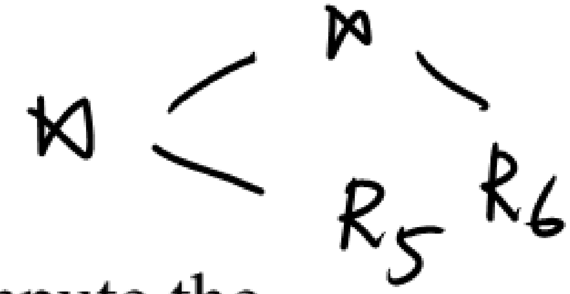
Dynamic Programming

- Idea: for each subset of {R1, …, Rn}, compute the best plan for that subset

Drinker

$Beers \bowtie Sells \bowtie$ ~~Bar~~

- In increasing order of set cardinality:
  - K=1 – Step 1: for {R1}, {R2}, …, {Rn} {Beers}, {Sells}, {Drinker}
  - K=2 – Step 2: for {R1,R2}, {R1,R3}, …, {Rn-1, Rn} $\{B \bowtie S\}, \{S \bowtie D\}$
  - – …
  - K=n – Step n: for {R1, …, Rn} $B \bowtie S \bowtie D$
- It is a bottom-up strategy
- A subset of {R1, …, Rn} is also called a *subquery*



Dynamic Programming

- For each subquery Q ⊆ {R1, …, Rn} compute the following:

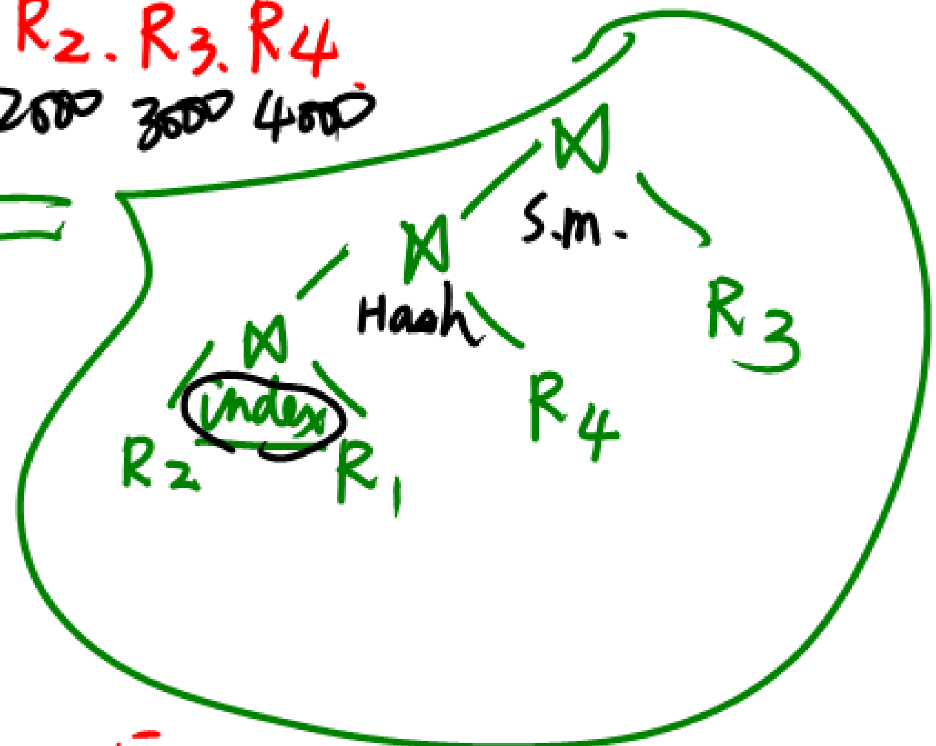
R1 . R2 . R3. R4

1000 2000 3000 4000

① – Size(Q) = ?

