CS186 Discussion 3

(Join Algorithms)

Matthew Deng

Join Algorithms

Cost Notation

- [R] = number of pages in Table R
- p_R = number of records per page of R
- |R| = number of records in R

$$|R| = p_R * [R]$$

Simple Nested Loop Join

```
for record r in R:
  for record s in S:
    if theta(r, s):
      add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + |R| * [S]$$

Simple Nested Loop Join

```
for record r in R:
  for record s in S:
    if theta(r, s):
      add join(r, s) to result
```

[R] = number of pages in Table R

p_R = number of records per page of R

$|R|$ = number of records in R

$$[R] + |R| * [S]$$

Simple Nested Loop Join

```
for record r in R:
  for record s in S:
    if theta(r, s):
      add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + |R| * [S]$$

Page Nested Loop Join

```
for page $p_r$ in R:
  for page $p_s$ in S:
    for record r in $p_r$:
      for record s in $p_s$:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] * [S]$$

Page Nested Loop Join

```
for page $p_r$ in R:
  for page $p_s$ in S:
    for record r in $p_r$:
      for record s in $p_s$:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] * [S]$$

Page Nested Loop Join

```
for page $p_r$ in R:
  for page $p_s$ in S:
    for record r in $p_r$:
      for record s in $p_s$:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] * [S]$$

Chunk Nested Loop Join

```
for chunk $c_r$ in R:
  for page $p_s$ in S:
    for record r in $c_r$:
      for record s in $p_s$:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] / (B - 2) * [S]$$

B-2 pages in each chunk

Chunk Nested Loop Join

```
for chunk $c_r$ in R:
  for page $p_s$ in S:
    for record r in $c_r$:
      for record s in $p_s$:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] / (B - 2) * [S]$$

B-2 pages in each chunk

Chunk Nested Loop Join

```
for chunk $c_r$ in R:
  for page $p_s$ in S:
    for record r in $c_r$:
      for record s in $p_s$:
        if theta(r, s):
          add join(r, s) to result
```

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$[R] + [R] / (B - 2) * [S]$$

B-2 pages in each chunk

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$
$$= 5 * ([R] + [S])$$

Sort Merge Join

1. External Sort R
2. External Sort S
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$4 * ([R] + [S]) + [R] + [S]$$

$$= 5 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R
   (pass 0 only)
2. Internal Sort S
   (pass 0 only)
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R

   (pass 0 only)

2. Internal Sort S

   (pass 0 only)

3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R

   (pass 0 only)

2. Internal Sort S

   (pass 0 only)

3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$

$$= 3 * ([R] + [S])$$

Optimized Sort Merge Join

1. Internal Sort R
   (pass 0 only)
2. Internal Sort S
   (pass 0 only)
3. Merge R and S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$

$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Hash Join

1. Partition R
2. Partition S
3. Build Hash Table for R
4. Stream and Probe S

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

$$2 * ([R] + [S]) + [R] + [S]$$
$$= 3 * ([R] + [S])$$

Join Algorithms Worksheet #1

Join Algorithms Exercises

1. Describe when you would want to use a chunk nested loop join, a sort-merge join and a hash join:

Join Algorithms Exercises

1. Describe when you would want to use a chunk nested loop join, a sort-merge join and a hash join:

Chunk Nested Loop Join:

- Cartesian Product
- Non-equality Predicate

Sort Merge Join:

- Sorting
- Skewed Data
- Limited Memory

Hash Join:

- Hashing
- Uneven Relations
- Hybrid Hashing

Join Algorithms Worksheet #2

[R] = number of pages in Table R
p_R = number of records per page of R
|R| = number of records in R

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables [R] and [S] where [R] is 100 pages and [S] is 50 pages. [R] holds 100 tuples per page and [S] holds 50 tuples per page.

How many disk reads are needed to perform a Simple Nested Loops join?

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables [R] and [S] where [R] is 100 pages and [S] is 50 pages. [R] holds 100 tuples per page and [S] holds 50 tuples per page.

How many disk reads are needed to perform a Simple Nested Loops join?

Using S as the outer relation yields the lowest I/O count.
p_S*[R]*[S] + [S] = 50*100*50 + 50 = 250050 I/O's

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables [R] and [S] where [R] is 100 pages and [S] is 50 pages. [R] holds 100 tuples per page and [S] holds 50 tuples per page.

How about a Sort Merge Join? (Utilize the optimized version)

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables [R] and [S] where [R] is 100 pages and [S] is 50 pages. [R] holds 100 tuples per page and [S] holds 50 tuples per page.

How about a Sort Merge Join? (Utilize the optimized version)

3([R] + [S]) = 3*150 = 450 I/O’s

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables [R] and [S] where [R] is 100 pages and [S] is 50 pages. [R] holds 100 tuples per page and [S] holds 50 tuples per page.

How about a Hash Join? (Assume no recursive partitioning and ignore output costs)

Join Algorithms Exercises

2. We have 15 pages of memory, and we want to join two tables [R] and [S] where [R] is 100 pages and [S] is 50 pages. [R] holds 100 tuples per page and [S] holds 50 tuples per page.

How about a Hash Join? (Assume no recursive partitioning and ignore output costs)

Partitioning Phase: 2([R]+[S])

Matching Phase: [R]+[S]

Total = 3([R] + [S]) = 3* 150 = 450 I/O's