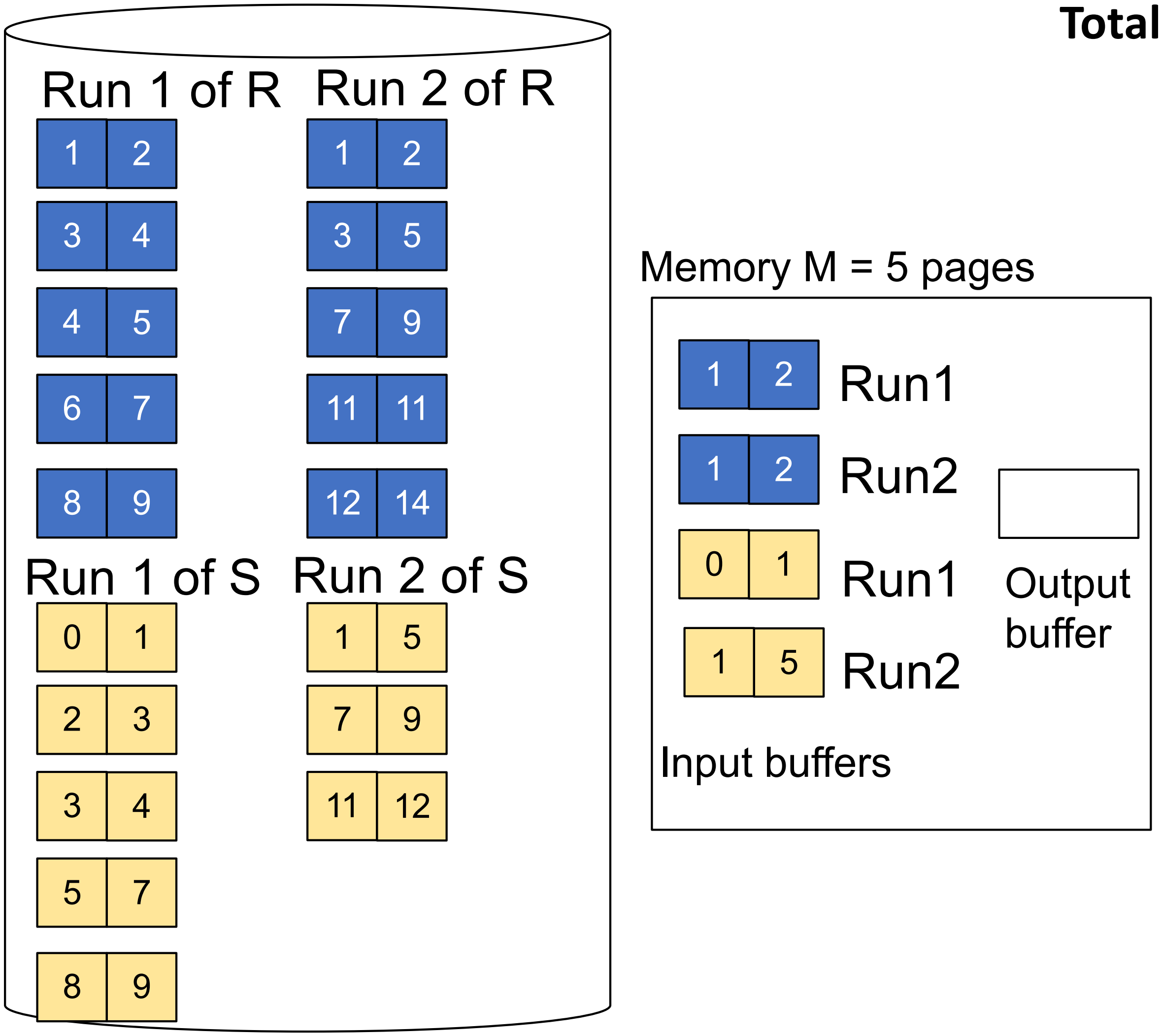
Total cost: 3B(R) + 3B(S)

