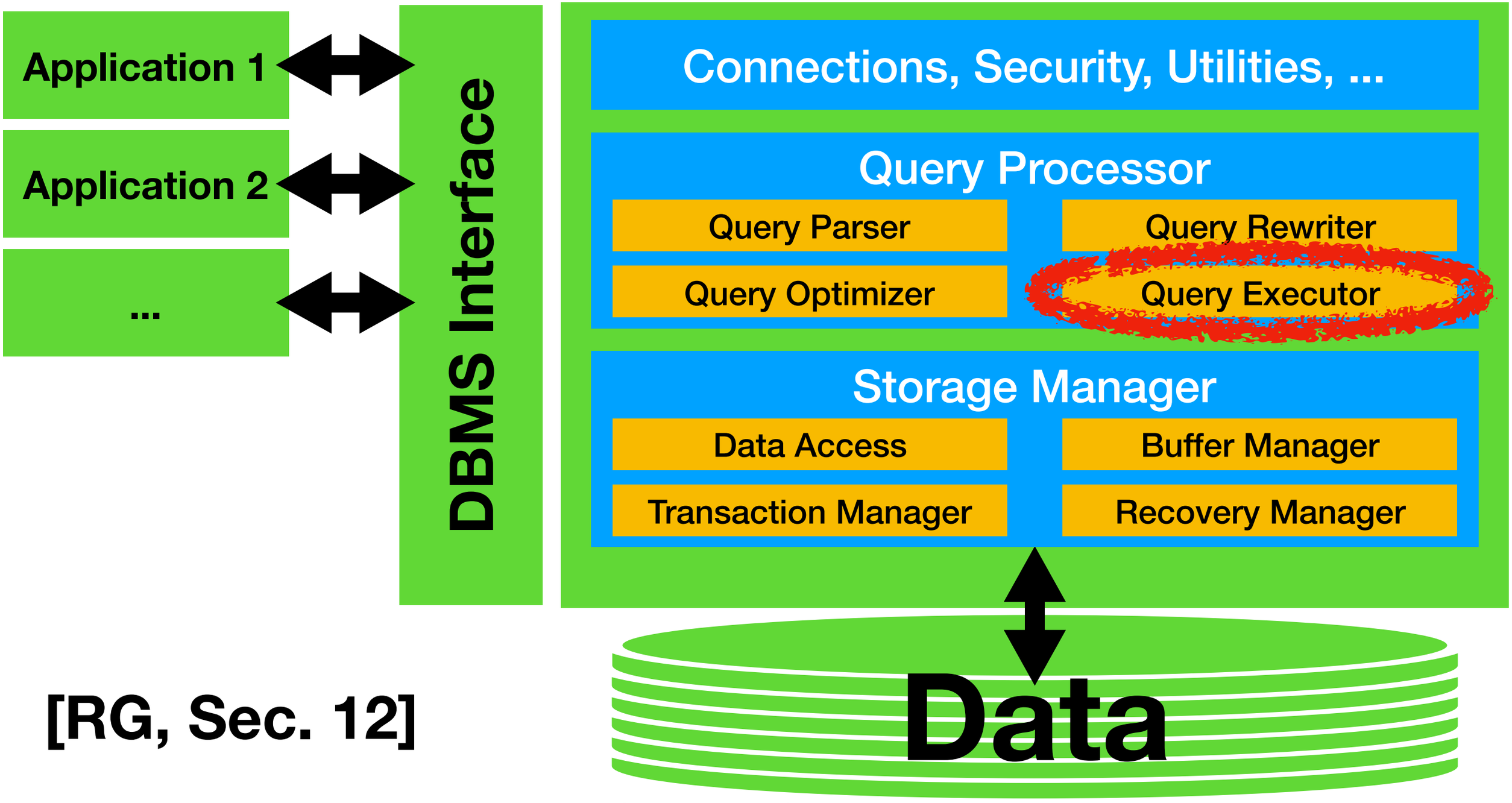
Hash Join,
Sort-Merge Join

Immanuel Trummer
itrummer@cornell.edu
www.itrummer.org

Database Management Systems (DBMS)



[RG, Sec. 12]

Slides by Immanuel Trummer, Cornell University

Alternatives for Equality Joins?

Slides by Immanuel Trummer, Cornell University

Hash Join

- Want tuples with **same value** in join column
- Same value in join column implies **same hash** value
- Join **Phase 1**
  - **Partition data** by hash values in join columns
    - Make partitions small enough to **fit into memory**
- Join **Phase 2**
  - Join each partition pair (same hash value) **separately**

Slides by Immanuel Trummer, Cornell University

More Notations

- **Hash**(Tuple): Calculates hash function for tuple

- **Full**(P): Whether page P has no more space left

- **WriteAndClear**(P): Write P to disk and erase

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

```
For ep in Pages(E):
  LoadPage(ep)
  For et in Tuples(ep):
    Add et to EB[Hash(et)]
    If (Full(EB[Hash(et)])):
      WriteAndClear(EB[Hash(et)])
```

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

For ep in **Pages**(E):

LoadPage(ep) ← *For each page in E*

For et in **Tuples**(ep):

Add et to EB[**Hash**(et)]

If (**Full**(EB[**Hash**(et)])):

WriteAndClear(EB[**Hash**(et)]))

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

For ep in **Pages**(E):

  LoadPage(ep) ⟵ *For each page in E*

  For et in **Tuples**(ep):

    Add et to EB[**Hash**(et)]

      If (**Full**(EB[**Hash**(et)])):

        WriteAndClear(EB[**Hash**(et)])) ⟵ *For each page in E*

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

For ep in **Pages**(E):

LoadPage(ep) ⟵ *For each page in E*

For et in **Tuples**(ep):

Add et to EB[**Hash**(et)]

If (**Full**(EB[**Hash**(et)])):

WriteAndClear(EB[**Hash**(et)])) ⟵ *For each page in E*

Cost = pages in E* IO cost * 2

Hash Join: Phase 1

$\bowtie_{E.Sid=S.Sid}$

For sp in **Pages**(S):

LoadPage(sp) ⟵ *For each page in S*

For st in **Tuples**(sp):

Add st to SB[**Hash**(st)]

If (**Full**(SB[**Hash**(st)])):

WriteAndClear(SB[**Hash**(st)]) ⟵ *For each page in S*

Cost = pages in S* IO cost * 2

Hash Join: Phase 2

$\bowtie_{E.Sid=S.Sid}$

For h in Hash Values:

  LoadPages(EB[h])

  For sp in **Pages**(SB[h]):

    Load(sp)

    For ep in **Pages**(EB[h]), st in sp, et in ep:

      If (et.Sid=st.Sid):

        Output(et ⋈ st)

Hash Join: Phase 2

$\bowtie_{E.Sid=S.Sid}$

For h in Hash Values:

LoadPages(EB[h]) ⟵ *For each page in E*

For sp in **Pages**(SB[h]):

Load(sp) ⟵ *For each page in S*

For ep in **Pages**(EB[h]), st in sp, et in ep:

If (et.Sid=st.Sid):

Output(et ⋈ st)

*Cost = (pages in E in S) * IO cost*

How Much Memory?

- **Phase 1**
  - Space to store **current page** read for partitioning
  - Store one buffer page for **each hash bucket**
- **Phase 2**
  - Store all pages from **one hash bucket**
  - Store **current page** from other table bucket
  - One **output buffer** page

How Many Buckets?

- **Constraint in Phase 1**
  - 1 + Nr. Buckets ≤ Memory
- **Constraint in Phase 2**
  - 2 + Nr. Pages in Smaller Table/Nr. Buckets ≤ Memory
- **Rule of thumb**
  - Want memory > **Sqrt(Nr. Pages in Smaller Table)**

Example

Property	Value
Enrollment Pages	1,000
Student Pages	100
Available Buffer	11
Hash Join Cost	Sqrt(100)<11 Cost: 3*(100+1,000)

Details on Calculations

- Have enough buffer space to execute join as discussed
  - Rule of thumb: Sqrt(100) = 10 < 11
- Phase 1 reads and writes each input table page once
  - Cost is 2 * (100 + 1,000)
- Phase 2 reads and writes each input table page once
  - However, we do not count the output cost, as usual
  - Therefore, we only count cost 1 * (100 + 1,000)

What If We Lack Memory?

- Number of buffer pages limits number of **output buckets**
- Not enough buckets means **too much data** per bucket
- Prevents us from **loading one bucket** entirely in Phase 2
- Hence, perform **multiple passes** over data in phase 1
  - In each pass, buckets are partitioned into **sub-buckets**
  - Iterate until data per bucket **fits** into main memory

Sort-Merge Join: Idea

- Also specific to **equality** join conditions
- **Phase 1 (Sort)**
  - **Sort** joined tables on the join column
- **Phase 2 (Merge)**
  - Efficiently **merge** sorted tables together

Join Phase 1: Overview

- Lots of **sorting algorithms** proposed in the literature
- However, typically assume that we access **single** entries
- But random data access can be very **inefficient**
- Hence, want to access **pages** of entries instead
- Need specialized ("**external**") sort algorithms

Algorithm Sketch

- **Step 1**: load chunk of data and **sort**, write back to disk
- **Step 2 .. n**: **merge** sorted runs to produce larger runs
- Each merging step **reduces** number of runs (but longer)
- Finally, have only **one sorted run** left - we're done!