② – A best plan for Q: Plan(Q) =

③ – The cost of that plan: Cost(Q)



?

for subsequnt query operations

Dynamic Programming

- **Step 1**: For each {Ri} do:
  - Size({Ri}) = B(Ri) # blocks.
  - Plan({Ri}) = Ri
  - Cost({Ri}) = (cost of scanning Ri)

$R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5$

join is binary

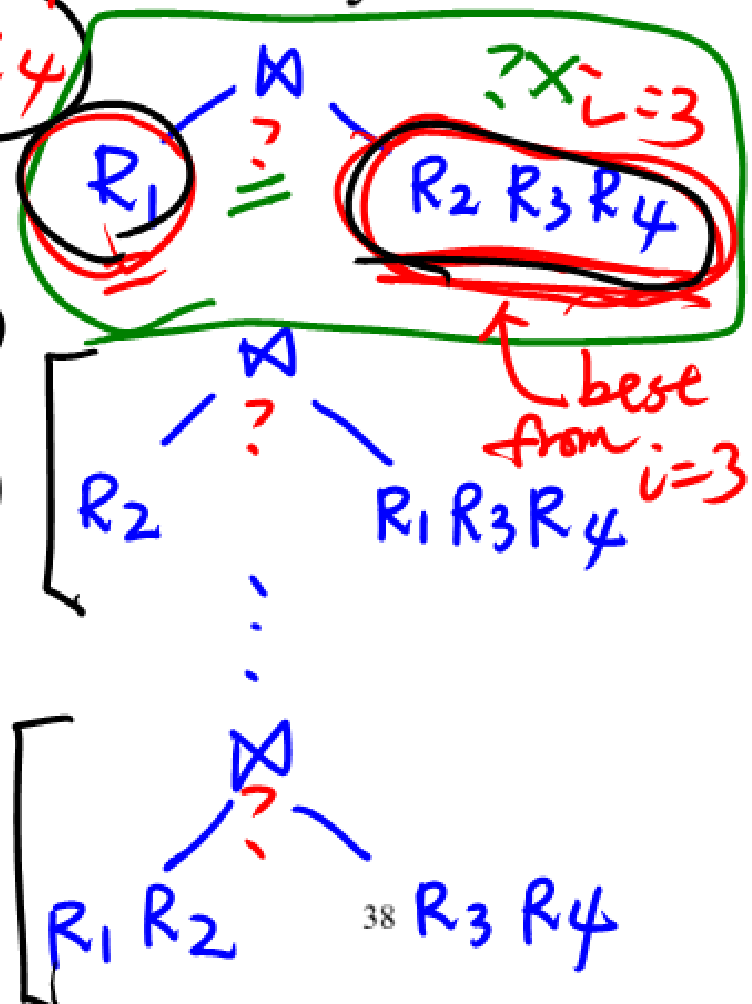
Dynamic Programming

$i=4$

- **Step i**: For each Q ⊆ {R1, …, Rn} of cardinality i do: $= R_1 . R_2 . R_3 . R_4$
  - Compute Size(Q) (later…)
  - For every pair of subqueries Q', Q'' s.t. Q = Q' ∪ Q'' compute cost(Plan(Q') ⋈ Plan(Q''))
  - Cost(Q) = the smallest such cost
  - Plan(Q) = the corresponding plan

$i=5$

best, for $i = 4$



$i = n$

Dynamic Programming

- Return Plan({R1, …, Rn})

this is why we need to remember cost of subq.

Dynamic Programming

To illustrate, we will make the following simplifications:

- Cost($P1 \bowtie P2$) = Cost(P1) + $\boxed{\text{Cost(P2)}}$ + size(intermediate result)

  R_1 $R_2R_3R_4$

  e.g. $P_2 = R_2R_3R_4$

- Intermediate results:

  size(P_1) + size(P_2)

  - If P1 = a join, then the size of the intermediate result is size(P1), otherwise the size is 0

    if P_1. P_2 are joins.

