Advanced Safe DAAT

Wei Wang

UNSW

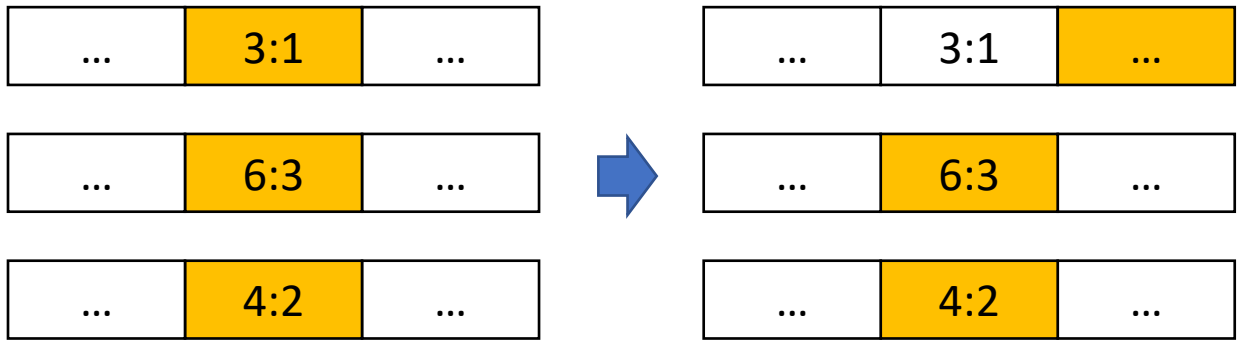
DAAT

- Idea
  - Access and score each document before moving to the next, based on the inverted index
- Invariant
  - All the documents wit urDoc has been processed

#doc to lists of the query

… : curDoc

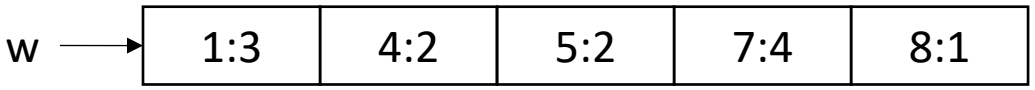
Always move the smaller



DAAT

- Top-k optimization
  - Current top-k-th document's score = threshold
- Optimization
  - No need to access/sco e is < threshold
    - ➔ Skipping docIDs, but how?
- Preliminaries:
  - UB(w): upper bound of the score contribution of any document in w's postings list

$$UB(w) = \text{idf}(w) \cdot \max_{e \in L} e.\text{tf}$$



← Assume no normalization on the raw tf, UB(w) = idf(w)*4

Idea 1

Generating the candidates

Scoring the candidates

- Consider Q = A B C D
  - Assume |A| ≤ |B| ≤ |C| ≤ |D|
  - Can we split the query into, e.g., Q1 = A OR B, and Q2 = C D
  - Answering Q1 is more efully only accessing lists in Q2 for the final scoring
- Working out a sufficient condition for the abo

MaxScore(d3) = ?, if d3 does not exist in A or B's list

➔ ___

If threshold ≥ 2, what can you infer?

Determine the al Terms

- Only need to focus on documents that occurred **ONLY** in the lists of optional terms ➔ Estimate their upper bounding score

- Algorithm:
  - sort terms in decreasi
  - find the largest suffix of the terms suc cumulative UB values is larger than the threshold (current top- nt's score)

Simplified MaxS

- Assume all idf = 1, threshold = 2, and Q1 = A B (required terms)
- Step 1: generate candi
  - T = Result(Q1)
- Step 2: score candidates and get to
  - Foreach d in T:
    - Score(d) /* using lists in Q2 = C */
  - Keep top-k documents as the answer

optional terms

Any problem with this alg?

- Large candidate size:
  - [|A|, |A|+|B|]
- The same threshold is used throughout

Killing two birds with one stone!