Details on Step 1

- Assume we have **B buffer pages** available
- **Load chunks** of B pages into the buffer
- For each chunk, **sort** by standard sort algorithm
  - Can use standard algorithm as **all data in memory**
- Then, **write** sorted data to hard disk
- A sorted sequence of data is called a "**run**"

Step 1 Example

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	12, 29	9, 10	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

1, 8	12, 29	9, 10

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	12, 29	9, 10	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

1, 8	9, 10	12, 29

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	12, 29	9, 10	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

1, 8	9, 10	12, 29

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

1, 8	9, 10	12, 29

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

15, 3	26, 4	14, 17

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

3, 4	14, 15	17, 26

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	15, 3	26, 4	14, 17	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

3, 4	14, 15	17, 26

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

19, 54	8, 90	6, 12

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

6, 8	12, 19	54, 90

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	19, 54	8, 90	6, 12	5, 73	2, 42	3, 9

Step 1 Example

6, 8	12, 19	54, 90

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	5, 73	2, 42	3, 9

Step 1 Example

6, 8	12, 19	54, 90

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	5, 73	2, 42	3, 9

Step 1 Example

5, 73	2, 42	3, 9

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	5, 73	2, 42	3, 9

Step 1 Example

2, 3	5, 9	42, 73

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	5, 73	2, 42	3, 9

Step 1 Example

2, 3	3, 5	9, 73

Buffer Pool (3 Pages)

Hard Disk (12 Pages)

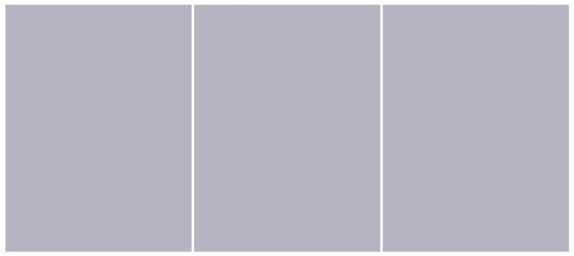
1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

Details on Steps 2 .. n

- (Still have B buffer pages available)
- Enables us to **merge B-1 sorted runs** into one in one step
  - **Load first page** of each sorted run into B-1 pages
  - **Copy minimum entry** in input buffers to output buffer
    - If output buffer full, **write to disk and clear**
  - **Erase minimum** entry from input buffer
    - If input buffer becomes empty, **load next page**

Step 2 Example

Input 1 Input 2 Output



Buffer Pool (3 Pages)

Hard Disk

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

Step 2 Example

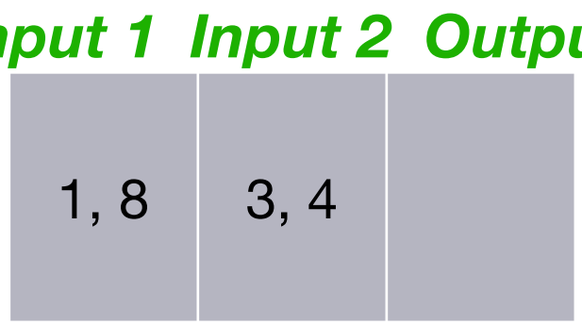
Input 1 Input 2 Output

Buffer Pool (3 Pages)

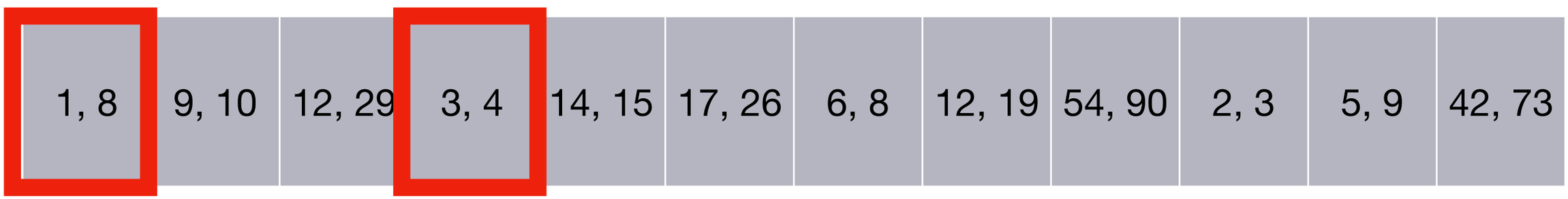
Hard Disk

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

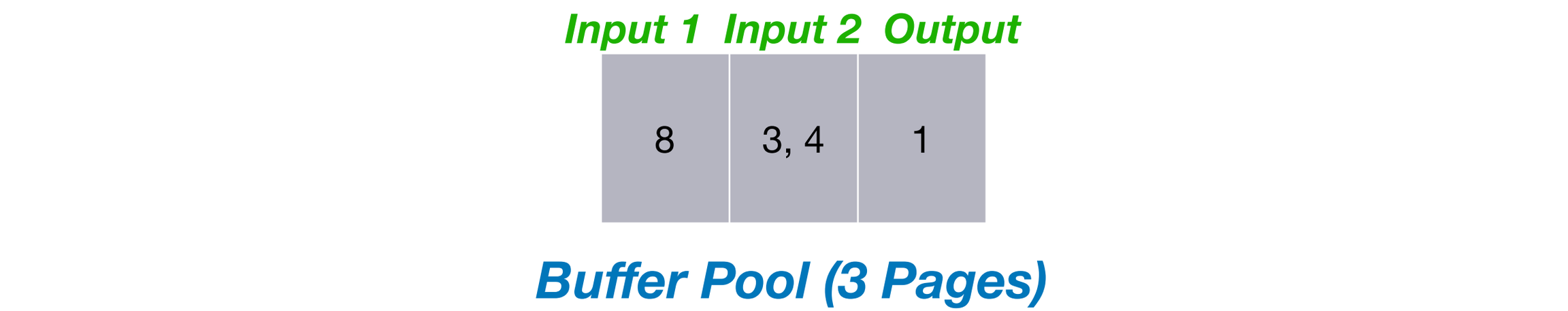
Step 2 Example



Buffer Pool (3 Pages)



Step 2 Example



Step 2 Example

Input 1 Input 2 Output

8	4	1, 3

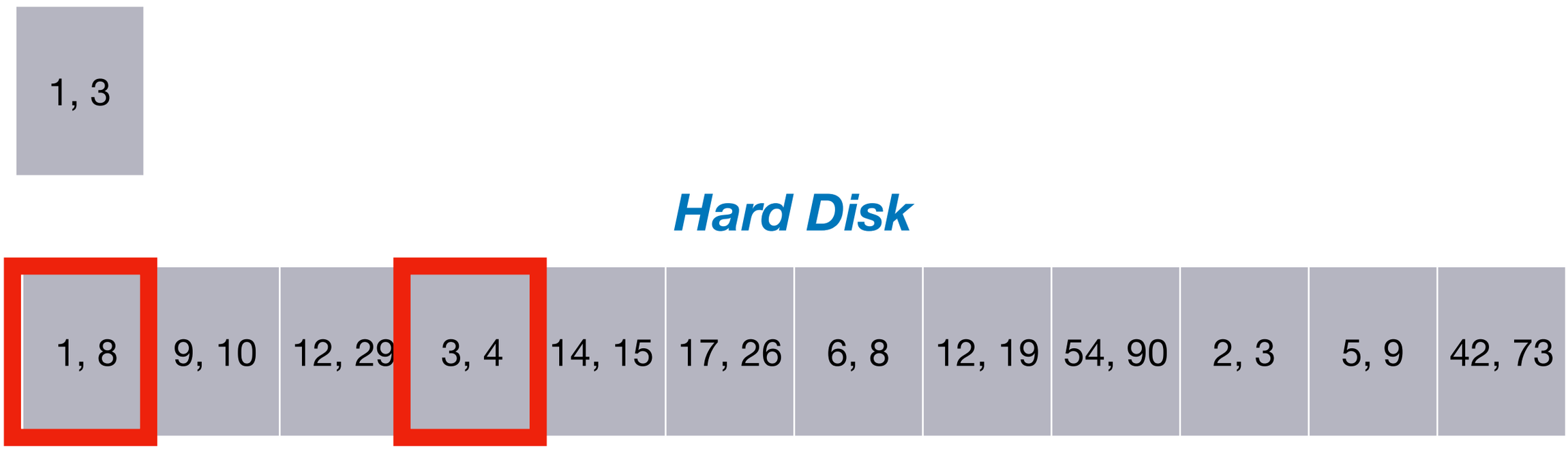
Buffer Pool (3 Pages)

