CS 50: Software Design and Implementation

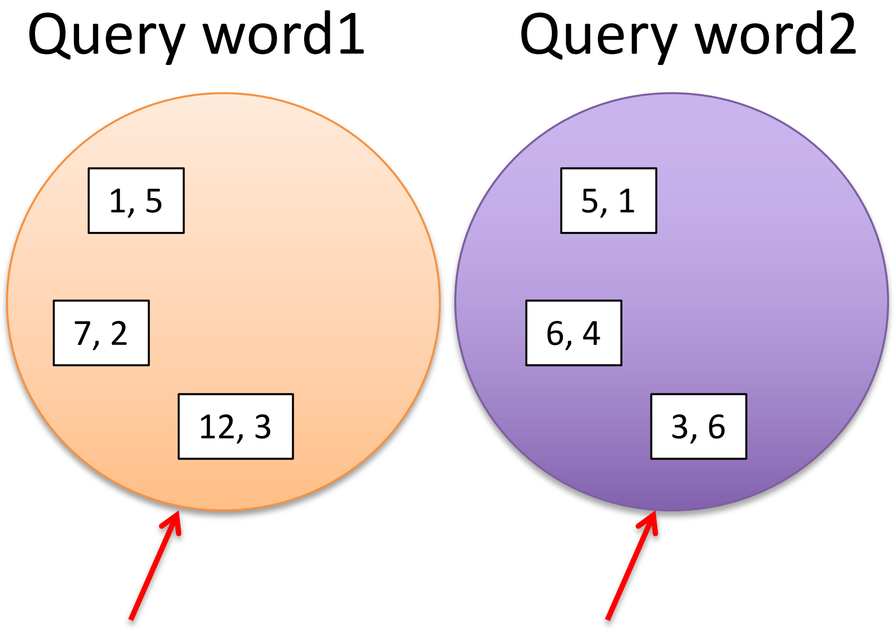
Querier design

Agenda

1. Union and intersect
2. Math operator precedence
3. Query operator precedence
4. Activity

Users enter query words, and the querier must implement AND and OR operations

Note: currently no pages contain both words



DocID, count of pages containing dartmouth

Example:
Doc1 contains word1 5 times

DocID, count of pages containing algorithm