A	…	1:3	…	UB = 5
B	…	23:2	…	UB = 3

T = {1, 23}

C	1:1	33:1	55:1

score(d1) = 4

score(d23) = 2

MaxScore

- [Step 0] Obtain the initial threshold α
  - Update the required terms
- Repeat until the *stoppi*

[Step 1]
  - Perform one DAAT ste tain (d, s1), where s1 is the partial score of d on required terms

[Step 2]
  - Score d using the optional terms via sk

[maintenance]
  - If d's final score is larger than α
    - Update the top-k results
    - Update α
    - Update the required terms and optional terms

- Fixed order of terms (decreasing UB values)
- From the rear of the list, find the maximum number of terms such that their UB sum ≤ α

Fixed order = C B A

Example (k = 2)

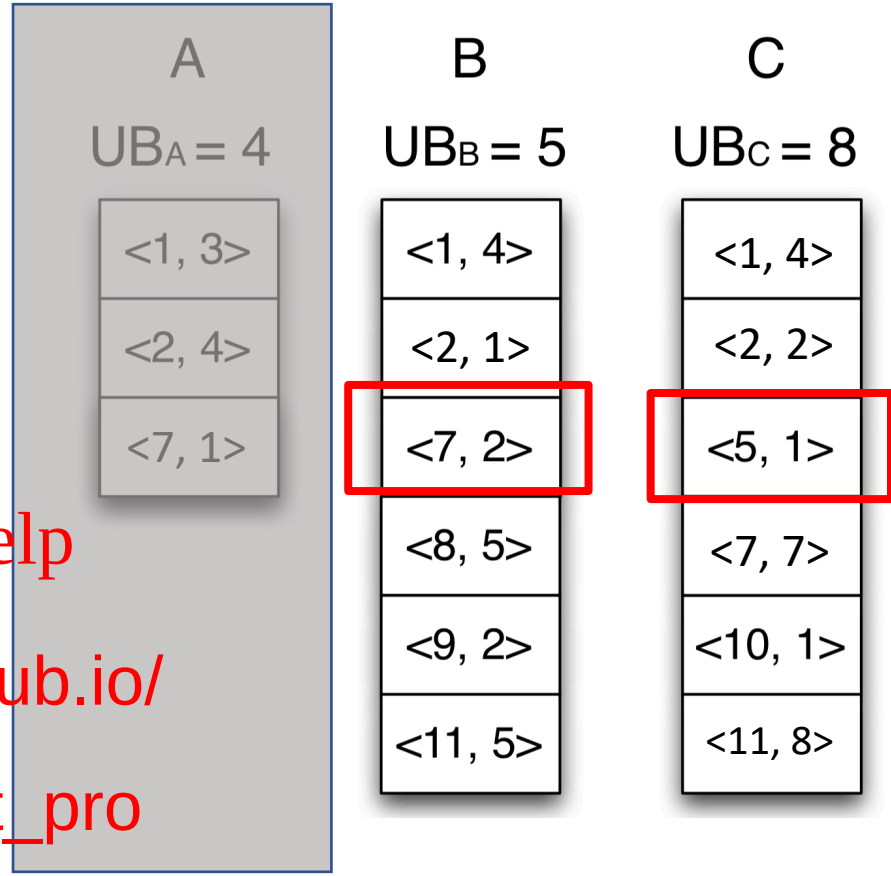
- Step 0:
  - d1 = 11, d2 = 7
  - α =
  - Q1 =

A	B	C
$UB_A = 4$	$UB_B = 5$	$UB_C = 8$
<1, 3>	<1, 4>	<1, 4>
<2, 4>	<2, 1>	<2, 2>
<7, 1>	<7, 2>	<5, 1>
	<8, 5>	<7, 7>
	<9, 2>	<10, 1>
	<11, 5>	<11, 8>

Fixed order = C B A

Example (k = 2)