Hard Disk

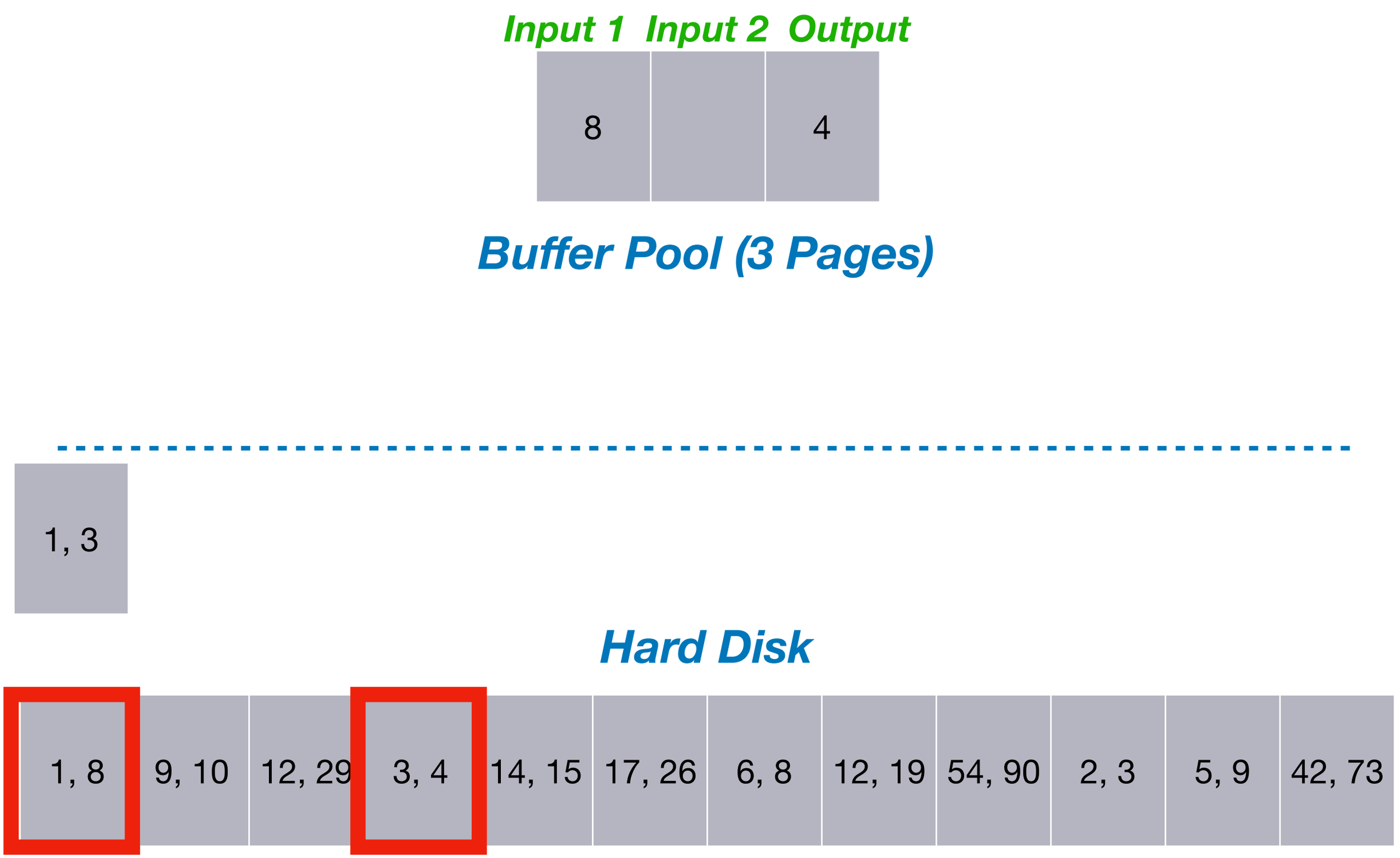
1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

Step 2 Example

Buffer Pool (3 Pages)



Step 2 Example



Step 2 Example

Input 1 Input 2 Output

8	14, 15	4

Buffer Pool (3 Pages)

1, 3

Hard Disk

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

Step 2 Example

Input 1 ***Input 2*** ***Output***

	14, 15	4, 8

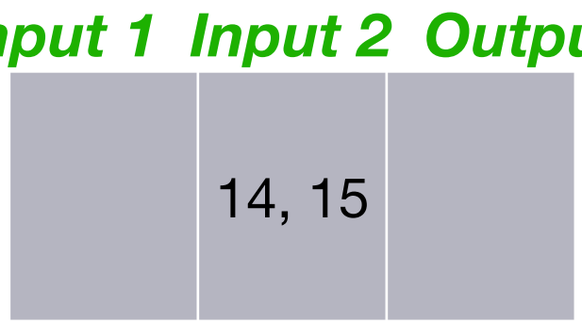
Buffer Pool (3 Pages)

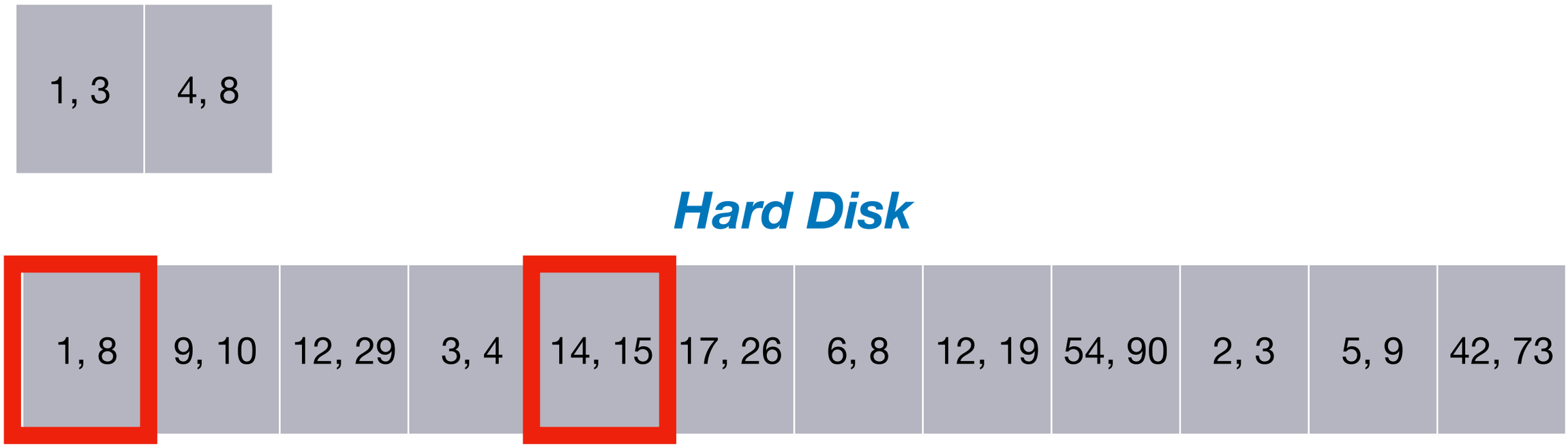
1, 3

Hard Disk

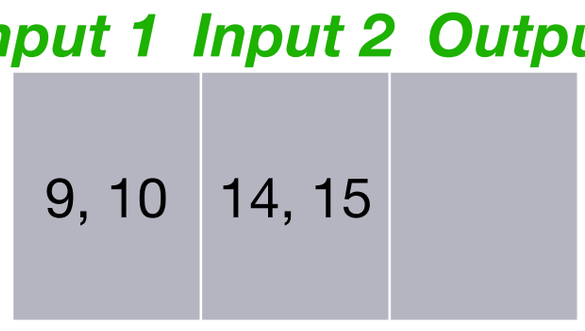
1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

Step 2 Example

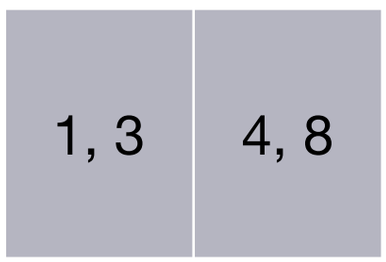




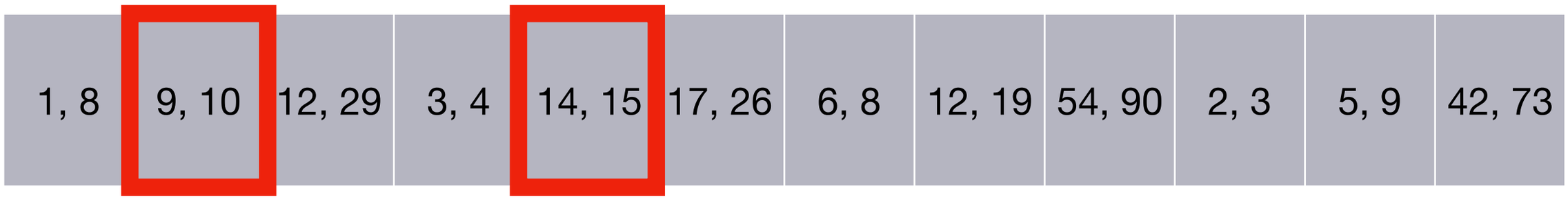
Step 2 Example



Buffer Pool (3 Pages)



Hard Disk



Step 2 Example

Input 1	*Input 2*	*Output*
10	14, 15	9

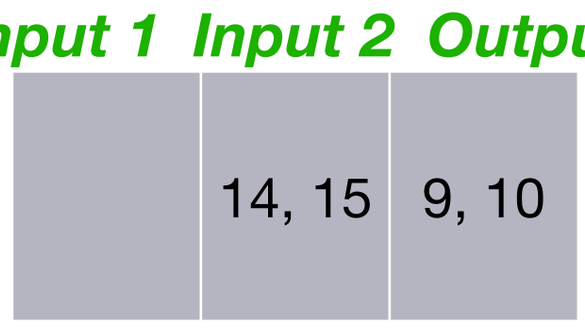
Buffer Pool (3 Pages)