Step 2: Join while merging
Output tuples

Merge-Join Example

Step 2: Join while merging sorted runs

Total cost: 3B(R) + 3B(S)



Step 2: Join while merging

Output tuples

(1,1)

(1,1)

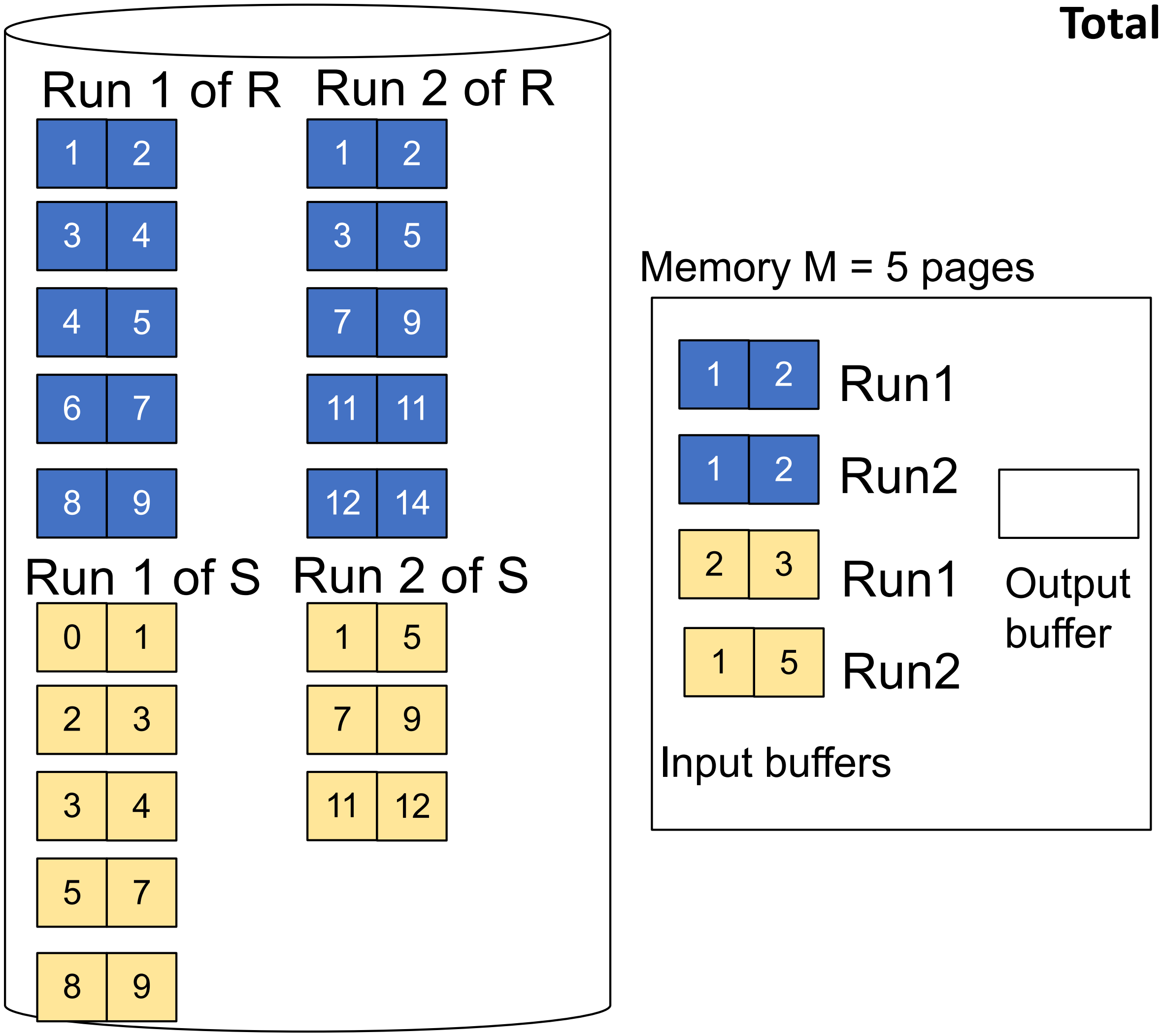
(1,1)