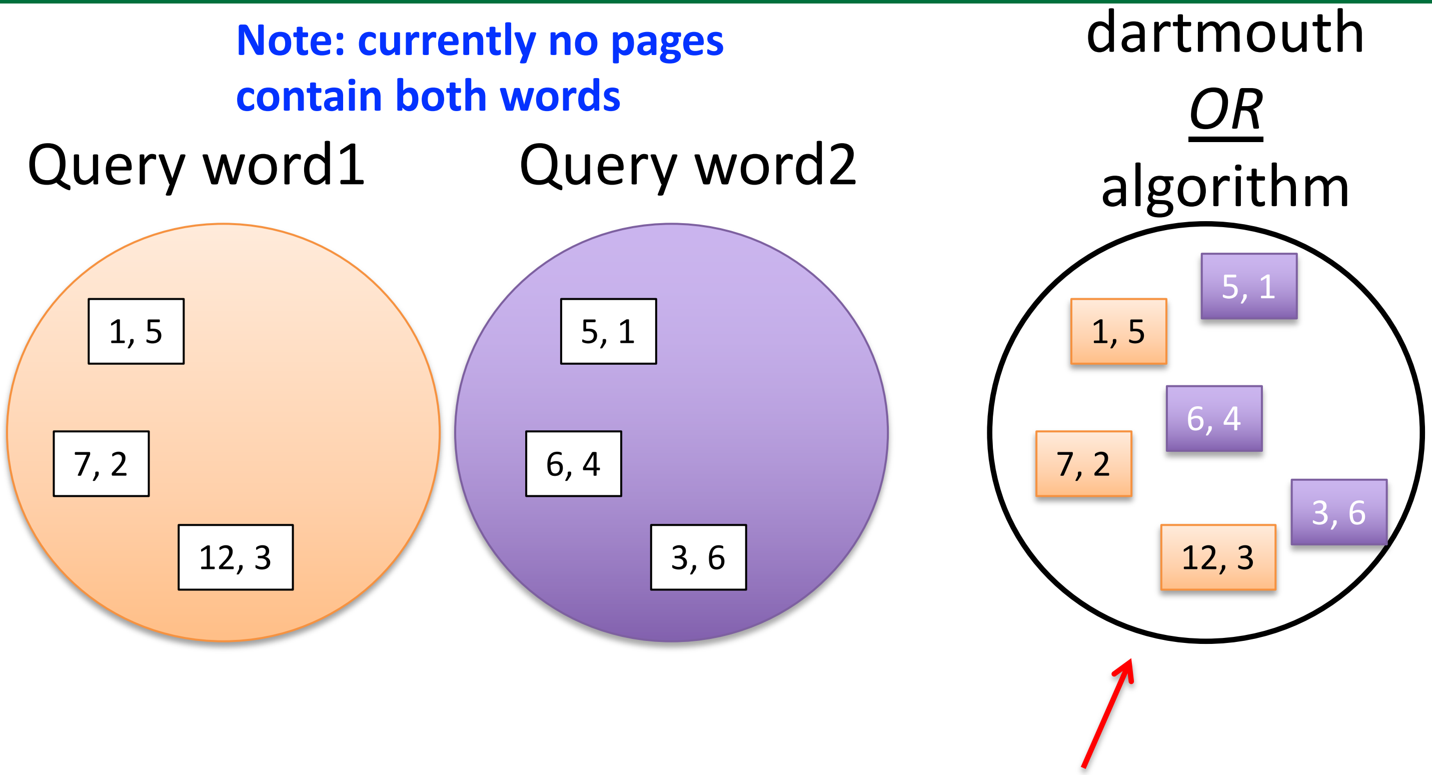
Example:
Doc 6 contains word2 4 times

Example
Word 1 = dartmouth
word 2 = algorithm

Dartmouth appears on sites 1, 7, and 12

Algorithm appears on sites 3, 5, and 6

OR is the UNION of two sets

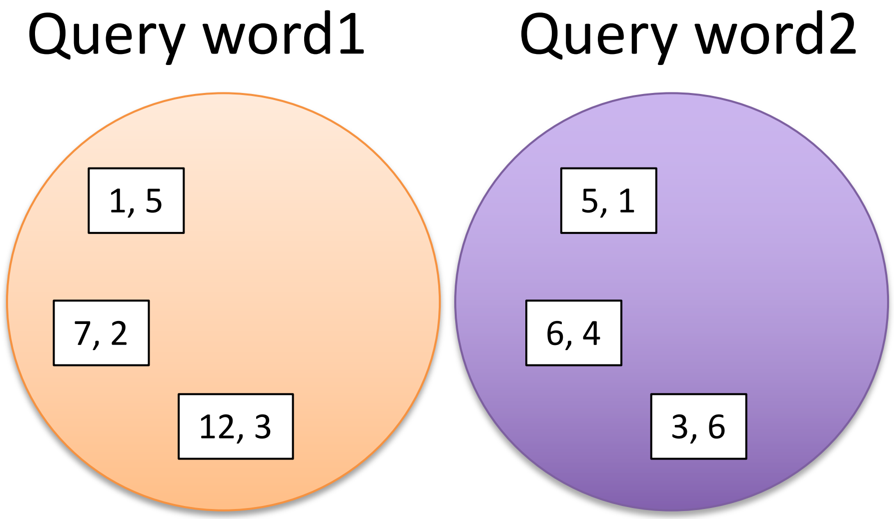


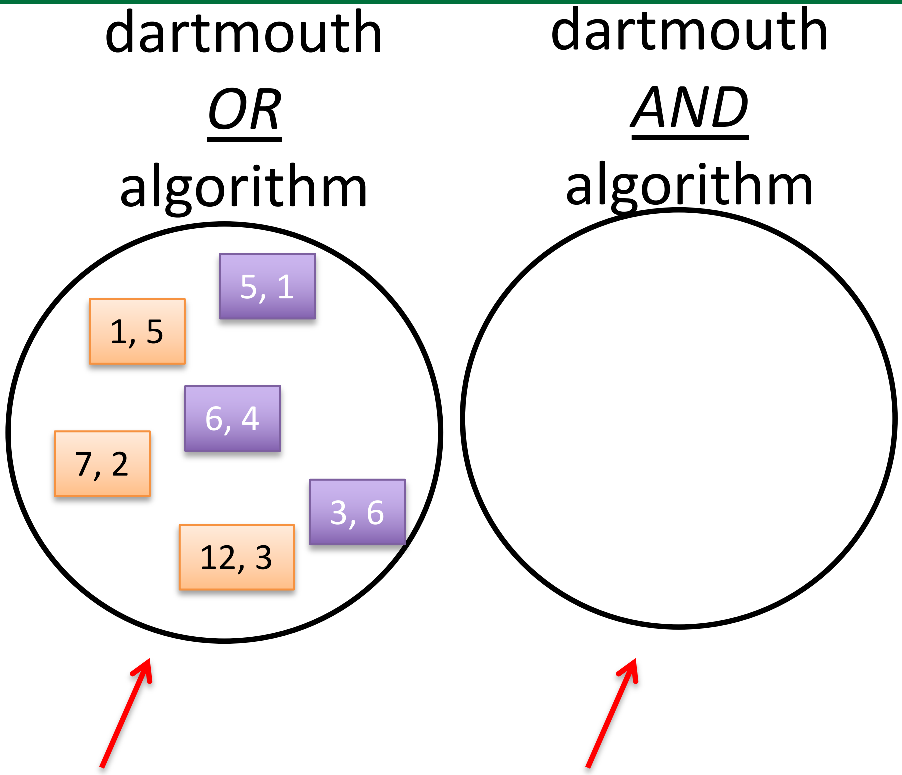
OR returns sites that mention *either* word (UNION)

Return sites sorted by count (3, 1, 6, 12, 7, 5)