1, 3	4, 8

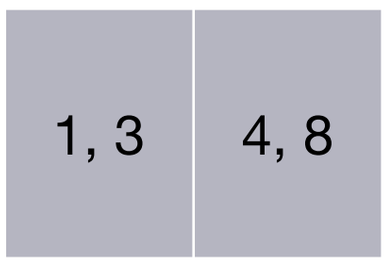
Hard Disk

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

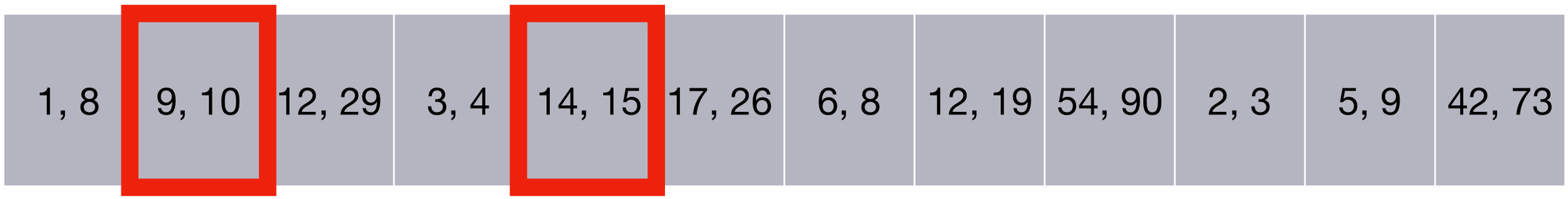
Step 2 Example



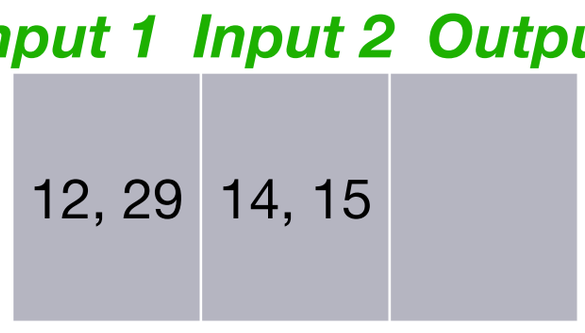
Buffer Pool (3 Pages)



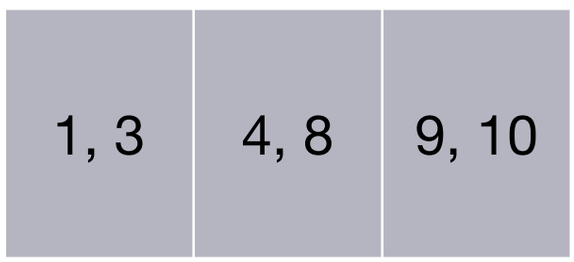
Hard Disk



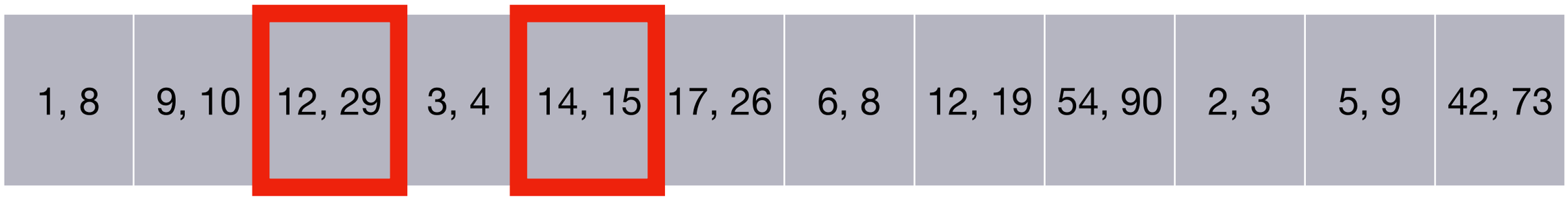
Step 2 Example



Buffer Pool (3 Pages)



Hard Disk



Step 2 Example

Input 1	*Input 2*	*Output*
29	14, 15	12

Buffer Pool (3 Pages)

1, 3	4, 8	9, 10

Hard Disk

1, 8	9, 10	12, 29	3, 4	14, 15	17, 26	6, 8	12, 19	54, 90	2, 3	5, 9	42, 73

Example Summary

- Have **12 pages** to sort with **3 buffer** pages
- First step: produce **4 sorted runs** of **length 3**
- Can **merge 2 runs** in each merge step
- Second step: produce **2 sorted runs** of **length 6**
- Third step: produce **1 sorted run** of **length 12**

Cost Analysis (Phase 1)

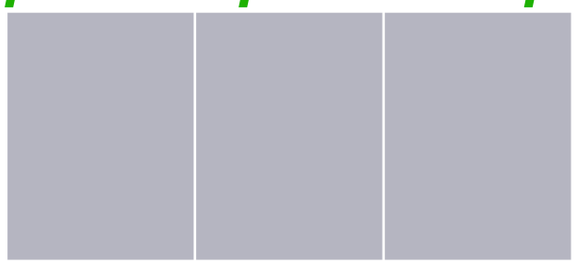
- Multiple **sorting passes**, we read and write data once in each
  - Cost per pass is **2 * N (N is number of pages)**
- **How many steps** must we make with B buffer pages?
  - **First step** produces runs of length B
  - **Second step** produces runs of length (B-1) * B
  - **Third step** produces runs of length (B-1) * (B-1) * B ...
  - Stop once **$(B-1)^{steps-1} * B \geq N$**, after **$1+Ceil(\log_{B-1}(N/B))$** steps

Slides by Immanuel Trummer, Cornell University

Join Phase 2: Overview

- (Have **sorted** both input tables by their join column)
- **Load first page** of both sorted tables into memory
- **Find matching tuples** and add to join result output
- **Load next page** for table with smallest last entry
- **Keep doing** until no pages left for one table

Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

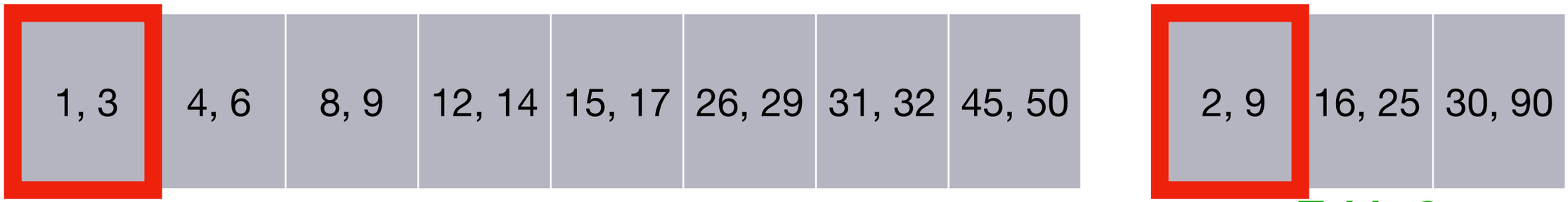
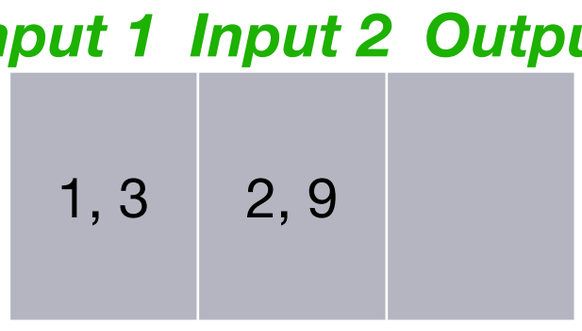


Table 1

Table 2

Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

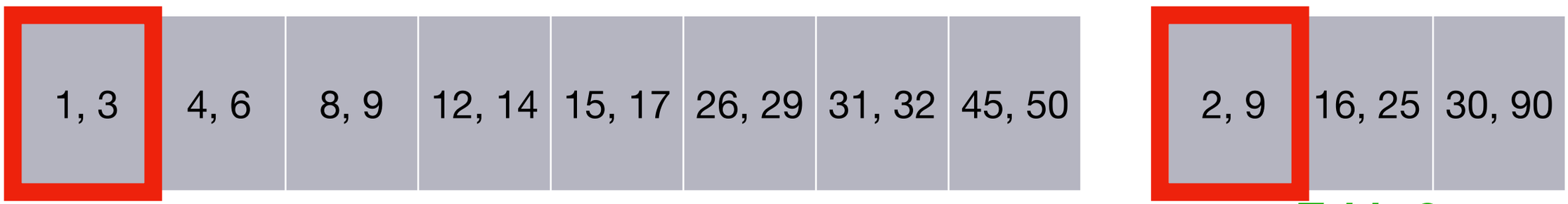
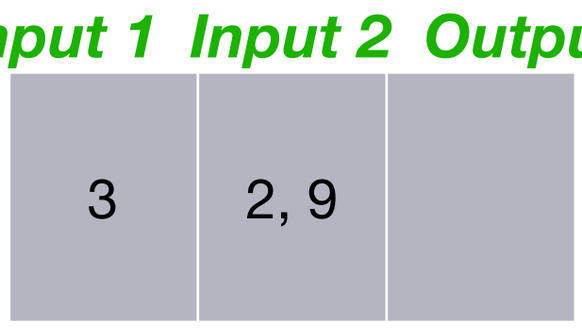