    $P_1 = R_1$ ✗

  - Similarly for P2
- Cost of a scan = 0

Dynamic Programming

- Example:
- Cost(R1 ⋈ R2) = 0 (no intermediate results)
- Cost((R2 ⋈ R1) ⋈ R7)

  = Cost(R2 ⋈ R1) + Cost(R7) + size(R2 ⋈ R1)

  = size(R2 ⋈ R1)

① $$\text{cost}(R_1 \bowtie R_2) = \underbrace{\text{cost}(R_1)}_{=0} + \underbrace{\text{cost}(R_2)}_{=0} + \cancel{\text{size}(R_1)} = 0$$

$$+ \cancel{\text{size}(R_2)} = 0$$

$$= 0$$

② $$\text{cost}\big(\overbrace{(R_1 \bowtie R_2)}^{P_1} \bowtie \overbrace{R_3}^{P_2}\big) = \overbrace{C(R_1 \bowtie R_2)}^{0} + C(R_3)^{=0}$$

$$+ S(R_1 \bowtie R_2) + \cancel{S(R_3)} = 0$$

$$= \boxed{S(R_1 \bowtie R_2)}$$

③ $$C\big((R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)\big) = S(R_1 \bowtie R_2) + S(R_3 \bowtie R_4)$$

④ $$C\big((R_1 \bowtie R_2 \bowtie R_3) \bowtie R_4\big)$$

$$= C(R_1 \bowtie R_2 \bowtie R_3) + S(R_1 \bowtie R_2 \bowtie R_3)$$

Dynamic Programming

- Relations: R, S, T, U
- Number of tuples: 2000, 5000, 3000, 1000
- Size estimation: T(A ⋈ B) = 0.01*T(A)*T(B)

A : 1000 tuples

B : 50 tuples

$A \bowtie B = ?\ 0.01 \times 1000 \times 50 = 500$

$R \bowtie S \bowtie T \bowtie U$?

$i=2$ rel

$C^4_2 = \frac{4 \cdot 3}{2 \cdot 1} = 6$

$i=3$

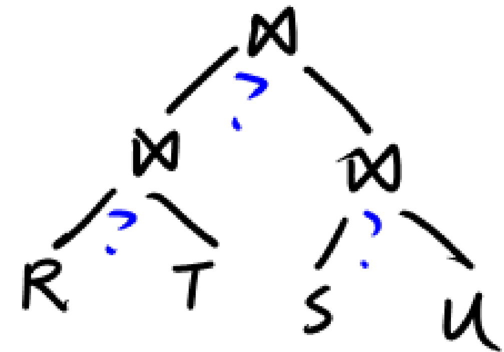
$C^4_3 = 4$

$i=4$

$C^4_4 = 1$

Subquery	Size	Cost	Plan
RS 1	0.1 × R × S = 100k	0	$R \bowtie S$
RT 2			
RU 3			
ST 4			
SU 5			
TU 6			
RST			
RSU			
RTU			
STU			
RSTU			$(RT)\bowtie(SU)$ =

Bushy



Subquery	Size	Cost	Plan
RS	100k	0	RS
RT	60k	0	RT
RU	20k	0	RU
ST	150k	0	ST
SU	50k	0	SU
TU	30k	0	TU
RST	3M	60k	(RT)S
RSU	1M	20k	(RU)S
RTU	0.6M	20k	(RU)T
STU	1.5M	30k	(TU)S
RSTU	30M	60k+50k=110k	(RT)(SU)

i=2

i=3

$C_3^4 = 4$

i=4

RSTU

$R \bowtie S \bowtie T$ $P = ?$

How to assemble?

thm $\bowtie$

P_1: $(RS) \bowtie T$, RS: i=2

P_2: $(RT) \bowtie S$, RT: i=2, 60k

P_3: $(ST) \bowtie R$, ST: i=2, 150k

size: $100k \times 3k \times 0.01 = 3M$

cost: $C(RS \bowtie T) = S(RS) = 100k$

$\underline{i=4}$ (FINALLY!) RSTU

$4 = \cancel{0+4}$

$\underline{1+3}$

$\underline{2+2}$

$\cancel{3+1}$

$\cancel{4+0}$

our cost model is insensitive to le/right

L.D.

$i=3!$

$\underline{1+3}$

$\underline{\underline{C^4_1 = 4}}$

$U \bowtie RST$

$T \bowtie \ldots$

$S \bowtie \ldots$

$U \bowtie \ldots$

$\frac{C^4_2}{2} = 3$

Bushy

$i=2!$

2+2

$RS \bowtie \ldots$

$\boxed{RT \bowtie SU}$

$RU \bowtie \ldots$

!!