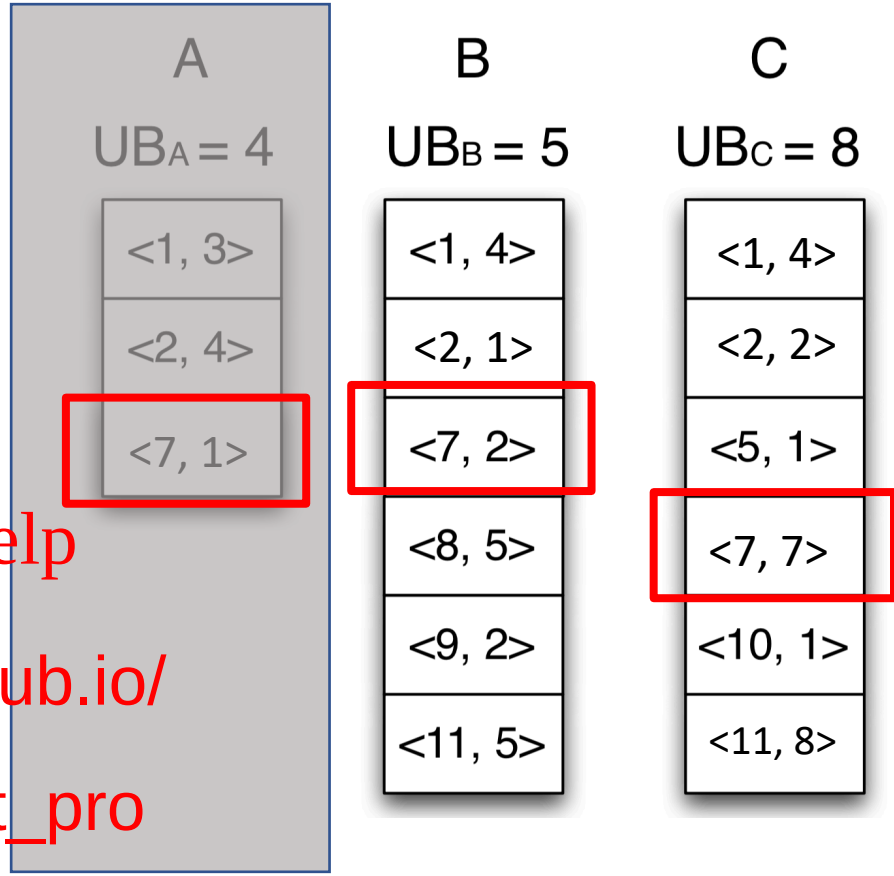
- α = 7
- Iter 1:
  - curDoc = d5
  - partial score = 1
  - "probe" A to get full score = 1
  - Nothing to update



Fixed order = C B A

Example (k = 2)

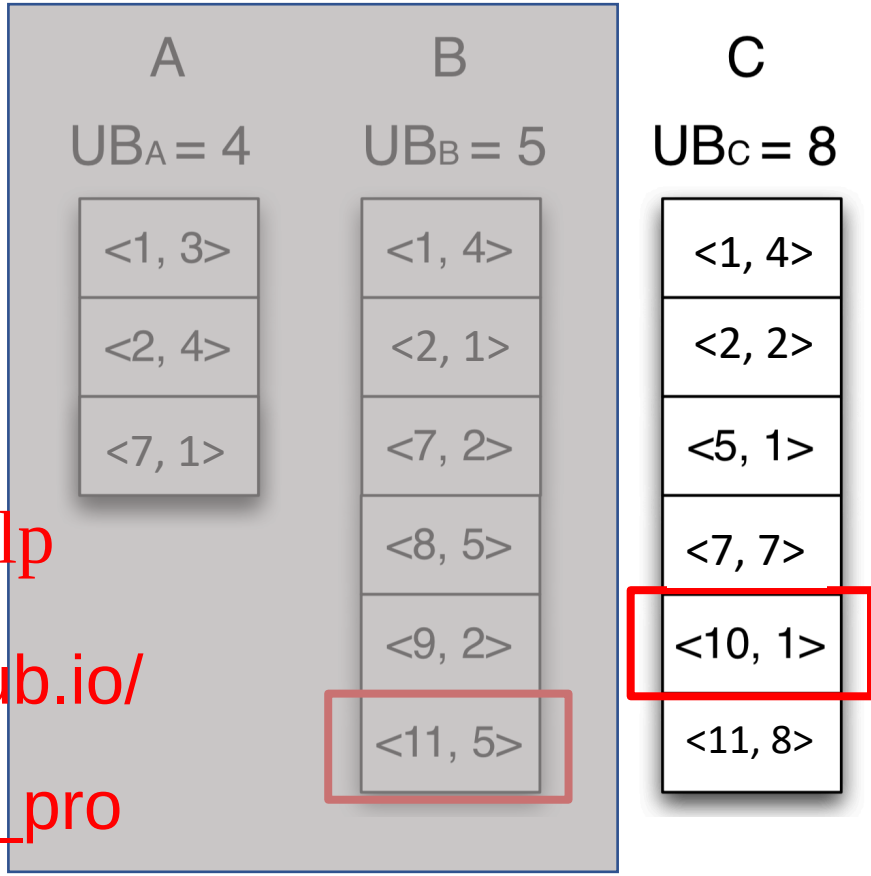
- α = 7
- Iter 2:
  - curDoc = d7
  - partial score = 9
  - "probe" A to get full score = 10
  - Update
    - α = 10
    - Q1 = C