AND is the INTERSECTION of two sets

Note: currently no pages contain both words





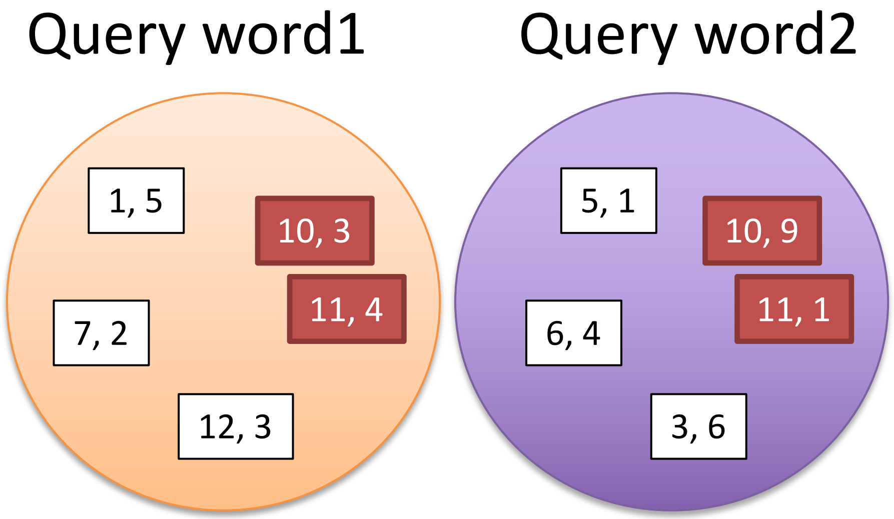
OR returns sites that mention *either* word (UNION)

Return sites sorted by count (3, 1, 6, 12, 7, 5)

AND returns sites that mention *both* words (INTERSECTION)