(1,1)

Merge-Join Example

Step 2: Join while merging sorted runs

Total cost: 3B(R) + 3B(S)

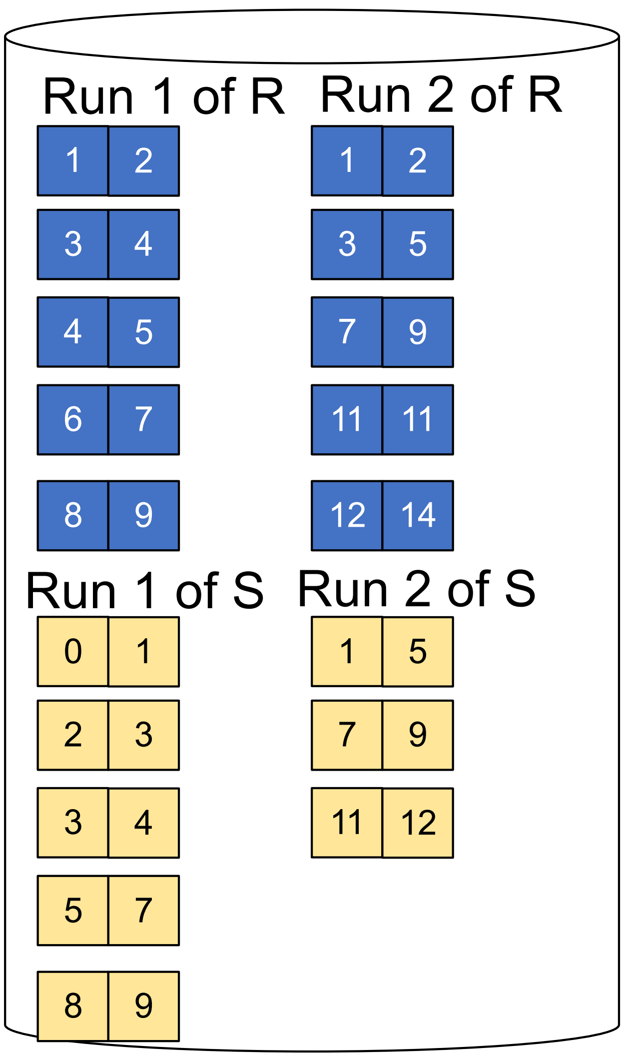


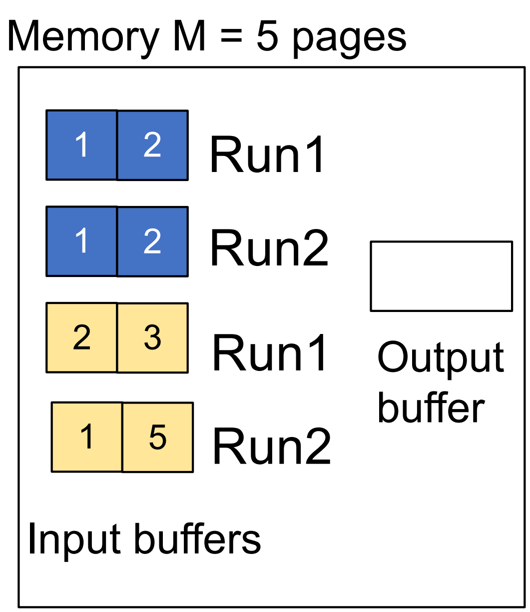
Step 2: Join while merging
Output tuples
(1,1)
(1,1)
(1,1)
(1,1)