Fixed order = C B A

Example (k = 2)

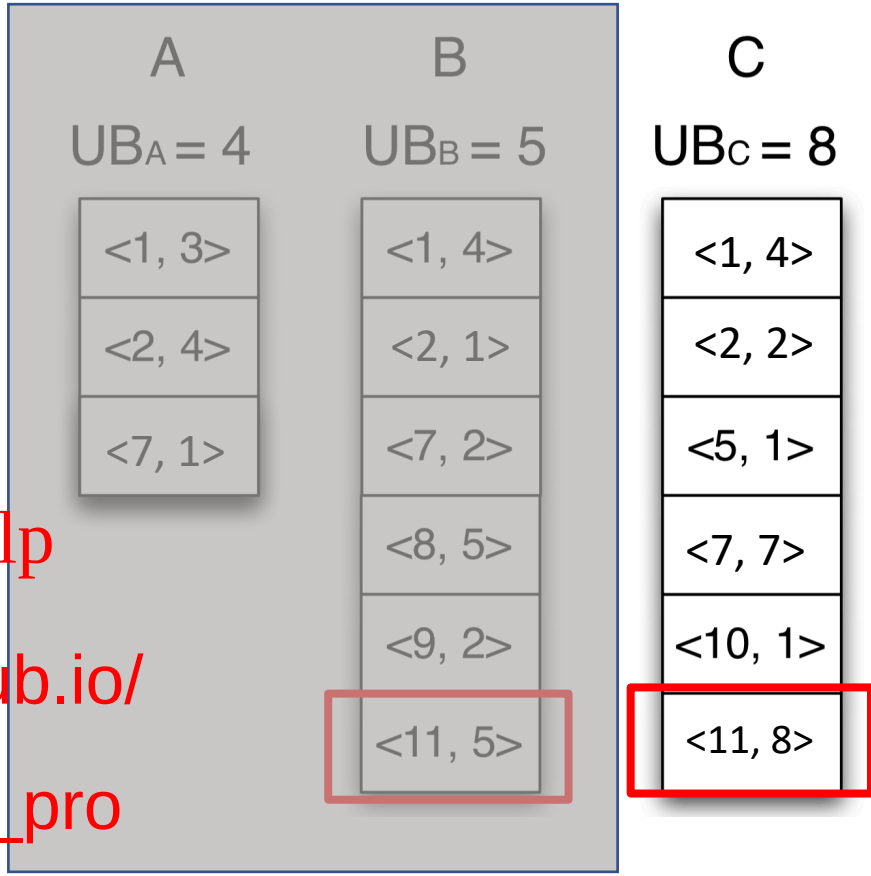
- α = 10
- Iter 3:
  - curDoc = d10
  - partial score = 1
  - "probe" A and B to get full score = 1
  - Nothing to update