Currently none

Sometimes the same site contains multiple query words

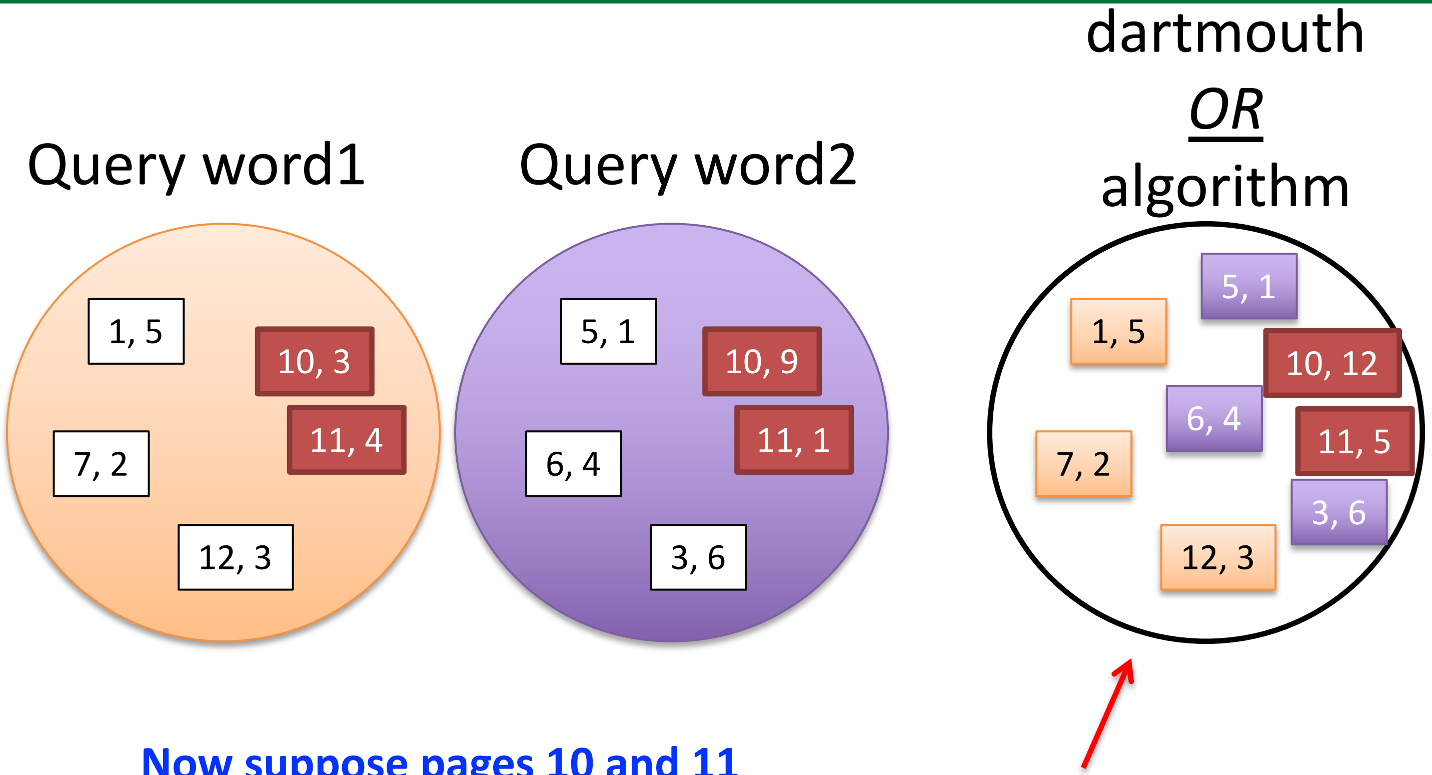


Page 10 contains Dartmouth 3 times and algorithm 9 times

Page 11 contains Dartmouth 4 times and algorithm 1 time

Now suppose pages 10 and 11 each contain query word 1 and 2

OR adds the counts from each site

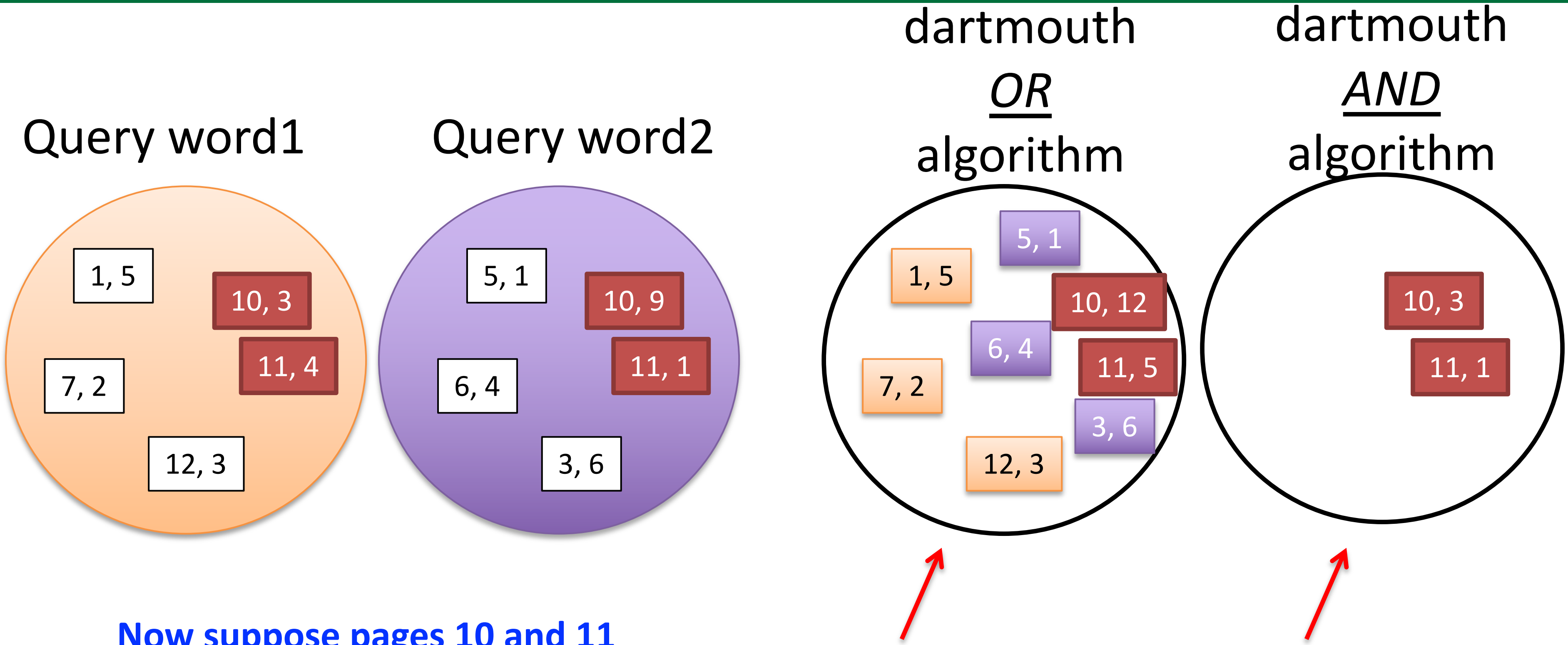


Now suppose pages 10 and 11 each contain query word 1 and 2

OR *adds* counts from each site for overlaps

Still return sites sorted by count (10, 3, 11, 1, 6, 12, 7, 5)

