CS 186 Discussion #3

Joins

Logistics

- HW2 coming soon, due 9/28
  - Query Planning and Joins
  - Partners Optional
    - Sharing one Github repo

Revisiting SQL…

```
[C]
SELECT sname
FROM
    (SELECT sid
     FROM Reserves
     EXCEPT
          (SELECT sid
          FROM
               (SELECT Reserves.sid, PinkBoats.bid
               FROM Reserves,
                    (SELECT bid
                    FROM Boats
                    WHERE color='pink') PinkBoats
                    EXCEPT SELECT sid, bid
                    FROM Reserves)))
R, Sailors S
WHERE R.sid = S.sid;
```

Joins

Left Outer Join

TABLE Students

SID	Name
23357852	Bob the Builder
24821529	Anonymous
27983421	Nicholas
27994495	Oskicat

TABLE Enrollment

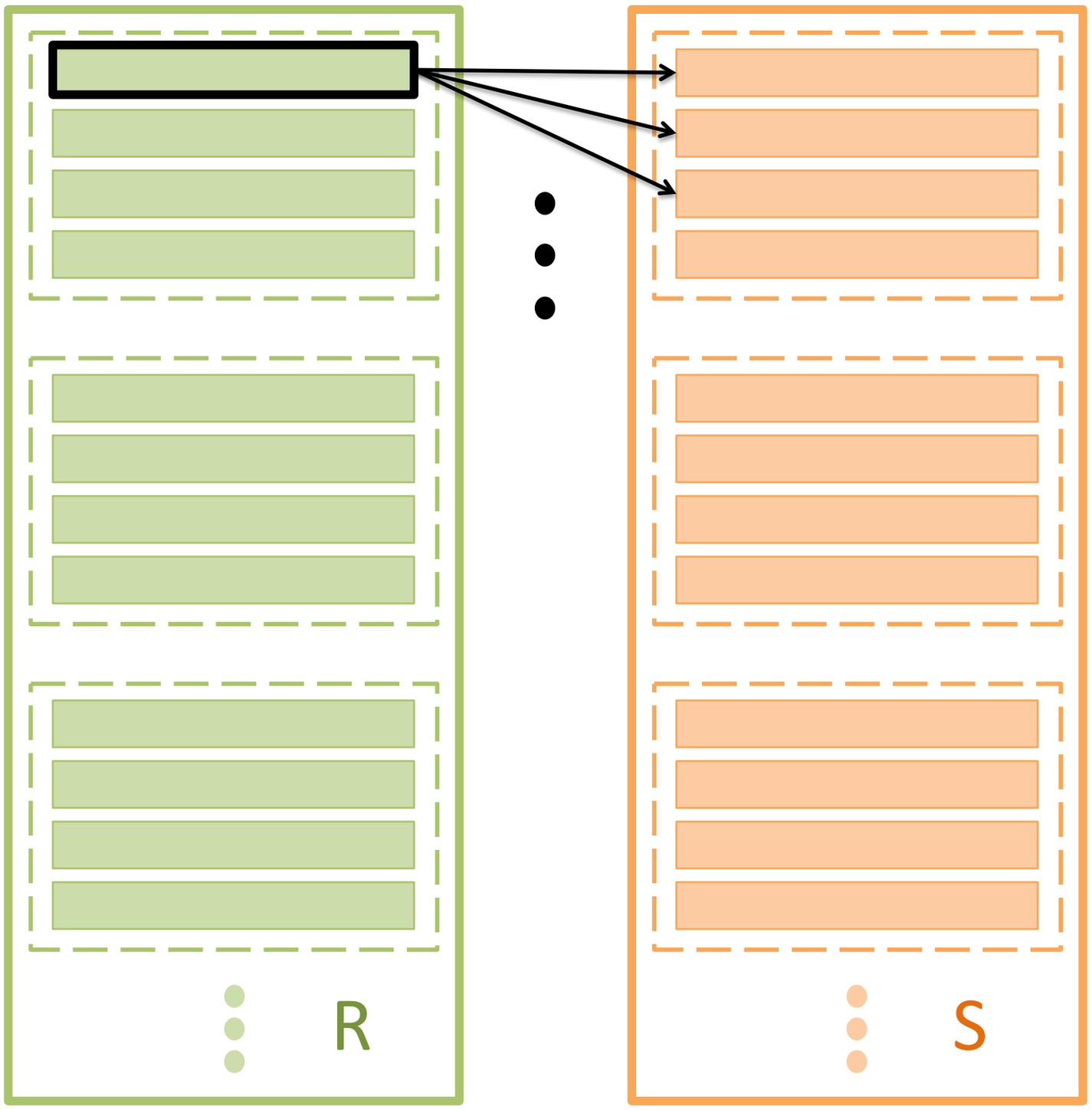
CCN	SID
25227	23357852
26586	23357852
21545	21592421
23495	23374219
26586	27994495