$\Rightarrow$

Dynamic Programming

- Summary: computes optimal plans for subqueries:
  - Step 1: {R1}, {R2}, …, {Rn}
  - Step 2: {R1, R2}, {R1, R3}, …, {Rn-1, Rn}
  - …
  - Step n: {R1, …, Rn}
- We used naïve size/cost estimations
- In practice:
  - more realistic size/cost estimations (next time)
  - heuristics for Reducing the Search Space
    - Restrict to left linear trees
    - Restrict to trees "without cartesian product": R(A,B), S(B,C), T(C,D) (R join T) join S has a cartesian product

Completing Physical Query Plan

How can get order for free?

order

$R \overset{\bowtie}{s.m} S$ — T

index scan

Completing the Physical Query Plan

- Choose algorithm to implement each operator
  - Need to account for more than cost:
    - How much memory do we have ?
    - Are the input operand(s) sorted ?
- Decide for each intermediate result:
  - To materialize
  - To pipeline

Additional detail

between operations

sorted

$R \overset{\bowtie}{index} S$ — $\bowtie$ — T

Hash join

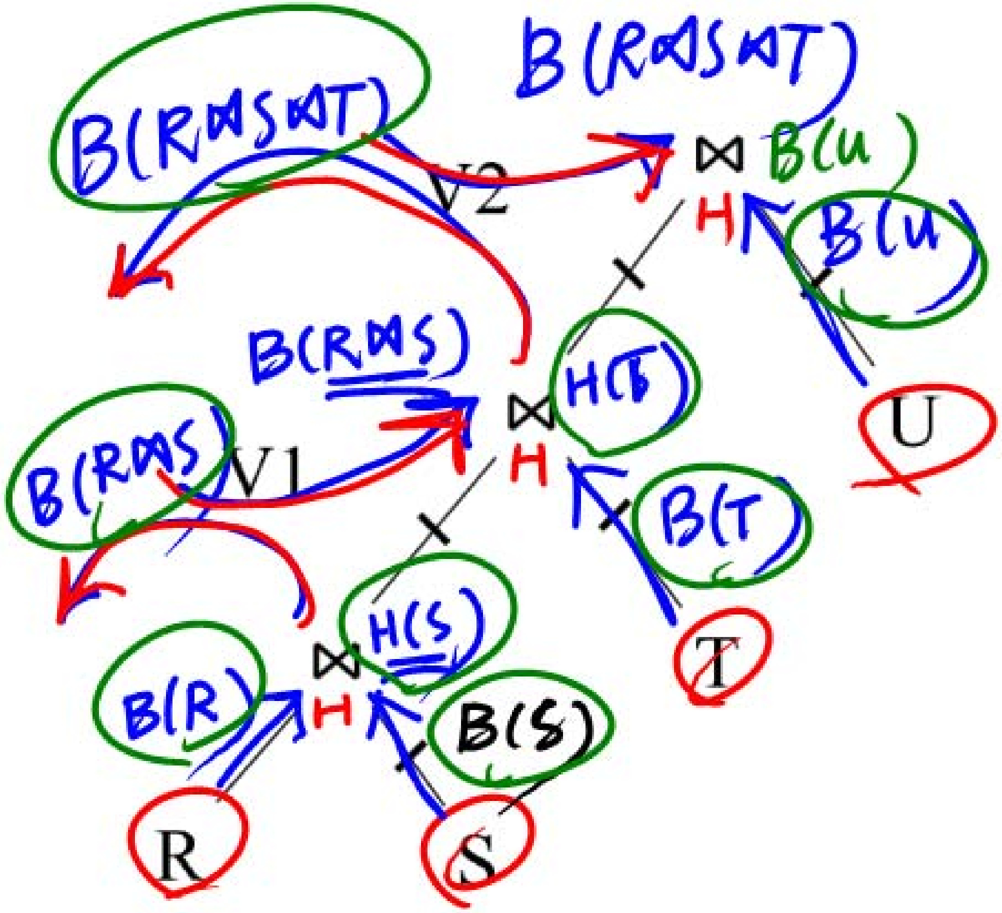
Join | index join ($R \bowtie_a S$) | s.m. join