AND takes the minimum count between both sites



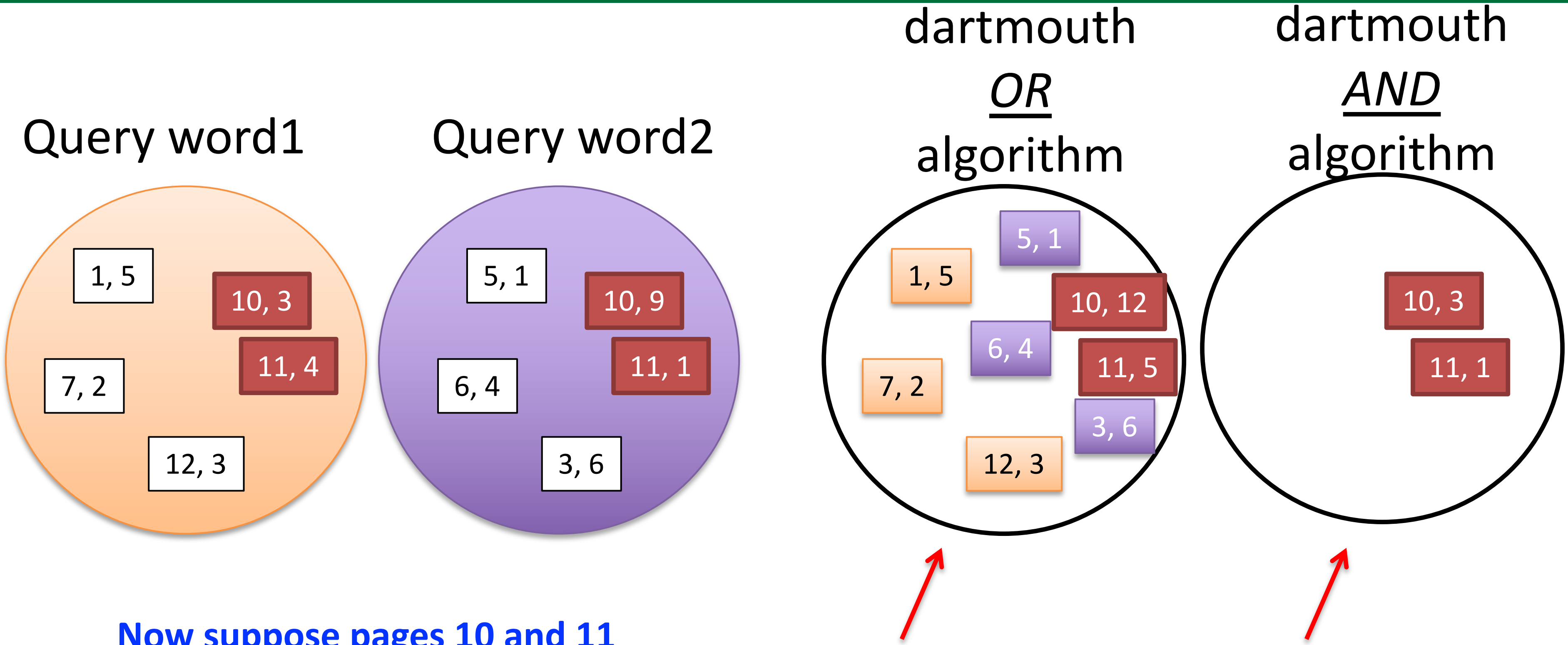
Now suppose pages 10 and 11 each contain query word 1 and 2

OR *adds* counts from each site for overlaps

Still return sites sorted by count (10, 3, 11, 1, 6, 12, 7, 5)

AND takes the *minimum* of each site
Return sites sorted by count (10, 11)

AND takes the minimum count between both sites



Now suppose pages 10 and 11 each contain query word 1 and 2

If query words appear on more than two sites, add counts for OR, take min for AND

OR *adds* counts from each site for overlaps

Still return sites sorted by count (10, 3, 11, 1, 6, 12, 7, 5)

AND takes the *minimum* of each site
Return sites sorted by count (10, 11)

Agenda

1. Union and intersect
2. ➡ Math operator precedence
3. Query operator precedence
4. Activity

In math, multiplication takes precedence over addition

```
sum = a * b + c * d * e + f + g * h * i
    = (a * b) + (c * d * e) + f + (g * h * i)
```

How to solve?

- **Find all multiplications**
- **Do multiplications**
- **Add results and any additions (like f)**

OR

Parse in one pass!

- **Can think of "stepping away" to do multiplication (higher precedence)**
- **Return for addition (lower precedence)**

In math, multiplication takes precedence over addition

$\text{sum} = a * b + c * d * e + f + g * h * i$
$= (a * b) + (c * d * e) + f + (g * h * i)$

Rules:

```
initialize sum = 0, prod = 1
read one token at a time
        If read number
            prod *= number
        if read *
            continue
        if read +
            sum += prod
            prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
 = (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	

Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$\text{sum} = a * b + c * d * e + f + g * h * i$$
$$= (a * b) + (c * d * e) + f + (g * h * i)$$

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a

Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i

    = (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue

Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
        If read number
            prod *= number
        if read *
            continue
        if read +
            sum += prod
            prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
      = (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b

Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
        If read number
            prod *= number
        if read *
            continue
        if read +
            sum += prod
            prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
= (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b

"Return" for addition

Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i

= (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c

Step away for multiplication

Rules:

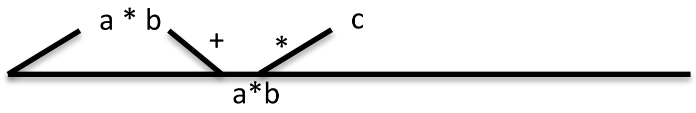
```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i

= (a * b) + (c * d * e) + f + (g * h * i)



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue

Step away for multiplication

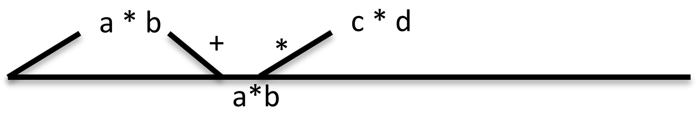
Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$\text{sum} = a * b + c * d * e + f + g * h * i$$
$$= (a * b) + (c * d * e) + f + (g * h * i)$$



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d

Step away for multiplication

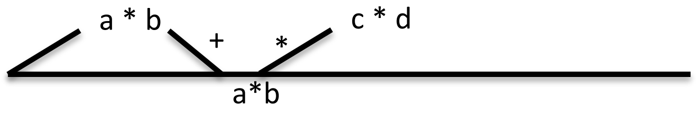
Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can’t end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
= (a * b) + (c * d * e) + f + (g * h * i)