SID	Name	CCN	SID
23357852	Bob the Builder	25227	23357852
23357852	Bob the Builder	26586	23357852
24821529	Anonymous		
27983421	Nicholas		
27994495	Oskicat	26586	27994495

Cost Notation

- [R] = number of pages in Table R
- p_R = number of records per page of R
- |R| = number of records in R
  - » (cardinality)
- Note: $|R| = p_R * [R]$

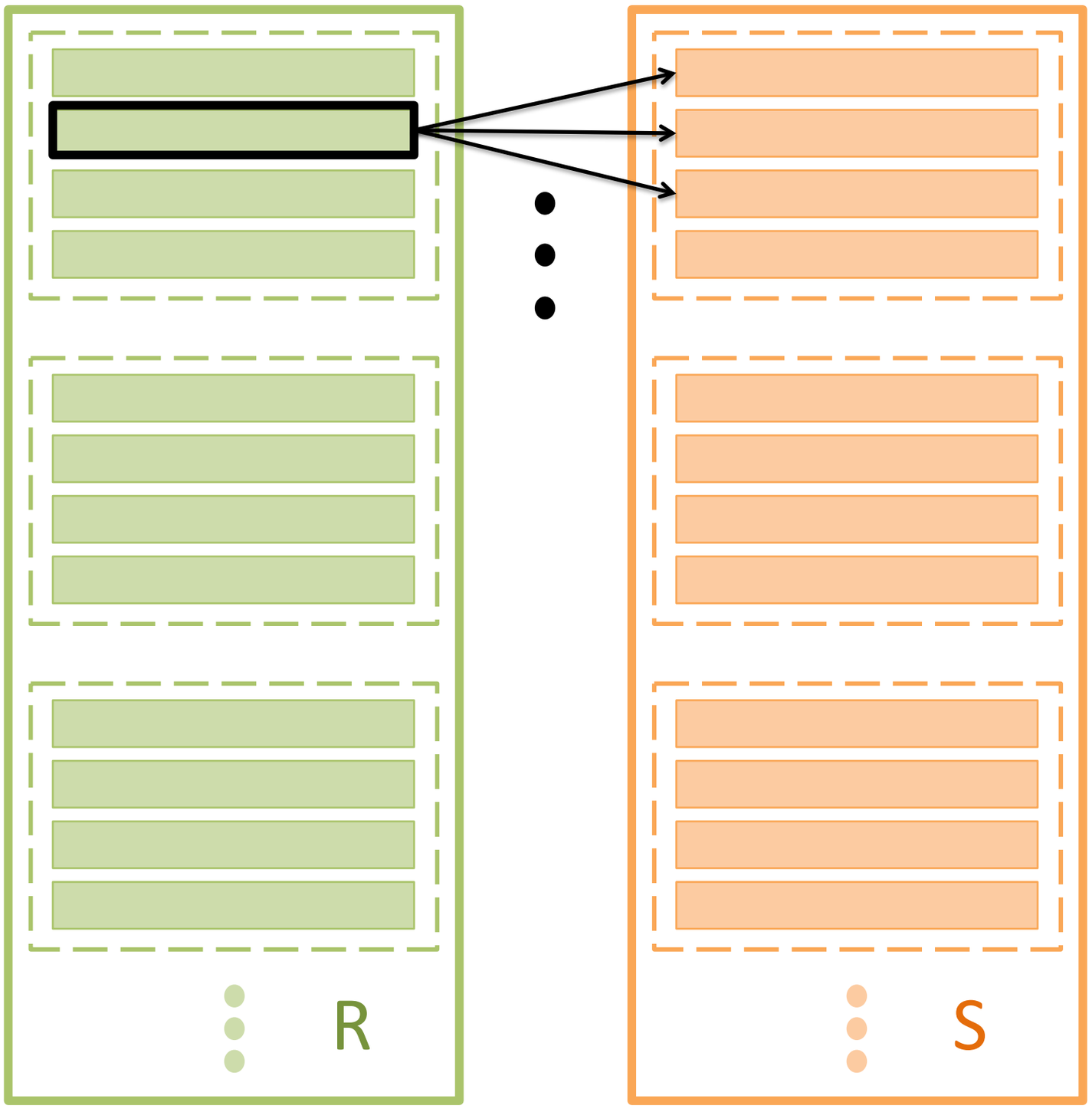
Simple Nested Loop Join



First iteration of outer loop…

Compare with all tuples in S
…and add matches to result

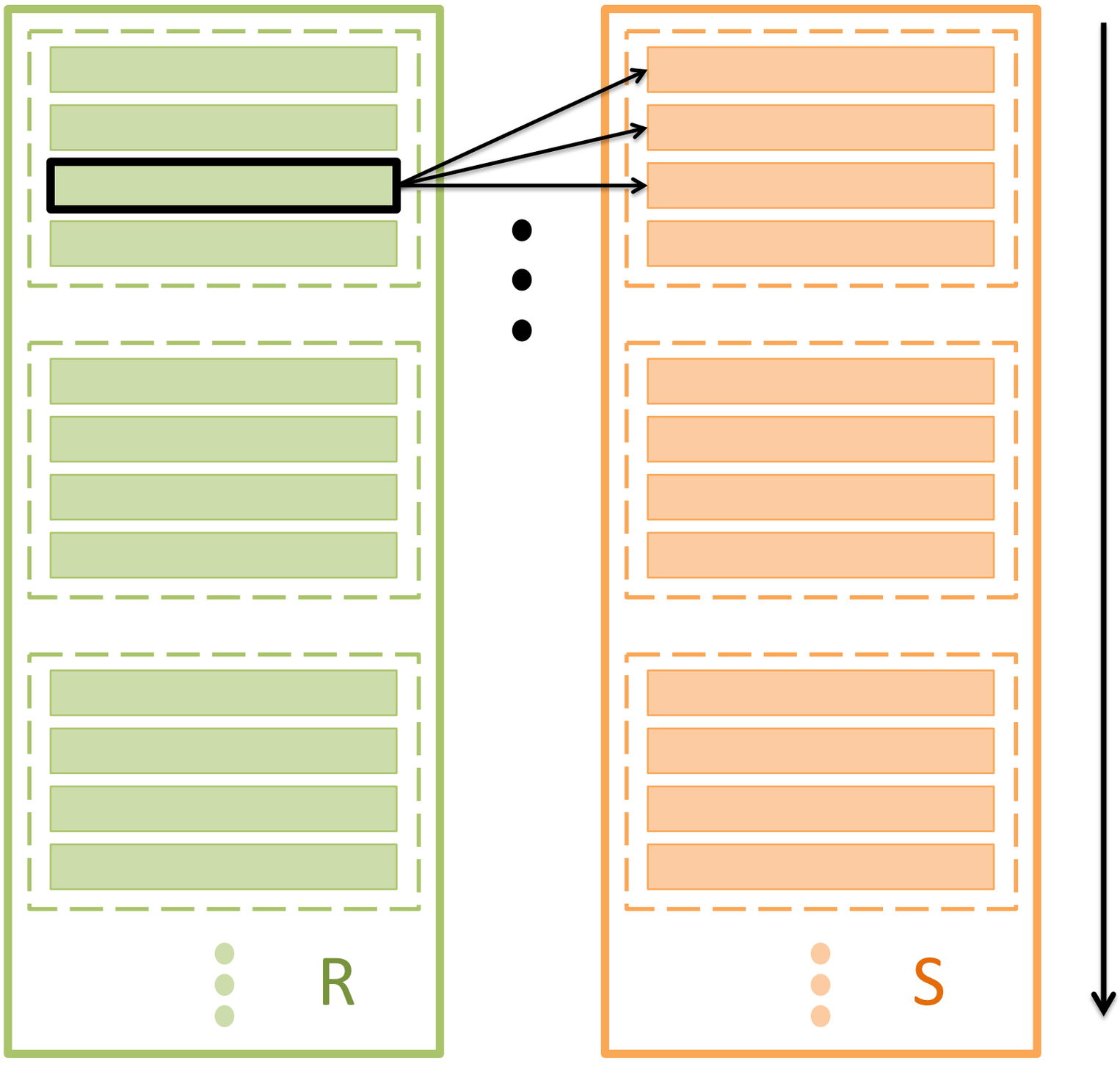
Simple Nested Loop Join