req: | S.a has index | X

mem: | 1 block (R) X | $B(R)+B(S) \le M^2$

cost: $B(R) + T(R) \times \frac{B(S)}{V(S,a)}$ | $3B(R) + 3B(S)$

Materialize Intermediate Results Between Operators



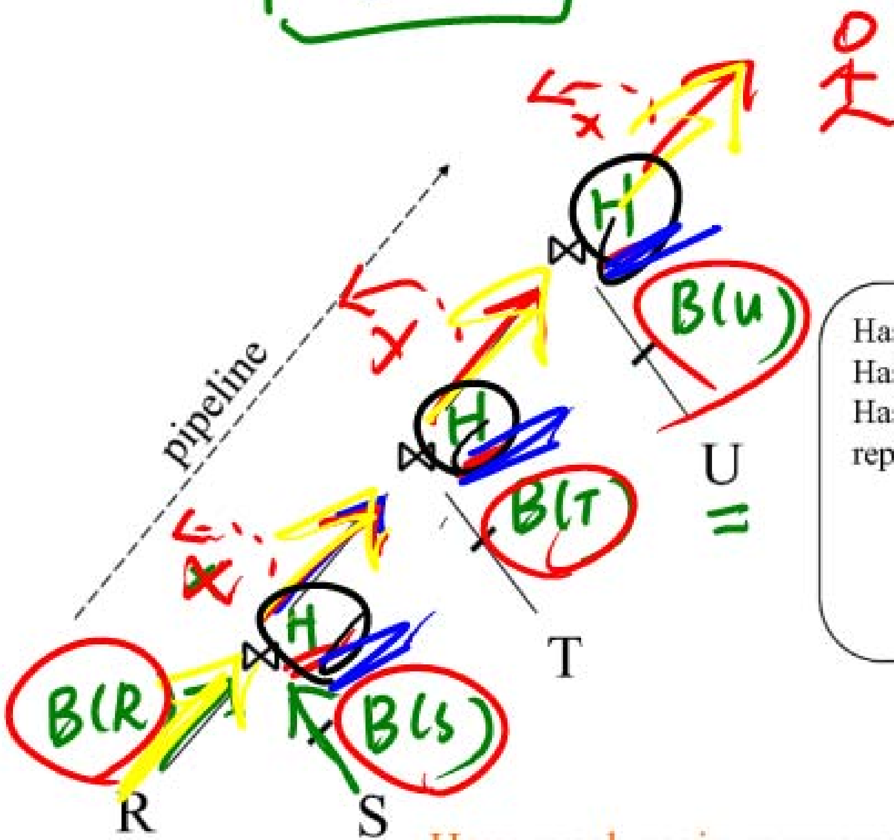

```
HashTable ← S
repeat    read(R, x)
          y ← join(HashTable, x)
          write(V1, y)

HashTable ← T
repeat    read(V1, y)
          z ← join(HashTable, y)
          write(V2, z)

HashTable ← U
repeat    read(V2, z)
          u ← join(HashTable, z)
          write(Answer, u)
```

Materialize Intermediate Results Between Operators

Given B(R), B(S), B(T), B(U)

- What is the total cost of the plan ?
  - Cost = $B(R) + B(S) + B(T) + B(U) + 2B(R \bowtie S) + 2B(R \bowtie S \bowtie T)$
- How much main memory do we need ?
  - M = $\max(B(S), B(T), B(U))$

Pipeline Between Operators




```
HashTable1 ← S
HashTable2 ← T
HashTable3 ← U
repeat    read(R, x)
          y ← join(HashTable1, x)
          z ← join(HashTable2, y)
          u ← join(HashTable3, z)
          write(Answer, u)
```

How much main memory do we need ? M =

↓ cost = B(R) + B(S) + B(T) + B(U) +

↑ Mem = Sum of B(S). B(T). B(U)

Pipeline Between Operators

Given B(R), B(S), B(T), B(U)

- What is the total cost of the plan ?
  - Cost =
- How much main memory do we need ?
  - M =