Merge-Join Example

Step 2: Join while merging sorted runs

Total cost: 3B(R) + 3B(S)





Step 2: Join while merging

Output tuples

(1,1)

(1,1)

(1,1)

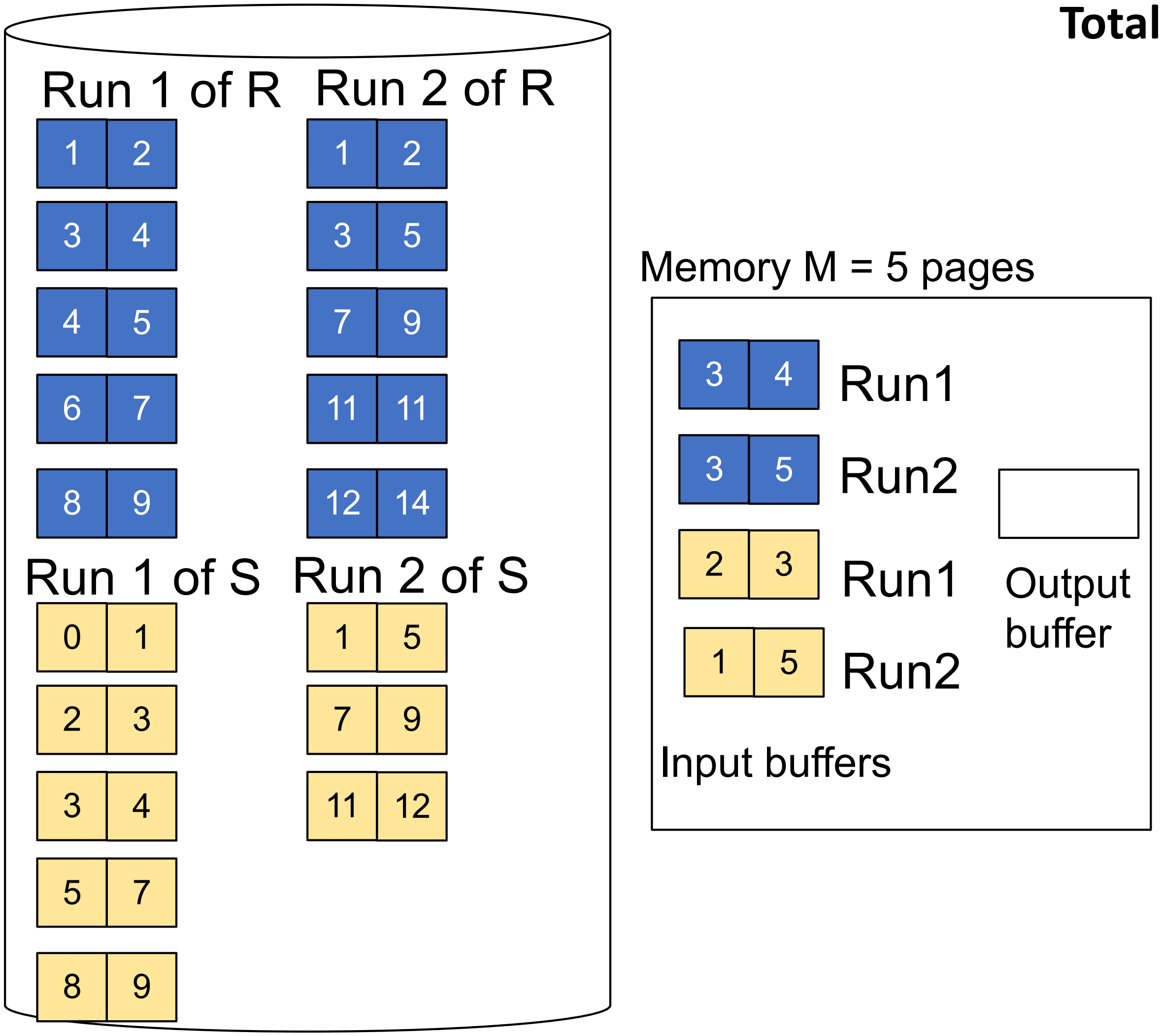
(1,1)