Second iteration of outer loop…

Compare with all tuples in S
…and add matches to result

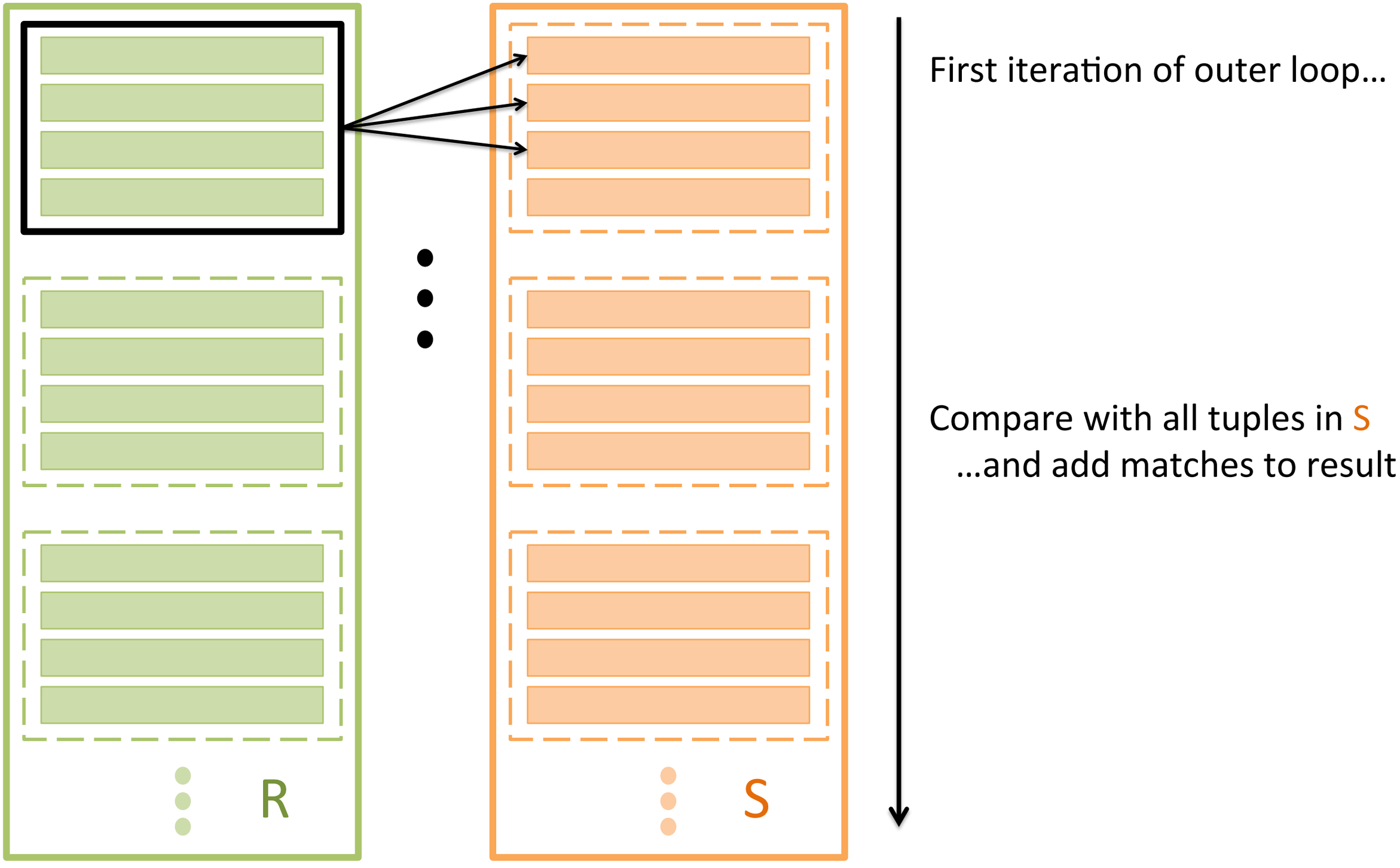
Simple Nested Loop Join



Cost =

$[R] + p_R[R][S]$

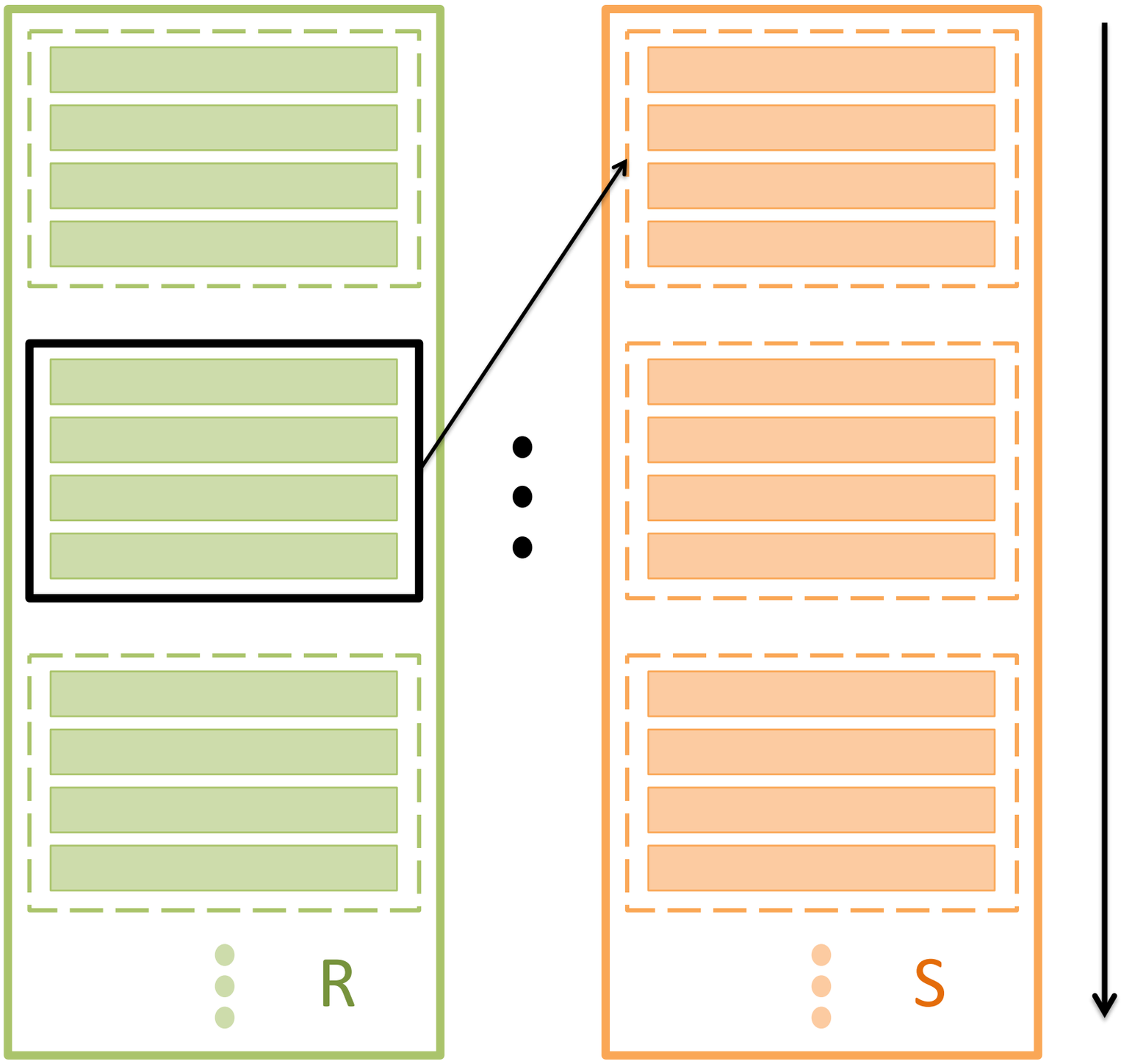
Page-Oriented Nested Loop Join



First iteration of outer loop…