Table 1

Table 2

Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

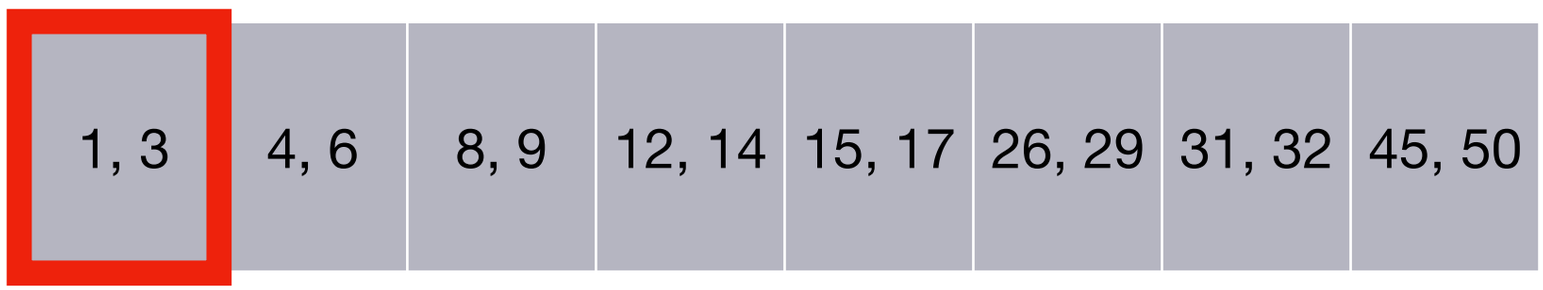


Table 1

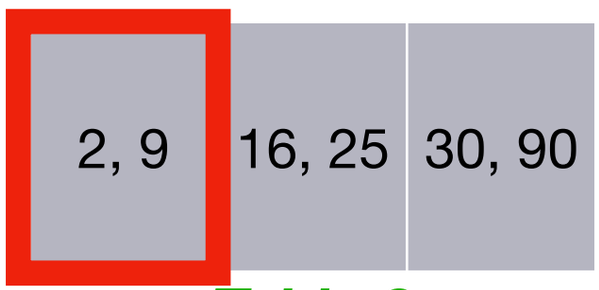
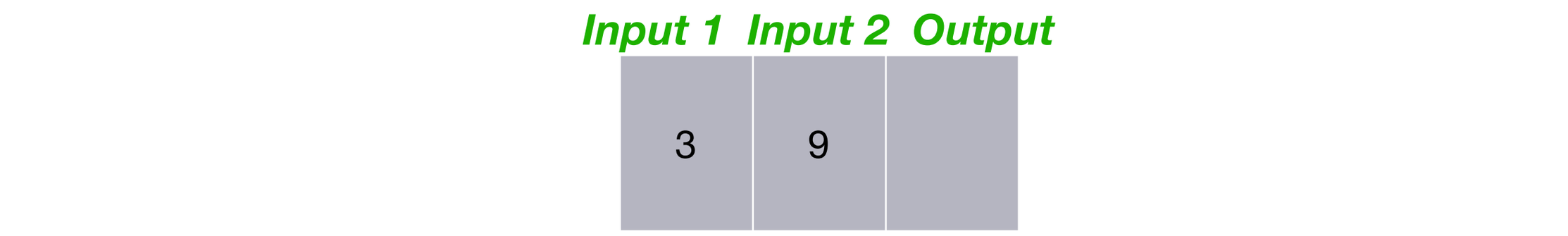
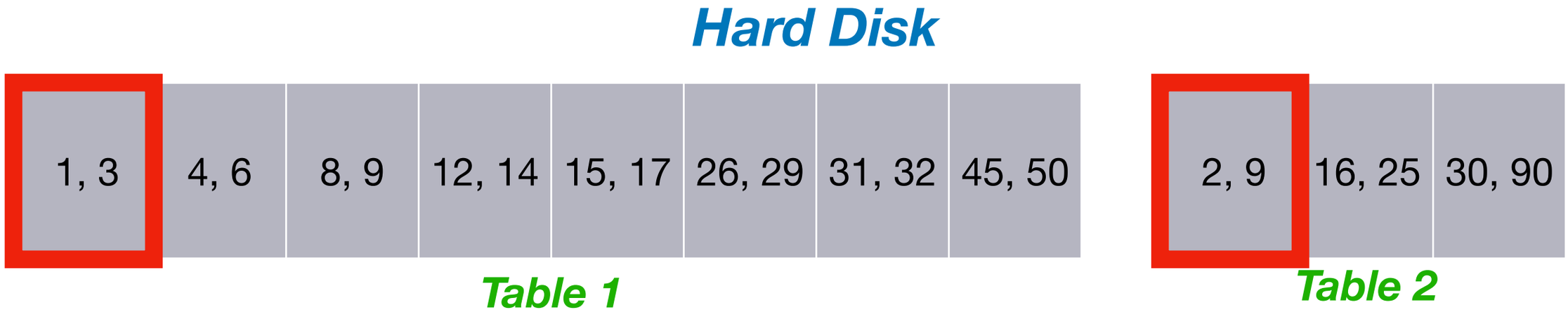


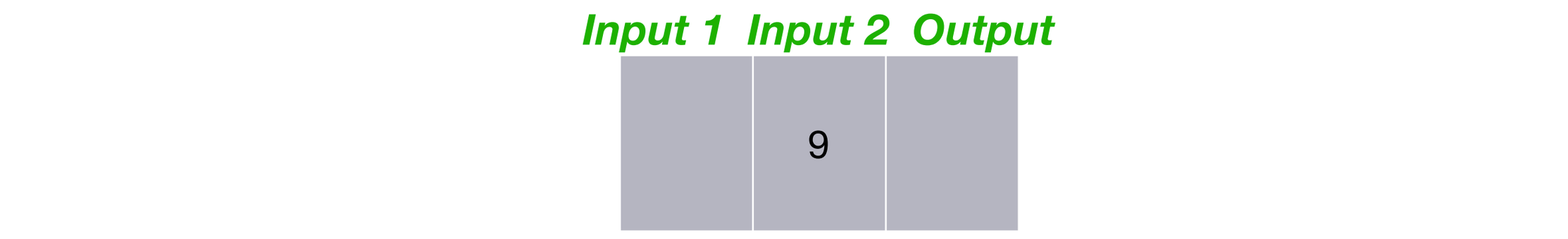
Table 2

Join Phase 2 Example

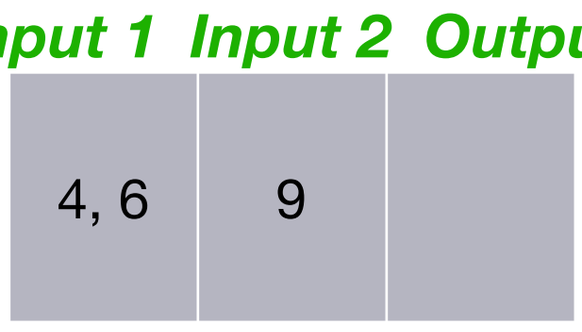




Join Phase 2 Example



Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

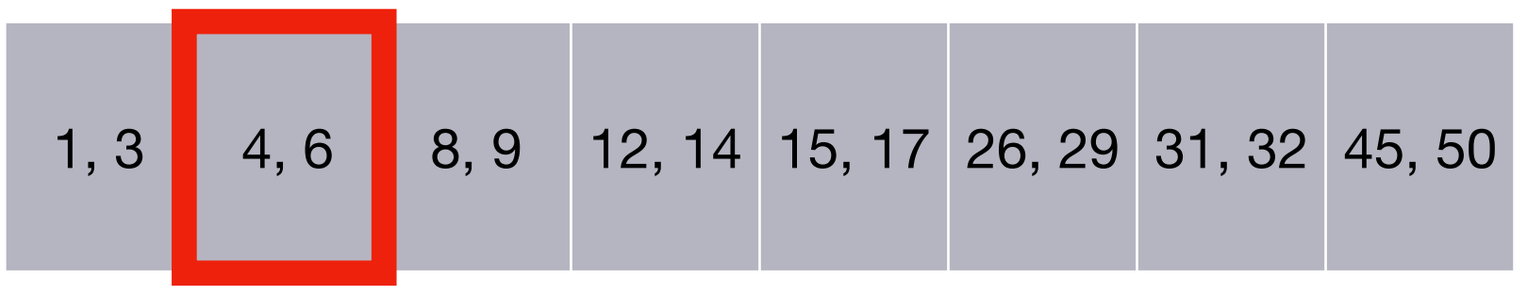


Table 1

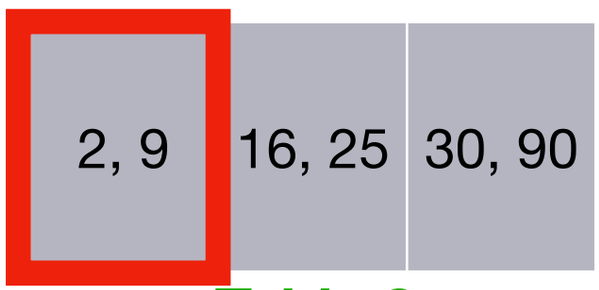
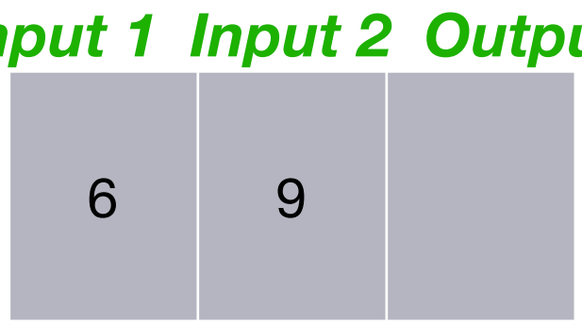