Fixed order = C B A

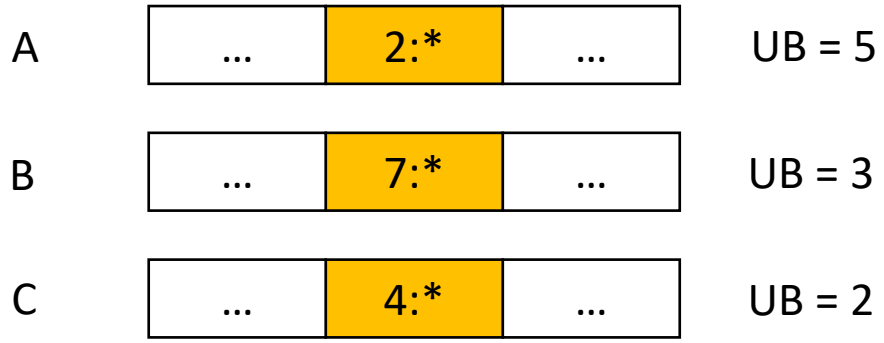
Example (k = 2)

- α = 10
- Iter 3:
  - curDoc = d11
  - partial score = 8
  - "probe" A and B to get full score = 13
  - Update
    - α = 11
    - Q1 = C
- End



- This is the typical stopping condition; There is another possible stopping condition though. Can you figure it out? (not in this example)
- We can further optimize the algorithm to remove unnecessary "probes" on Q2 list(s). Can you find it out?

Idea 2



Sorted Term	A	C	B
Doc	*	*	*
Cumulative Upper Bound	5	7	10

if α = 9, the first document that can score above α is from __B____

Pivot document is the current d m

Pivot term = B

Pivot doc = d7

Is it possible for any doc < PivotDoc to enter int

Case I: smallest DID ≠ PivotDoc

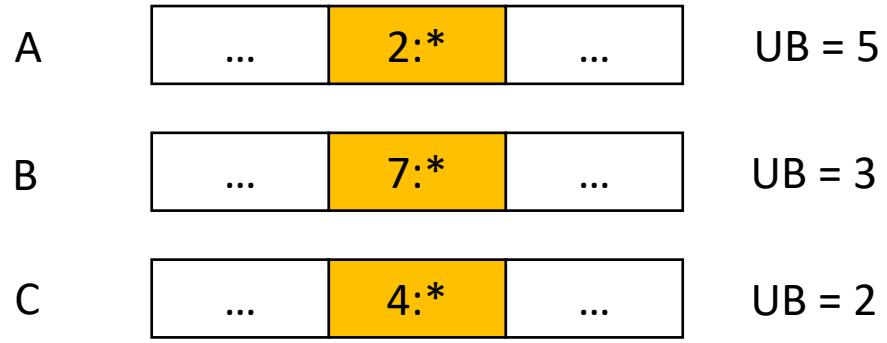
Sorted Term	A	C	B
Doc	2	*	7
Cumulative Upper Bound	5	7	10

score(d2) ≤ 8

➔ align preceding lists to PivotDoc:
A.skipTo(d7), C.skipTo(d7)

➔ Check again

Idea 2



Sorted Term	A	C	B
Doc	*	*	*
Cumulative Upper Bound	5	8	10

if α = 9, the first document that can score above α is from __B____

Pivot document is the current d m

Pivot term = B

Pivot doc = d7

Is it possible for any doc < PivotDoc to enter int

Case II: smallest DID = PivotDoc

Sorted Term	A	C	B
Doc	7	*	7
Cumulative Upper Bound	5	8	10

C.Doc must be d7

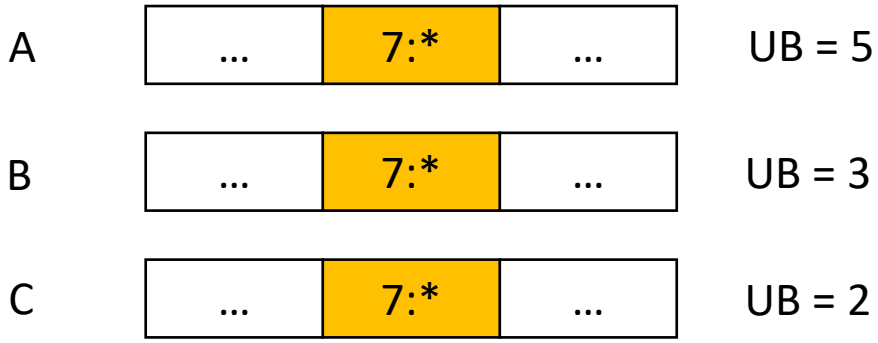
score(d7) could be larger than α

➔ fullScore(d7)