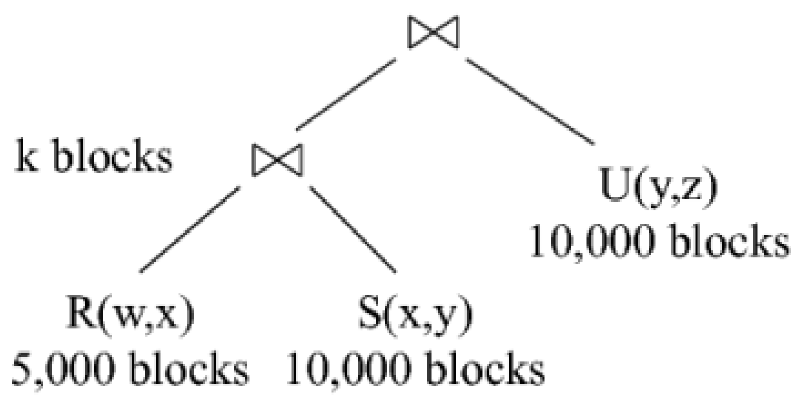
Completing the Physical Query Plan

- Choose algorithm to implement each operator
  - Need to account for more than cost:
    - How much memory do we have ?
    - Are the input operand(s) sorted ?
- Decide for each intermediate result:
  - To materialize
  - To pipeline

Skip

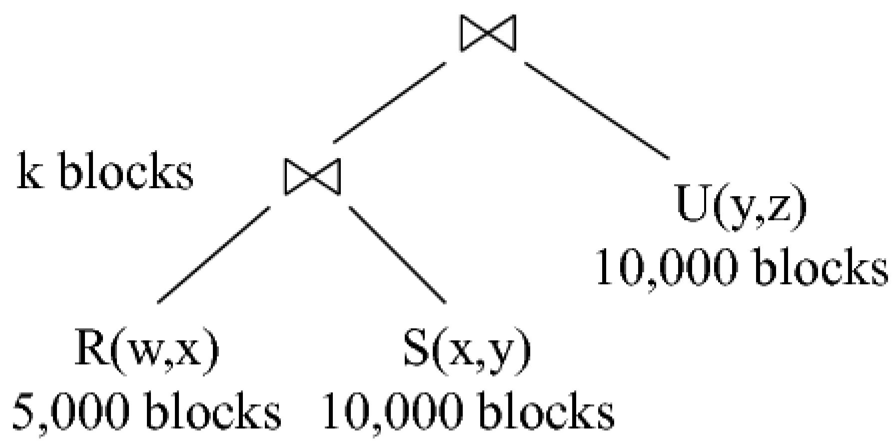
Example

- Logical plan is:



- Main memory M = 101 buffers

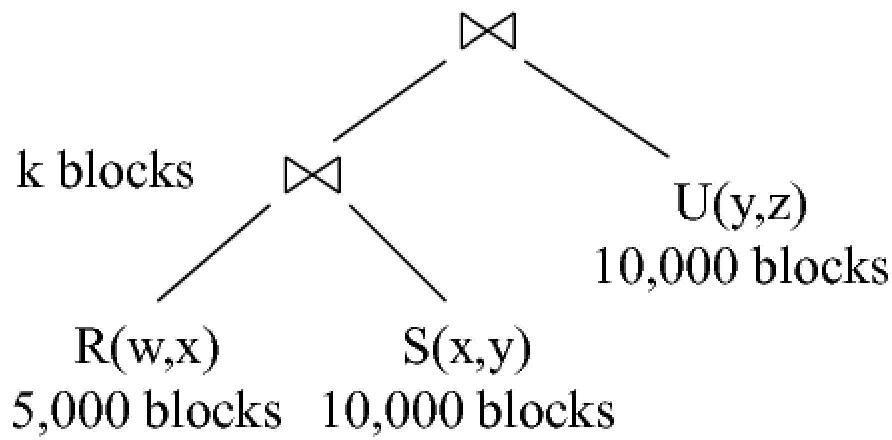
Example



Naïve evaluation:

- 2 partitioned hash-joins
- Cost 3B(R) + 3B(S) + 4k + 3B(U) = 75000 + 4k

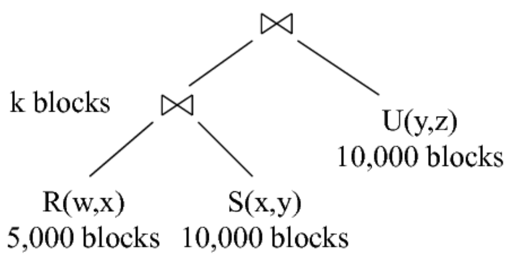
Example



Smarter:

- Step 1: hash R on x into 100 buckets, each of 50 blocks; to disk
- Step 2: hash S on x into 100 buckets; to disk
- Step 3: read each Ri in memory (50 buffer) join with Si (1 buffer); hash result on y into 50 buckets (50 buffers) -- here we *pipeline*
- Cost so far: 3B(R) + 3B(S)

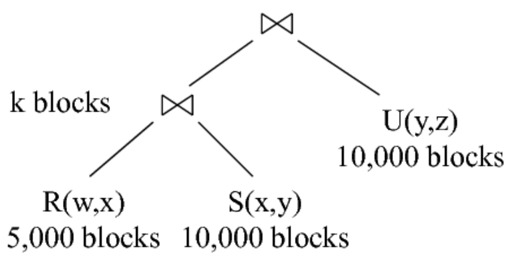
Example



Continuing:

- How large are the 50 buckets on y ? Answer: k/50.
- If k <= 50 then keep all 50 buckets in Step 3 in memory, then:
- Step 4: read U from disk, hash on y and join with memory
- Total cost: 3B(R) + 3B(S) + B(U) = 55,000

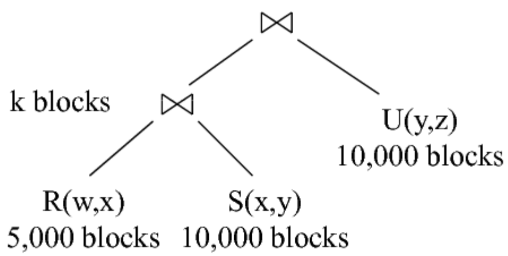
Example



Continuing:

- If 50 < k <= 5000 then send the 50 buckets in Step 3 to disk
  - Each bucket has size k/50 <= 100
- Step 4: partition U into 50 buckets
- Step 5: read each partition and join in memory
- Total cost: 3B(R) + 3B(S) + 2k + 3B(U) = 75,000 + 2k

Example



Continuing:

- If k > 5000 then materialize instead of pipeline
- 2 partitioned hash-joins
- Cost 3B(R) + 3B(S) + 4k + 3B(U) = 75000 + 4k

Example

Summary:

- If $k <= 50$, cost = 55,000
- If $50 < k <= 5000$, cost = 75,000 + 2k
- If $k > 5000$, cost = 75,000 + 4k

Estimating Sizes

$0.01 \times S(R) \times S(S)$

Estimating Sizes

- Need size in order to estimate cost
- Example:
  - Cost of partitioned hash-join E1 $\bowtie$ E2 is 3B(E1) + 3B(E2)
  - B(E1) = T(E1)/ block size
  - B(E2) = T(E2)/ block size
  - So, we need to estimate T(E1), T(E2)

Estimating Sizes

Estimating the size of a projection

- Easy: $T(\Pi_L(R)) = T(R)$
- This is because a projection doesn't eliminate duplicates

$R: 1000$

$\Pi_{dept}(R) = 1000$

$\approx 4 V(R, dept)$

Estimating Sizes

Estimating the size of a selection = $\underline{1000}$ tuple. $V(R,A)=4$

- $S = \sigma_{A=c}(R)$ $\sigma_{dept="cs"}(Stud)$ {"cs", "EE", "bio", "ME"}
  - T(S) can be anything from 0 to T(R) – V(R,A) + 1
  - Mean value: T(S) = T(R)/V(R,A) $= \frac{1000}{4} = 250$
- $S = \sigma_{A<c}(R)$ age<38
  - T(S) can be anything from 0 to T(R)
  - Heuristics: T(S) = T(R)/3

$1000 = \frac{1}{3} \times 1000$

The Lady's magic #. Selinger