(2,2)

(2,2)

Merge-Join Example

Step 2: Join while merging sorted runs



Total cost: 3B(R) + 3B(S)

Step 2: Join while merging

Output tuples

(1,1)

(1,1)

(1,1)

(1,1)

(2,2)

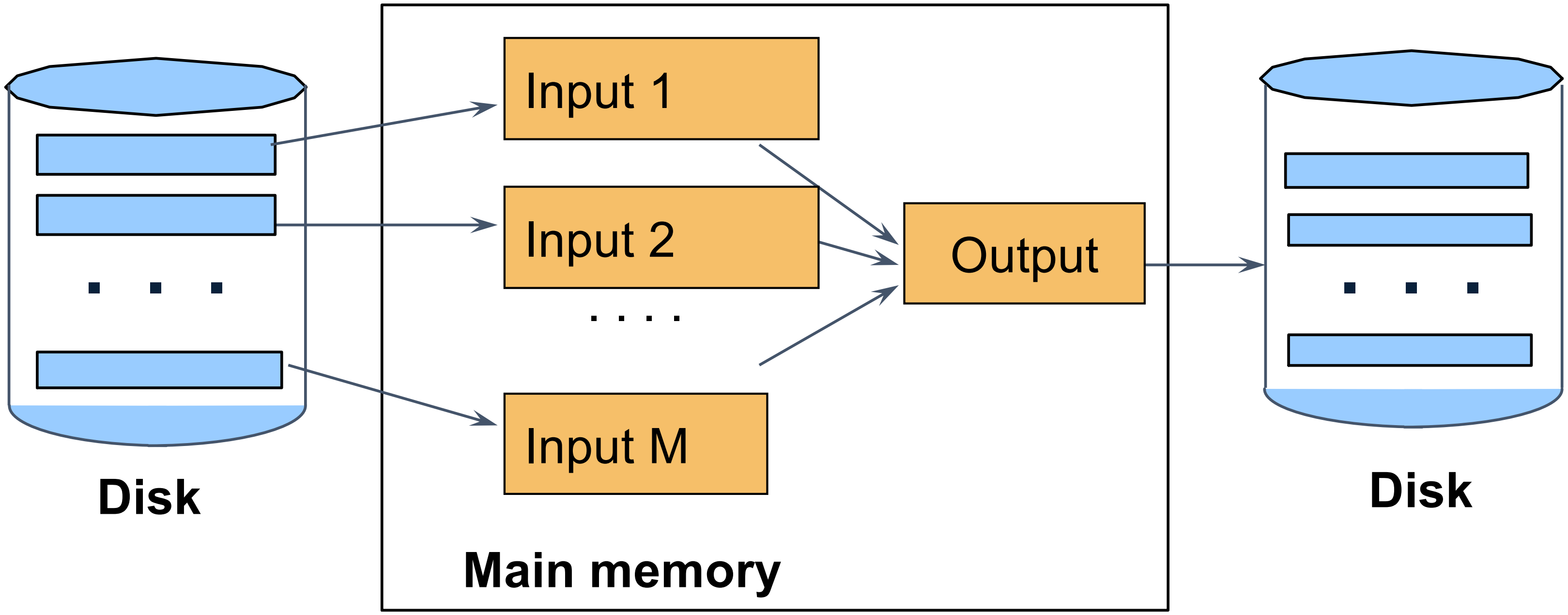
(2,2)

(3,3)

(3,3)

…

Merge-Join



M_1 = B(R)/M runs for R
M_2 = B(S)/M runs for S
Merge-join $M_1 + M_2$ runs;
need $M_1 + M_2 <= M$ to process all runs
i.e. $B(R) + B(S) <= M^2$