Table 2

Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

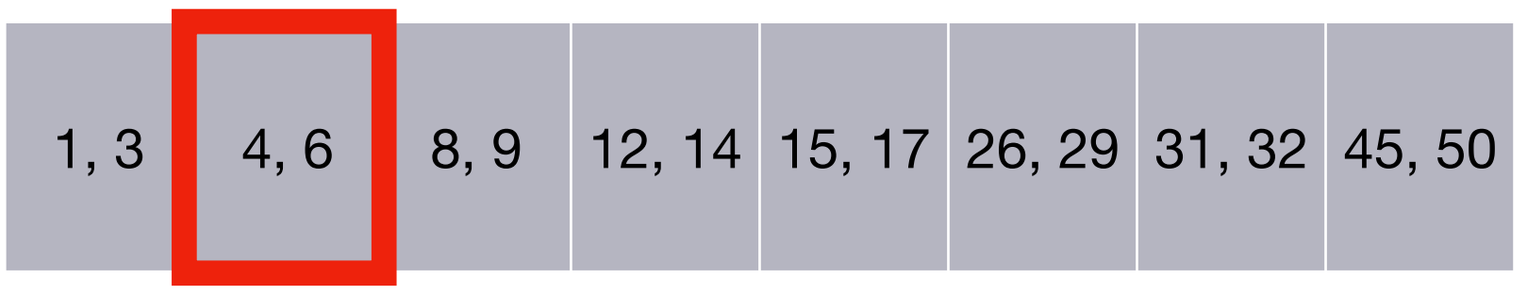


Table 1

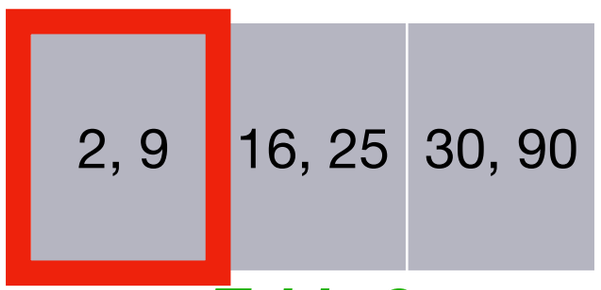
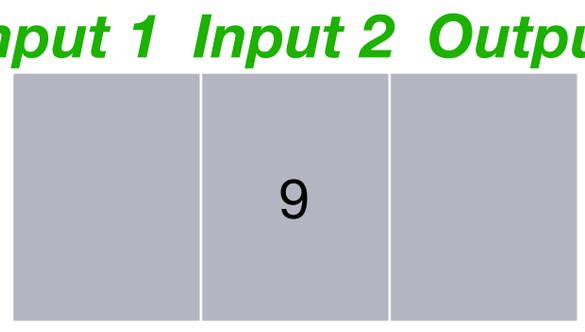
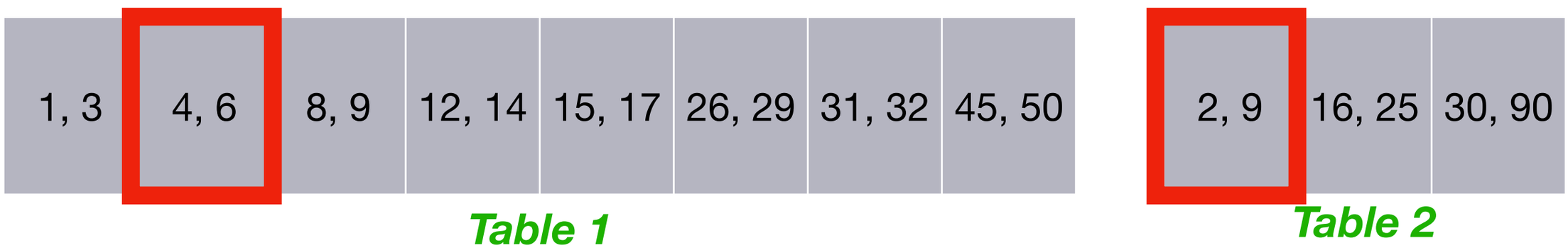


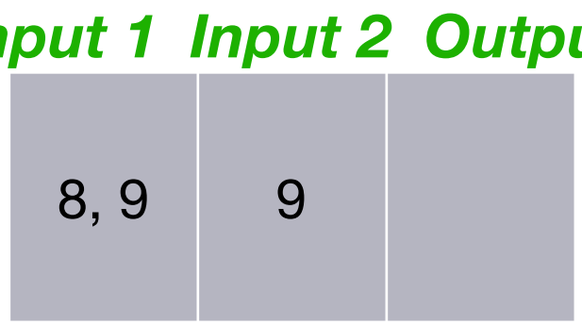
Table 2

Join Phase 2 Example





Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

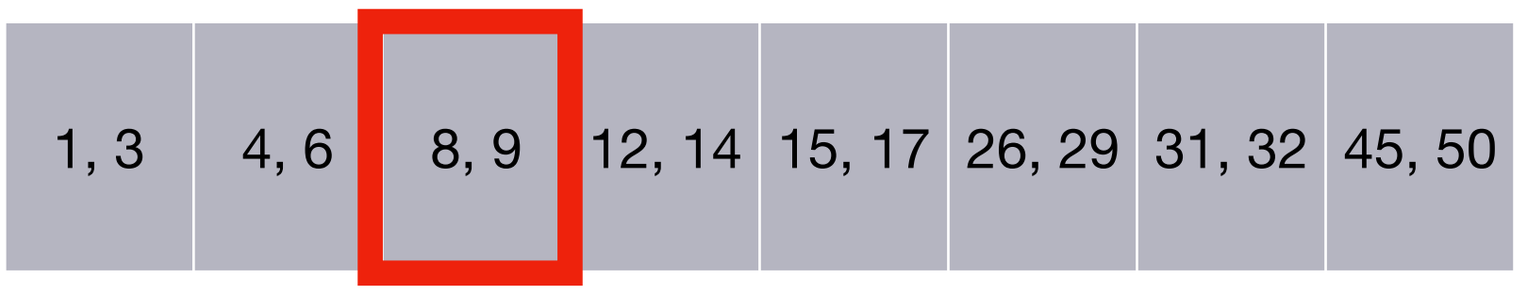


Table 1

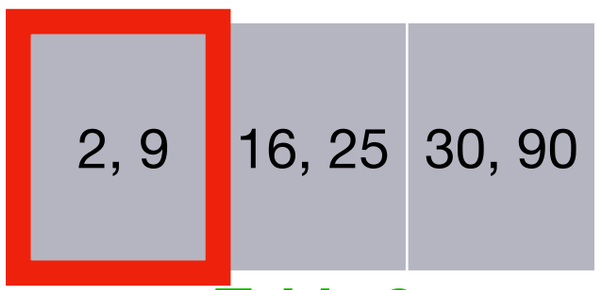
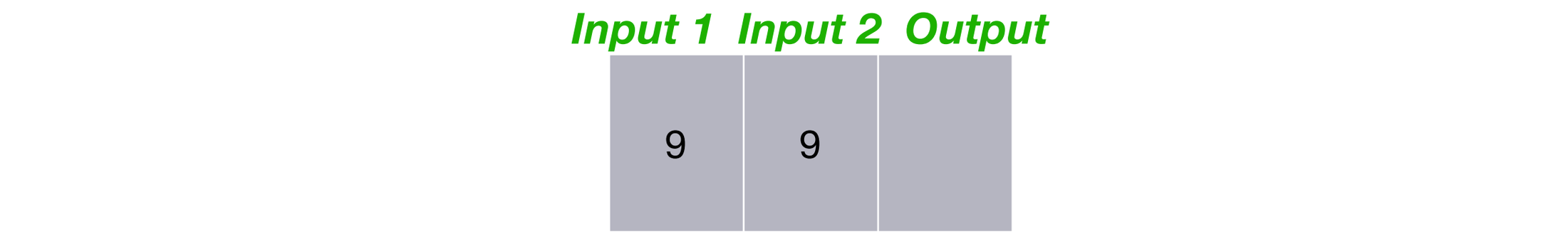


Table 2

Join Phase 2 Example



Join Phase 2 Example

Buffer Pool (3 Pages)

Hard Disk

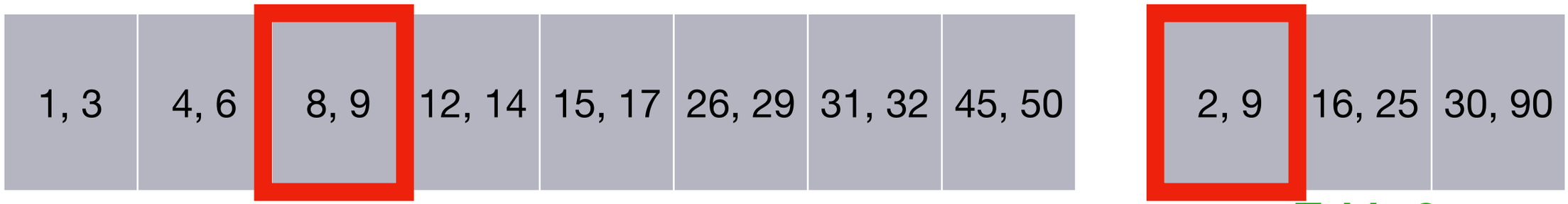
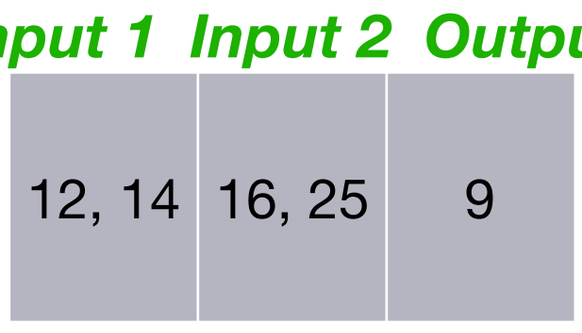