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue

Step away for multiplication

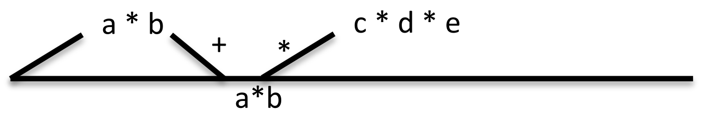
Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
 = (a * b) + (c * d * e) + f + (g * h * i)



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e

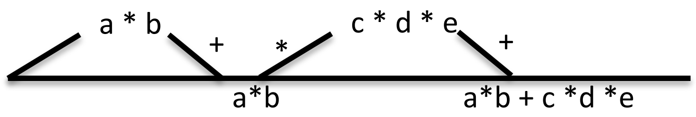
Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
        If read number
            prod *= number
        if read *
            continue
        if read +
            sum += prod
            prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
= (a * b) + (c * d * e) + f + (g * h * i)



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e

Return for addition

Notice: only add prod to sum on +

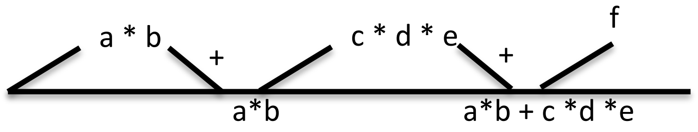
Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$sum = a * b + c * d * e + f + g * h * i$$
$$= (a * b) + (c * d * e) + f + (g * h * i)$$



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f

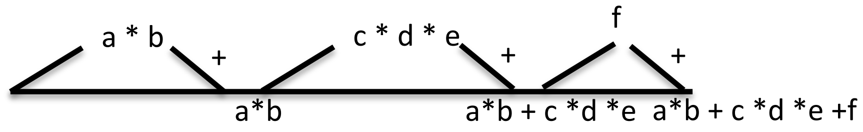
Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$\begin{aligned} sum &= a * b + c * d * e + f + g * h * i \\ &= (a * b) + (c * d * e) + f + (g * h * i) \end{aligned}$$



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f

Return for addition

Notice: only add prod to sum on +

Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
= (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f
g		prod * g	= g

Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$\begin{aligned} sum &= a * b + c * d * e + f + g * h * i \\ &= (a * b) + (c * d * e) + f + (g * h * i) \end{aligned}$$

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f
g		prod * g	= g
*			continue

Step away for multiplication

Rules:

```
initialize sum = 0, prod = 1
read one token at a time
        If read number
            prod *= number
        if read *
            continue
        if read +
            sum += prod
            prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$\text{sum} = a * b + c * d * e + f + g * h * i$$
$$= (a * b) + (c * d * e) + f + (g * h * i)$$

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f
g		prod * g	= g
*			continue
h		prod * h	= g * h

Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

$$\begin{aligned} sum &= a * b + c * d * e + f + g * h * i \\ &= (a * b) + (c * d * e) + f + (g * h * i) \end{aligned}$$

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f
g		prod * g	= g
*			continue
h		prod * h	= g * h
*			continue

Step away for multiplication

```
Rules:
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i
  = (a * b) + (c * d * e) + f + (g * h * i)

Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f
g		prod * g	= g
*			continue
h		prod * h	= g * h
*			continue
i		prod * i	= g * h * i

Step away for multiplication

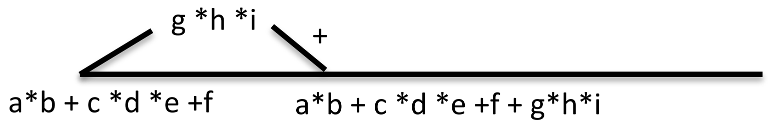
Rules:
```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i

= (a * b) + (c * d * e) + f + (g * h * i)



Read	Sum	Prod	Notes
Start	0	1	
a		prod *a	= a
*			continue
b		prod * b	= a * b
+	sum + prod	1	= 0 + a * b
c		prod * c	= c
*			continue
d		prod * d	= c * d
*			continue
e		prod * e	= c * d* e
+	sum + prod	1	= 0 + a * b + c * d * e
f		prod * f	= f
+	sum + prod	1	= 0 + a * b + c * d * e + f
g		prod * g	= g
*			continue
h		prod * h	= g * h
*			continue
i		prod * i	= g * h * i
end	sum + prod		= 0 + a * b + c * d * e + f + g * h * i

Return for addition

Notice: only add prod to sum on +

Rules:

```
initialize sum = 0, prod = 1
read one token at a time
    If read number
        prod *= number
    if read *
        continue
    if read +
        sum += prod
        prod = 1
return sum + prod
```

Formula can't end with * or +

Agenda

1. Union and intersect
2. Math operator precedence
3. Query operator precedence
4. Activity

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = NULL

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

```
./querier $loc/tse/tse-output/toscrape-depth-2 $loc/tse/tse-output/toscrape-index-2
```

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = NULL

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
   If read a word (not AND or OR)
      find counters for this word in index (index_find(index, word))
      if temp == NULL
         temp = counters for word
      else
         temp = temp ^ counters for word  //intersect on AND
   else if read OR
      result = result v temp //union on OR
      temp = NULL
   else if read AND
      continue to next word  //implicit AND between words
Return result v temp //union
```