Compare with all tuples in S
…and add matches to result

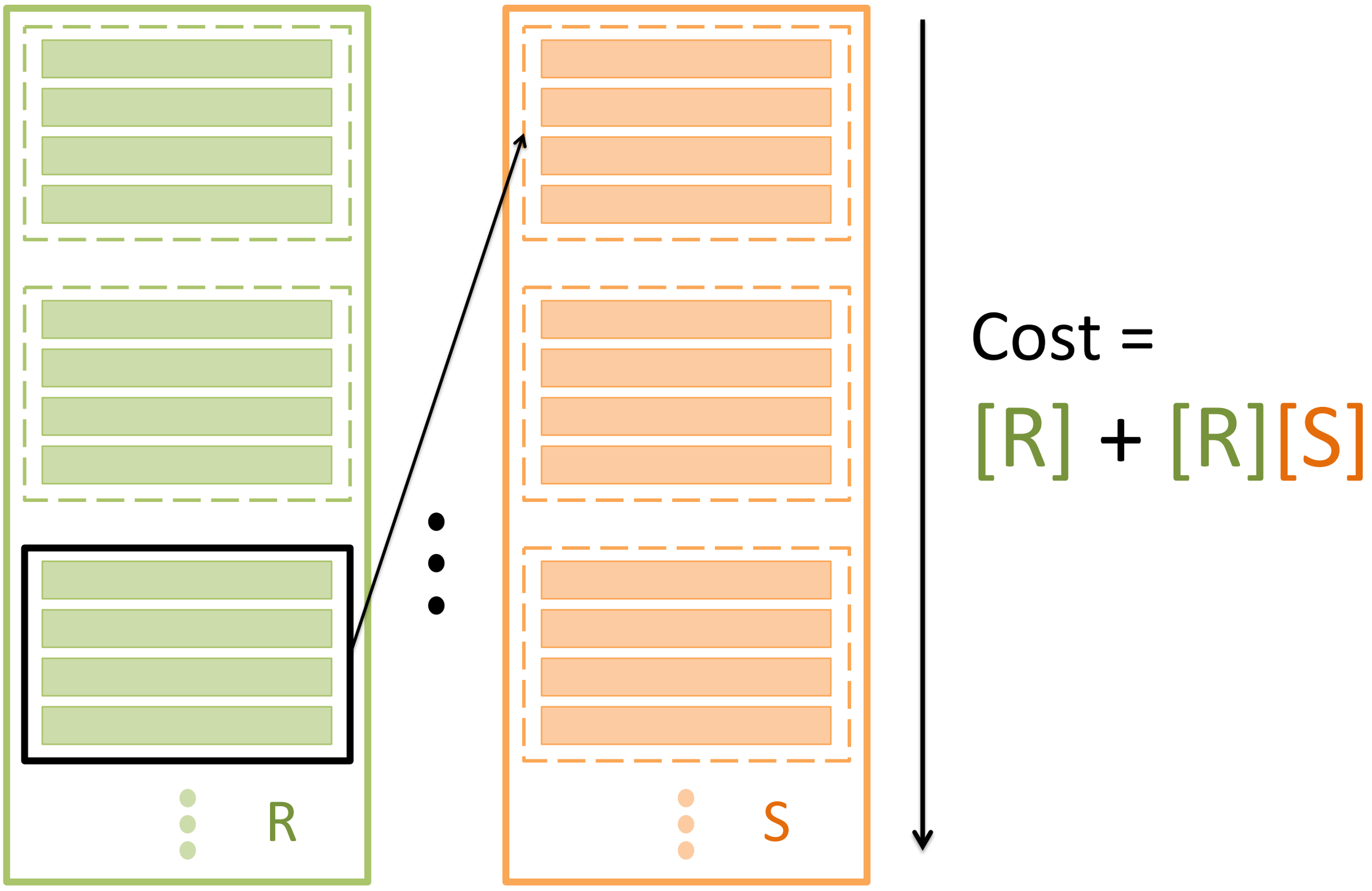
Page-Oriented Nested Loop Join



Second iteration of outer loop…

Compare with all tuples in S
…and add matches to result

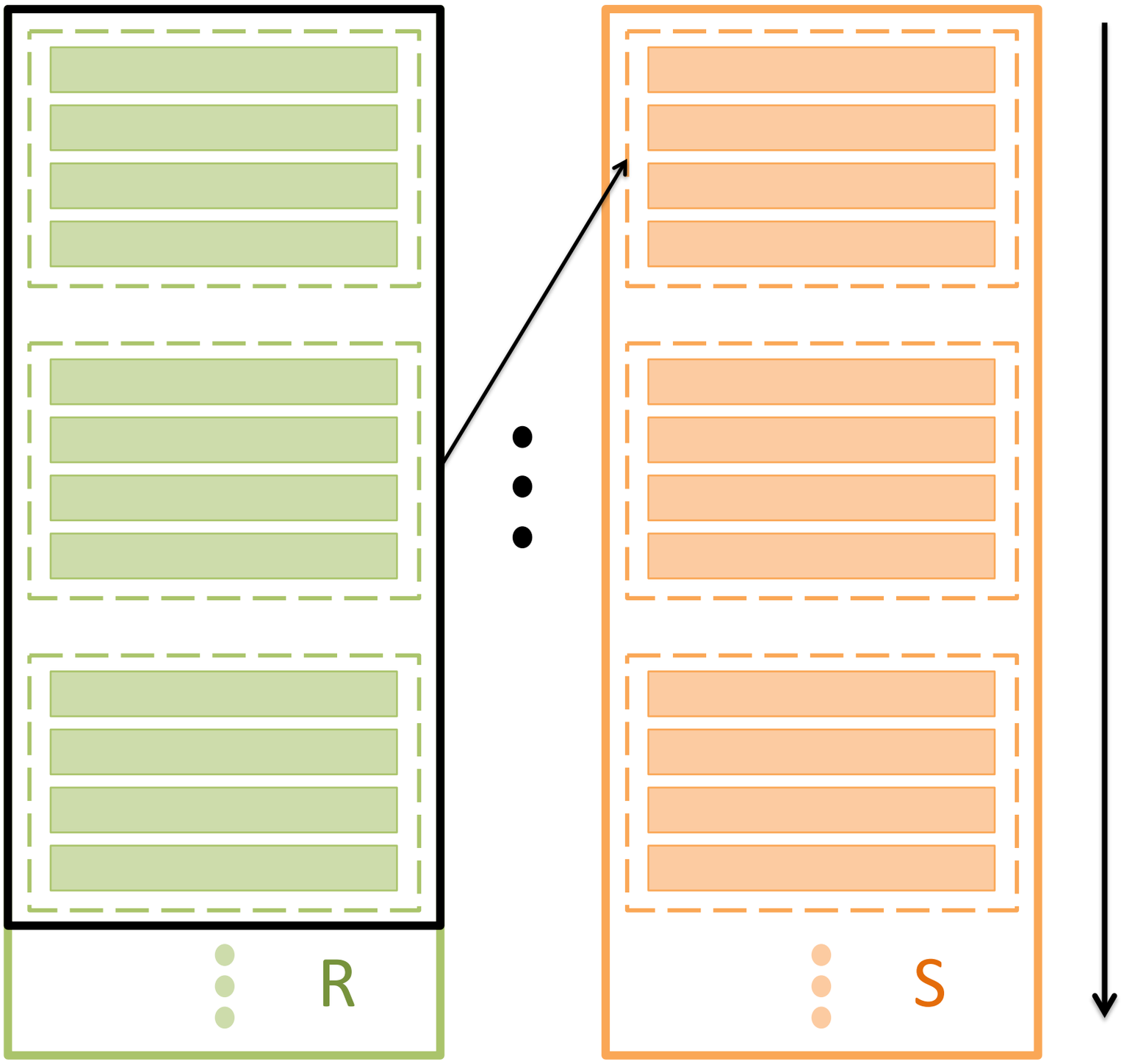
Page-Oriented Nested Loop Join



Cost =

[R] + [R][S]

Chunk Nested Loop Join



First iteration of outer loop…

B pages in memory!
Use B – 2 for R

Compare with all tuples in S
…and add matches to result

Cost of CNLJ?

$$[R] + (\#\ \text{chunks in } R) * [S]$$

$$= [R] + \left(\frac{[R]}{\text{chunksize}}\right) * [S]$$

$$= [R] + \frac{[R][S]}{(B-2)}$$

Sort-Merge Join

1. Sort R and S using external sorting:

$$4[R] + 4[S] \qquad \text{(2 passes)}$$

2. Scan sorted R and sorted S “in tandem” and output matches:

$$[R] + [S]$$

Does this include final write costs?