Table 1

Table 2

Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

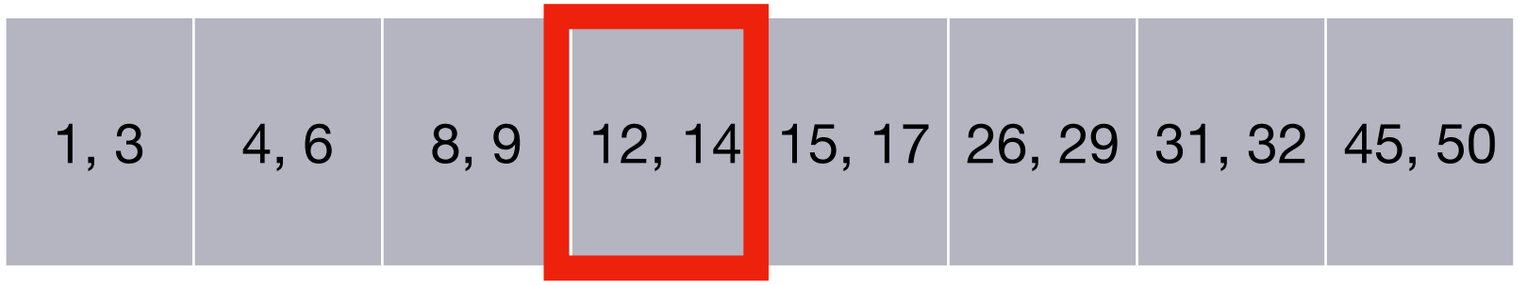


Table 1

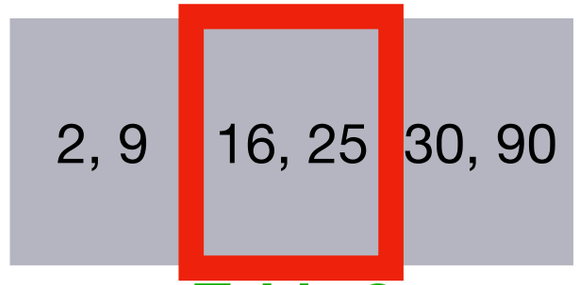
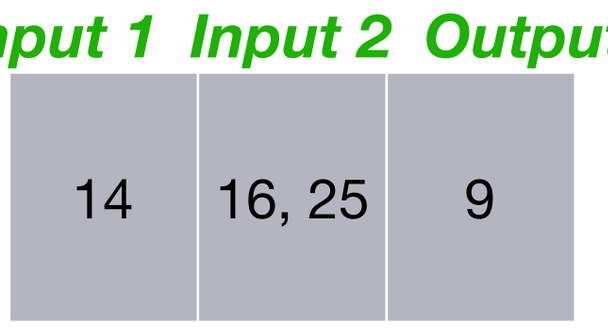


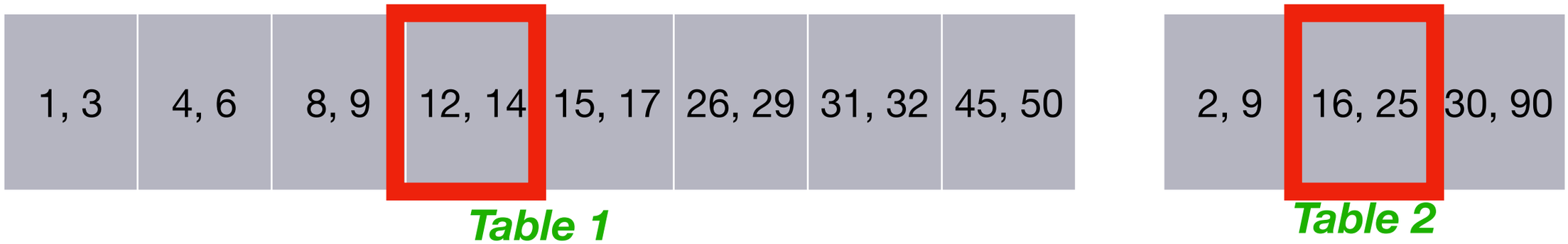
Table 2

Join Phase 2 Example

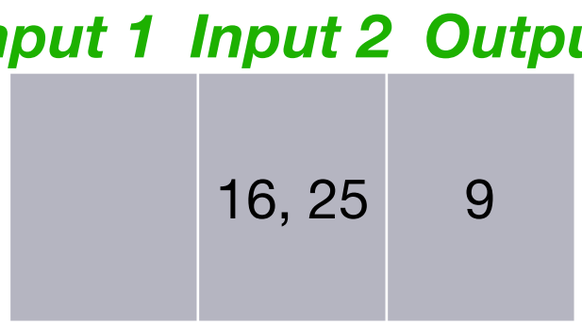


Buffer Pool (3 Pages)

Hard Disk



Join Phase 2 Example



Buffer Pool (3 Pages)

Hard Disk

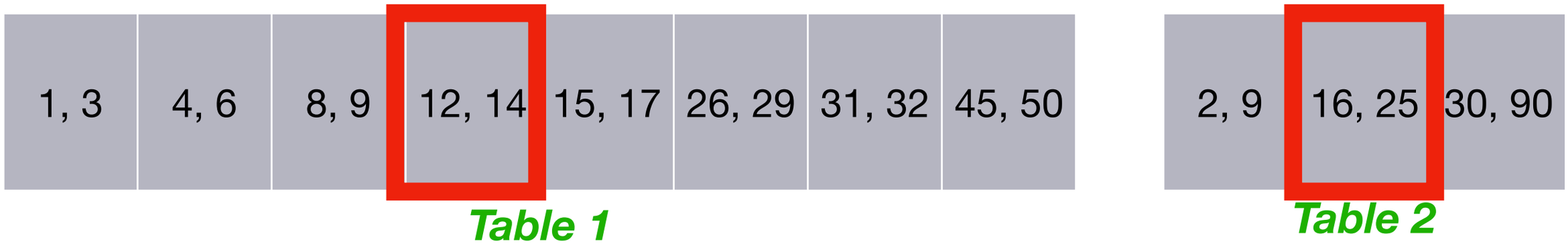


Table 1

Table 2

Join Phase 2 Example

Input 1 Input 2 Output

15, 17	16, 25	9

Buffer Pool (3 Pages)

Hard Disk

1, 3	4, 6	8, 9	12, 14	15, 17	26, 29	31, 32	45, 50

Table 1

2, 9	16, 25	30, 90

Table 2

Join Phase 2 Example

Input 1 Input 2 Output

17	16, 25	9

Buffer Pool (3 Pages)

Hard Disk

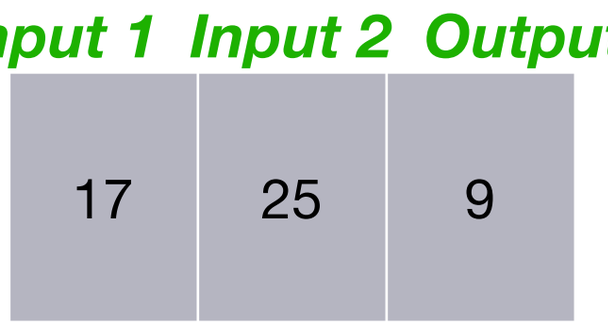
1, 3	4, 6	8, 9	12, 14	15, 17	26, 29	31, 32	45, 50