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = ~~NULL~~ (380,7) (166,2)

Query: computer
Matches 2 documents (ranked):
score   7 doc 380:
score   2 doc 166:

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = (380,7) (166,2)

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = (380,7) (166,2)

Query: science
Matches 129 documents (ranked):

```
score   9 doc  27:
score   6 doc  55:
score   6 doc 248:
score   4 doc 380:
<snip>
```

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = (380,7) ~~(166,2)~~

Query: science
Matches 129 documents (ranked):
score   9 doc 27:
score   6 doc 55:
score   6 doc 248:
score   4 doc 380:
<snip>

temp ^ science   //intersect: take min of counts
score   4 doc 380:

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = NULL
temp = (380,4) ~~(166,2)~~

Query: science
Matches 129 documents (ranked):

```
score   9 doc  27:
score   6 doc  55:
score   6 doc 248:
score   4 doc 380:
<snip>
```

temp ^ science   //intersect: take min of counts

```
score   4 doc 380:
```

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4)
temp = NULL

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
   If read a word (not AND or OR)
      find counters for this word in index (index_find(index, word))
      if temp == NULL
         temp = counters for word
      else
         temp = temp ^ counters for word  //intersect on AND
   else if read OR
      result = result v temp //union on OR
      temp = NULL
   else if read AND
      continue to next word  //implicit AND between words
Return result v temp //union
```

result = result v science   //union: take sum of counts
score   4 doc 380:

temp = NULL

Step back to calculate OR in result

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4)
temp = ~~NULL~~ (40,2) (240,2) (58,1)

Query: biology
Matches 3 documents (ranked):
score   2 doc 40:
score   2 doc 240
score   1 doc 58:

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (58,1)
temp = NULL

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
   If read a word (not AND or OR)
      find counters for this word in index (index_find(index, word))
      if temp == NULL
         temp = counters for word
      else
         temp = temp ^ counters for word  //intersect on AND
   else if read OR
      result = result v temp //union on OR
      temp = NULL
   else if read AND
      continue to next word  //implicit AND between words
Return result v temp //union
```

Step back to calculate OR in result

result = result v temp   //union: take sum of counts
temp = NULL

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (58,1)
temp = ~~NULL~~ (161,2) (318,2) (385,2) (330,1)

Query: depth
Matches 4 documents (ranked):
score   2 doc 161:
score   2 doc 318:
score   2 doc 385:
score   1 doc 330:

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (58,1)
temp = (161,2) (318,2) (385,2) (330,1)

Query: first
Matches 131 documents (ranked):
score 8 doc 27:
score 6 doc 37
score 6 doc 478:
<snip>
score 2 doc 385:
<snip>

Implicit AND

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (58,1)
temp = (161,2) (318,2) (385,2) (330,1)

```
Query: first
Matches 131 documents (ranked):
score   8 doc  27:
score   6 doc  37
score   6 doc 478:
<snip>
score 2 doc 385:
<snip>
```

temp ^ first   //intersect: take min of counts
score   2 doc 385:

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
   If read a word (not AND or OR)
      find counters for this word in index (index_find(index, word))
      if temp == NULL
         temp = counters for word
      else
         temp = temp ^ counters for word  //intersect on AND
   else if read OR
      result = result v temp //union on OR
      temp = NULL
   else if read AND
      continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (58,1)
temp = ~~(161,2) (318,2)~~ (385,2) ~~(330,1)~~

```
Query: first
Matches 131 documents (ranked):
score   8 doc  27:
score   6 doc  37
score   6 doc 478:
<snip>
score 2 doc 385:
<snip>
```

temp ^ first   //intersect: take min of counts
score   2 doc 385:

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Step away to calculate AND in temp

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (385,2) (58,1)
temp = (385,2)

result = result v temp   //union: take sum of counts

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
   If read a word (not AND or OR)
      find counters for this word in index (index_find(index, word))
      if temp == NULL
         temp = counters for word
      else
         temp = temp ^ counters for word  //intersect on AND
   else if read OR
      result = result v temp //union on OR
      temp = NULL
   else if read AND
      continue to next word  //implicit AND between words
Return result v temp //union
```

Step back to calculate OR in result

In TSE, AND takes precedence over OR, step away to handle AND, step back for OR

computer and science or biology or depth first

result = (380,4) (40,2) (240,2) (385,2) (58,1)
temp = (385,2)

result = result v temp //union: take sum of counts

```
counters_t *result = NULL
counters_t *temp = NULL
//Note: v = union, ^ = intersection
Read query one word at a time
  If read a word (not AND or OR)
    find counters for this word in index (index_find(index, word))
    if temp == NULL
      temp = counters for word
    else
      temp = temp ^ counters for word  //intersect on AND
  else if read OR
    result = result v temp //union on OR
    temp = NULL
  else if read AND
    continue to next word  //implicit AND between words
Return result v temp //union
```

Query: computer and science or biology or depth first
Matches 5 documents (ranked):
score 4 doc 380:
score 2 doc 40:
score 2 doc 240:
score 2 doc 385:
score 1 doc 58:

How to rank?
- **Loop over result, find largest and print**
- **Set largest count to 0**
- **Loop over result and find next largest**
- **Repeat**

Step back to calculate OR in result

Agenda

1. Union and intersect
2. Math operator precedence
3. Query operator precedence
4. Activity

set_iterate2.c demonstrated UNION of two sets