Optimized Sort-Merge Join

1. Sort R and S using external sorting, but stop before the final pass:

$$2[R] + 2[S]$$

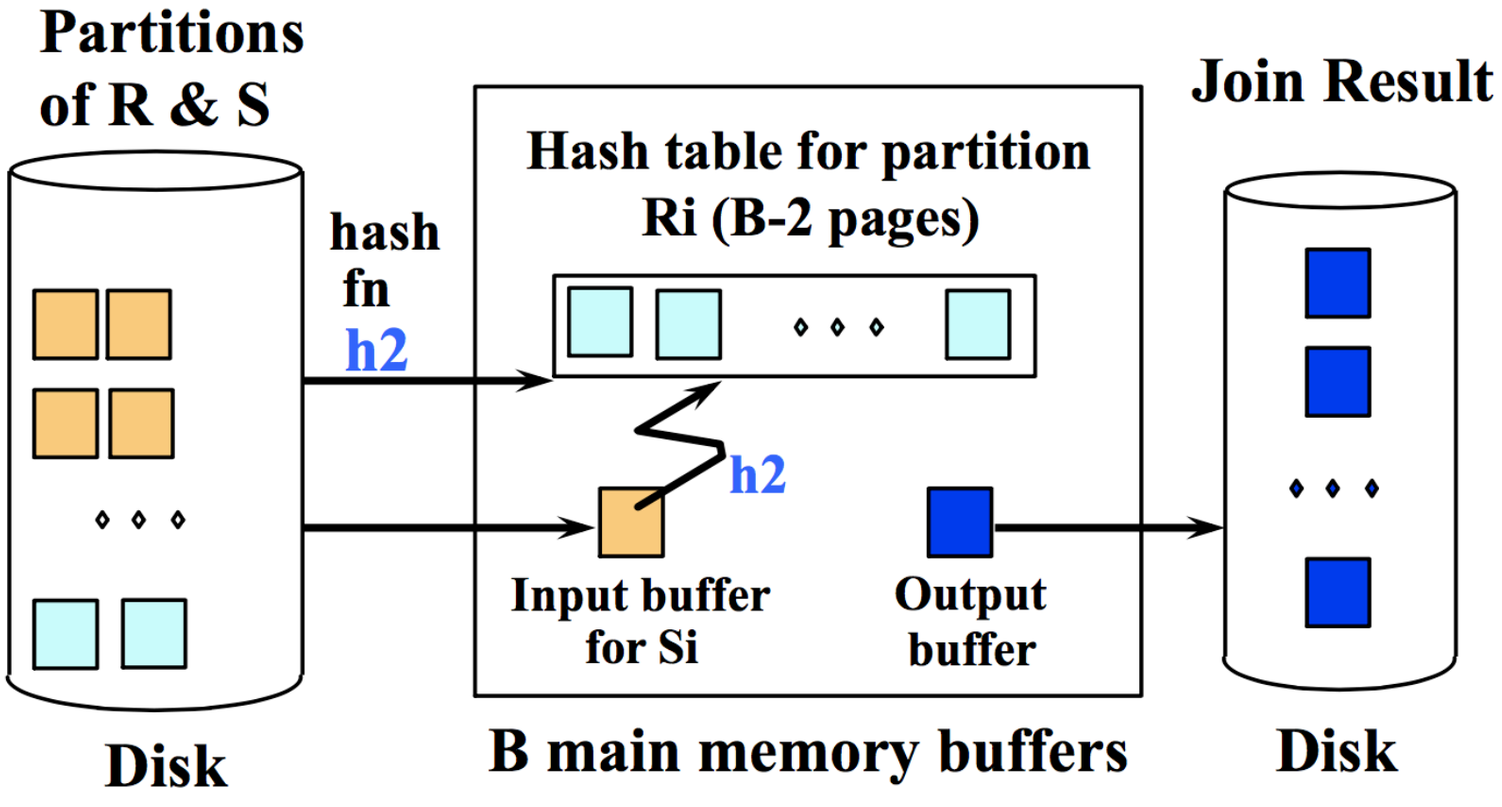
2. Join on the final merge pass!

$$[R] + [S] + [\text{output}]$$

Is [R] + [S] an upper bound on join cost?

Hash Join

- Partition both tables! => 2[R] + 2[S]
- Build a hash table for R
- Then match ("probe") => [R] + [S]:



- Even better with hybrid hashing!

Join Costs Overview

- Chunk Nested Loop Join

$$[R] + {}^{[R][S]}/_{(B-2)}$$

- Sort-Merge Join / Hash Join

$$3[R] + 3[S]$$

Sort-Merge vs. Hash Join

Chunk Nested Loop Join

- Works for cross (Cartesian) products
- Works for non-equality predicates
- Scales nicely with buffer size!

Sort-Merge Join

- Good with sorted input/ output
- Handles data skew + bad hashing (large partitions)
- Good with limited memory

Hash Join

- Good with hashed input/ output
- # passes bounded by **smaller** relation! *Why?*
- Hybrid hashing