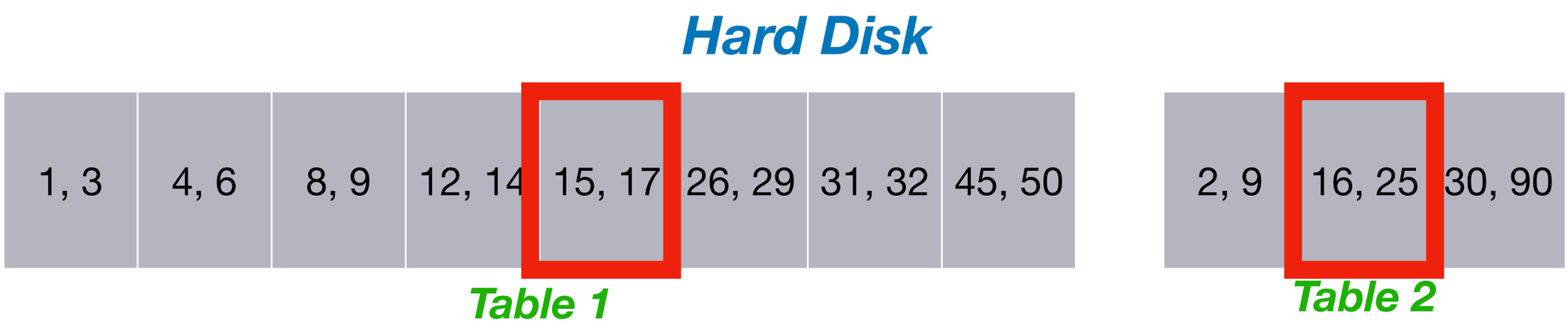
Table 1

2, 9	16, 25	30, 90

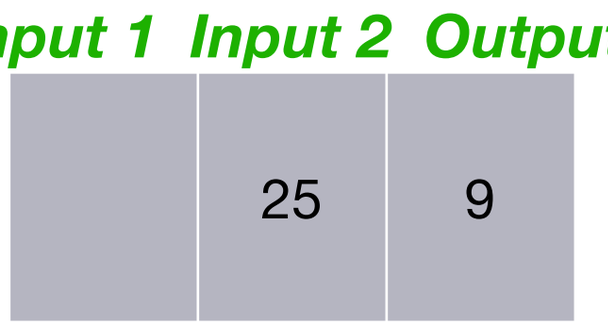
Table 2

Join Phase 2 Example



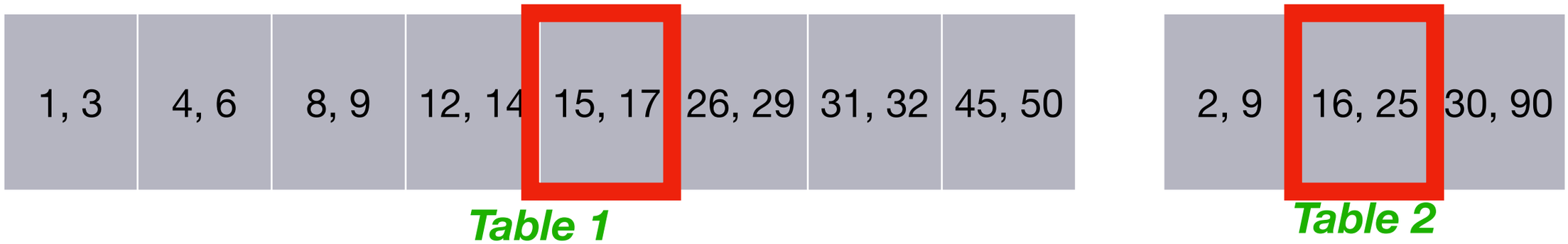


Join Phase 2 Example

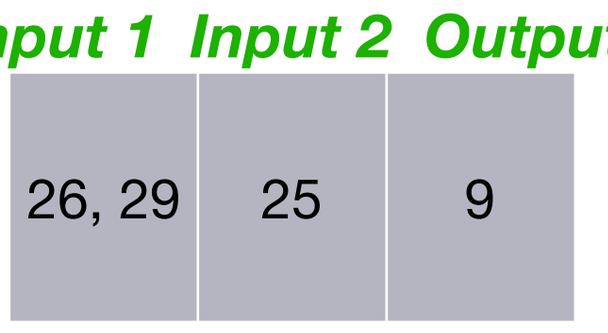


Buffer Pool (3 Pages)

Hard Disk

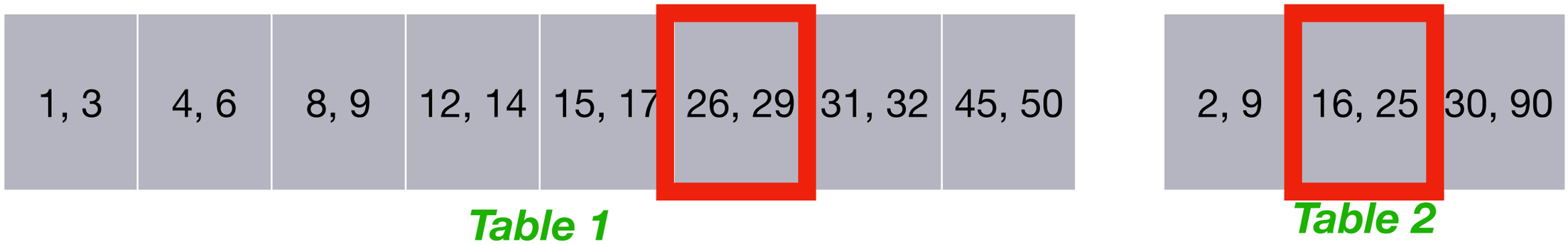


Join Phase 2 Example

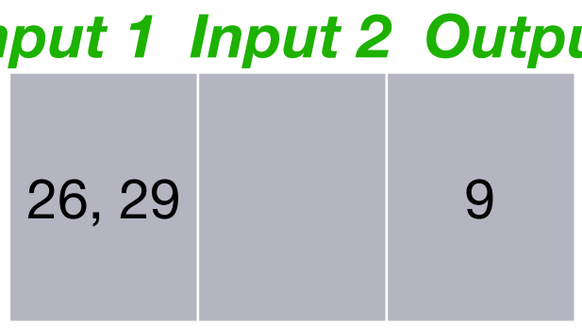


Buffer Pool (3 Pages)

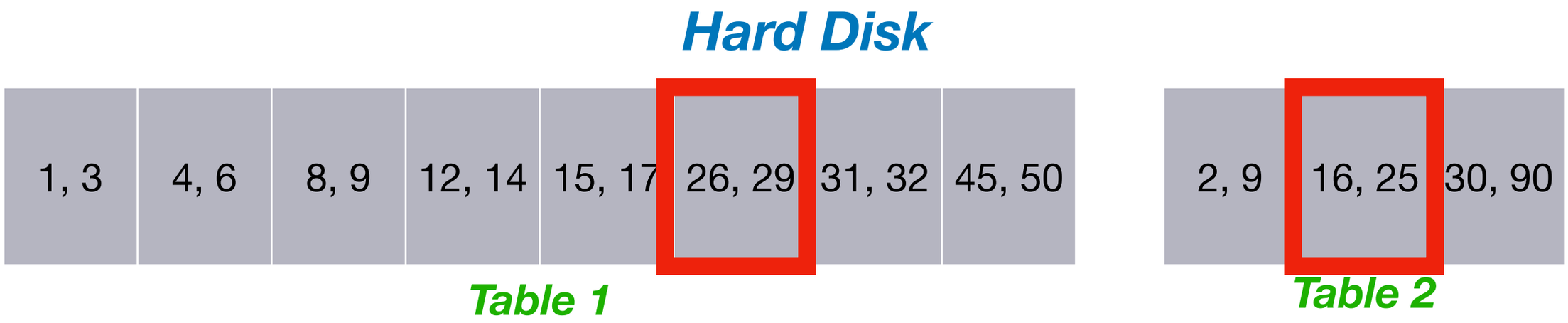
Hard Disk



Join Phase 2 Example



Buffer Pool (3 Pages)



Join Phase 2 Example

Input 1 Input 2 Output

26, 29	30, 90	9

Buffer Pool (3 Pages)

Hard Disk

1, 3	4, 6	8, 9	12, 14	15, 17	26, 29	31, 32	45, 50

Table 1

2, 9	16, 25	30, 90

Table 2

Join Phase 2 Example

Input 1 Input 2 Output

29	30, 90	9

Buffer Pool (3 Pages)

Hard Disk

1, 3	4, 6	8, 9	12, 14	15, 17	26, 29	31, 32	45, 50

Table 1

2, 9	16, 25	30, 90

Table 2

Handling Many Duplicates

- May have duplicates over **multiple pages**
  - Duplicate entry: **same value in join column**
- **Must revert** to first page with duplicate whenever we load new page from other table
- This makes the join more **expensive**

Cost Analysis (Phase 2)

- For now: assume all **duplicate entries** on same page
- Means that each input page is **only read once**
- Cost is proportional to **number of input pages**
  - I.e., Pages from **both** input tables

Total Join Cost

- Two input tables with M and N pages, B buffer pages
- First phase has cost
  - **$2*M*(1+Ceil(\log_{B-1}(M/B)))$** for sorting table 1
  - **$2*N*(1+Ceil(\log_{B-1}(N/B)))$** for sorting table 2
- Second phase has cost
  - **M+N** (we don't count cost for writing output!)

How Much Memory?

- First phase: try to exploit **all buffer pages**
  - More buffer means less merging passes!
- Second phase: only exploit **three buffer pages**
  - One for first input, one for second input, one output

How Much Memory?

- First phase: try to exploit **all buffer pages**
  - More buffer means less merging passes!
- Second phase: only exploit **three buffer pages**
  - One for first input, one for second input, one output

Seems Sub-Optimal!

Refined Sort-Merge Join

- Idea: can merge **more than two** sorted tables in phase 2
- Hence, **do not need to sort** tables completely in phase 1
- Means we can **save steps** (i.e., passes over the data)
- **First phase**: only sort data chunks that fit into memory
- **Second phase**: join all sorted chunks together (one step)

Refined Join Details

- Assume B buffer pages, tables with N and M pages
- **First phase**: load chunks of B pages, sort, write back
  - We now have (N+M)/B sorted chunks on disk
- **Second phase**: merge B-1 sorted chunks together
  - Can sort entries in-memory to find matches
- Cost is **2*(M+N)** (Phase 1) **+ 1 * (M+N)** (Phase 2)

How Much Memory?

- Again, B buffer pages, input sizes are M and N
- Have **(N+M)/B** sorted runs after first phase
- Need **B-1 ≥ (N+M)/B** to merge them in one step
- Rule of thumb if N>M: need **B ≥ 2*Sqrt(N)**

R-SMJ vs. Hash Join

	Hash Join	Refined Sort-Merge Join
Time	3 * Input Size	3 * Input Size
Memory	**> Sqrt(Smaller Table Size)**	> 2 * Sqrt(Larger Table Size)
Parallelization	**Advantage**	
Skew-Resistance		**Advantage**

Slides by Immanuel Trummer, Cornell University