```c
/* Merge the second set into the first set;
 * the second set is unchanged.
 */
static void
set_merge(set_t *setA, set_t *setB)
{
  set_iterate(setB, setA, set_merge_helper);
}

/* Consider one item for insertion into the other set.
 * If the other set does not contain the item, insert it;
 * otherwise, update the other set's item with sum of item values.
 */
static void
set_merge_helper(void *arg, const char *key, void *item)
{
  set_t *setA = arg;
  int *itemB = item;

  // find the same key in setA
  int *itemA = set_find(setA, key);
  if (itemA == NULL) {
    // not found: insert it
    set_insert(setA, key, intsave(*itemB));
    printf("\t%s added\n", key);
  } else {
    // add to the existing value
    *itemA += *itemB;
    printf("\t%s exists\n", key);
  }
}
```

Merge two sets, setB into setA
- **Iterate over setB**
- **For each node in setB, pass setA as a parameter (*arg*)**
- **Pass function to merge (*merge_helper*)**

Store result in setA

Get key and item from setB node

Cast *arg* as set (setA)

```c
void
set_iterate(set_t* set, void* arg,
       void (*itemfunc)(void* arg, const char* key, void* item) )
{
  if (set != NULL && itemfunc != NULL) {
    // call itemfunc with arg, on each item
    for (setnode_t* node = set->head; node != NULL; node = node->next) {
      (*itemfunc)(arg, node->key, node->item);
    }
  }
}
```

If setB's key not in setA
- **insert setB's key and item to setA**

else setB's key in setA
- **add items together**

In set_iterate
- **Loop over all nodes**
- **Pass setB's key and item to *merge_helper***

For TSE we will add document counts for UNION (OR)

counters_intersect.c demonstrated the INTERSECTION of two counters

```c
struct twocts {
   counters_t *result;
   counters_t *another;
};
.
.
.
// TODO: fill in this function
void counters_intersect(counters_t* ct1, counters_t* ct2)
{
   mem_assert(ct1, "counters 1 invalid");
   mem_assert(ct2, "counters 2 invalid");

   struct twocts args = {ct1, ct2};
   counters_iterate(ct1, &args, intersect_helper);
}

void intersect_helper(void *arg, const int key, const int count)
{
   struct twocts *two = arg;

   counters_set(two->result, key, min(count, counters_get(two->another, key)));
}
```

Intersect two counters, ct1 and ct2

- **Iterate over ct1**
- **For each node in ct1, pass ct1 and ct2 as arg parameter in struct with two counters**
- **Pass function to intersect (*intersect_helper*)**

Store result in ct1

Cast *arg* as struct

Remember, *counters_get* returns 0 if key not found

Update first counter in struct (ct1)

- **Set ct1's key to *min* of ct1's value or ct2's value**

For TSE we will *min* document counts in intersect (AND)

In math, multiplication takes precedence over addition

sum = a * b + c * d * e + f + g * h * i = (a * b) + (c * d * e) + f + (g * h * i)
becomes

```
sum = 0
      prod = 1
      prod = prod * a
      prod = prod * b
sum = sum + prod
      prod = 1
      prod = prod * c
      prod = prod * d
      prod = prod * e
sum = sum + prod
      prod = 1
      prod = prod * f
sum = sum + prod
      prod = 1
      prod = prod * g
      prod = prod * h
      prod = prod * I
sum = sum + prod
```

Track

- ***sum*** **starting at 0**
- ***prod*** **starting at 1**

If encounter term or multiplication, multiply ***prod*** **by term**

Add ***prod*** **to** ***sum*** **when encounter addition**

Reset ***prod*** **to 1**

If encounter term or multiplication, multiply ***prod*** **by term**

NOTICE: we never add anything to ***sum*** **other than** ***prod***

Can think of “stepping away” to do multiplication and returning for addition

In TSE, AND takes precedence over OR

Query: computer science or algorithm or depth first
= (computer AND science) OR algorithm OR (depth AND first)

Can think of “stepping away” to do AND (intersection) and returning to do OR (union)

```
counters_t andSequence = NULL; //step away to calculate AND sequence
counters_t orSequence = NULL;  //store final result combining OR sequences

Step away to calculate AND in andSequence
    computer: andSequence = find_index(index, “computer”)
    science: INTERSECT(andSequence, find_index(index, “science”)
OR (step back to merge andSequence with orSequence)
    UNION(orSequence, andSequence)  //add counts, store results in orSequence
    andSequence = NULL
Step away to calculate AND in andSequence
    algorithm: andSequence = find_index(index, “algorithm)
OR (step back to merge andSequence with orSequence)
    UNION(orSequence, andSequence) //add counts, store results in orSequence
    andSequence = NULL
Step away to calculate AND in andSequence
    depth: andSequence = find_index(index, “depth”)
    first: INTERSECT(andSequence, find_index(index, “first”)
All words process (step back to merge andSequence with orSequence)
    UNION(orSequence, andSequence) //add counts, store results in orSequence
    return orSequence
```

Accumulate results in orSequence