➔ Adjust α if necessary

➔ all list pointing to d7: next()

Idea 2



Sorted Term	A	C	B
Doc	*	*	*
Cumulative Upper Bound	5	8	10

if α = 9, the first document that can score above α is from __B____

Pivot document is the current d m

Pivot term = B

Pivot doc = d7

If all term preceding PivotTerm (in the sorted or n PivotDoc, then PivotDoc is the smallest document that may enter into top-k.

Full scoring:

- Need to "probe" lists that are sorted after the PivotTerm

None in our example, but one can easily add D list.

Based on ”Exploring the Magic of WAND” in ADCS 2013

WAND

1. Initialization

Based on ”Exploring the Magic of WAND” in ADCS 2013

WAND

2. Finding the Pivot

WAND

3a. Case II

fullScore()

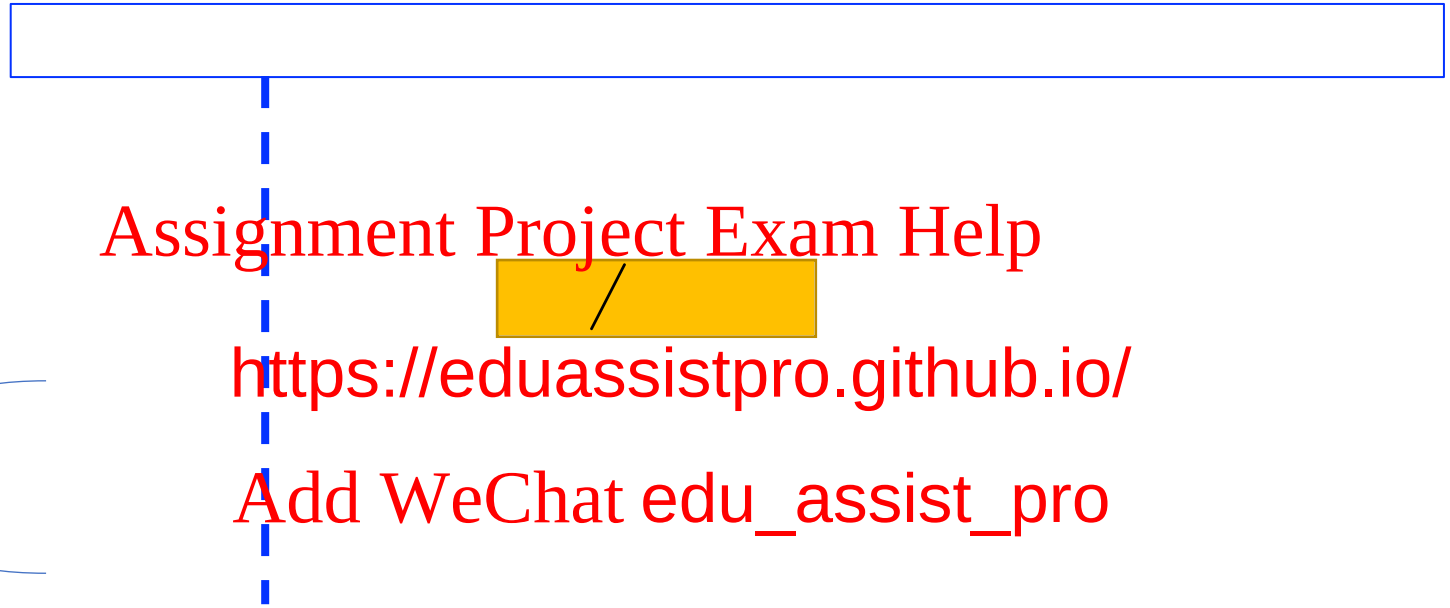
update

updatePointers-I

Based on ”Exploring the Magic of WAND” in ADCS 2013

WAND

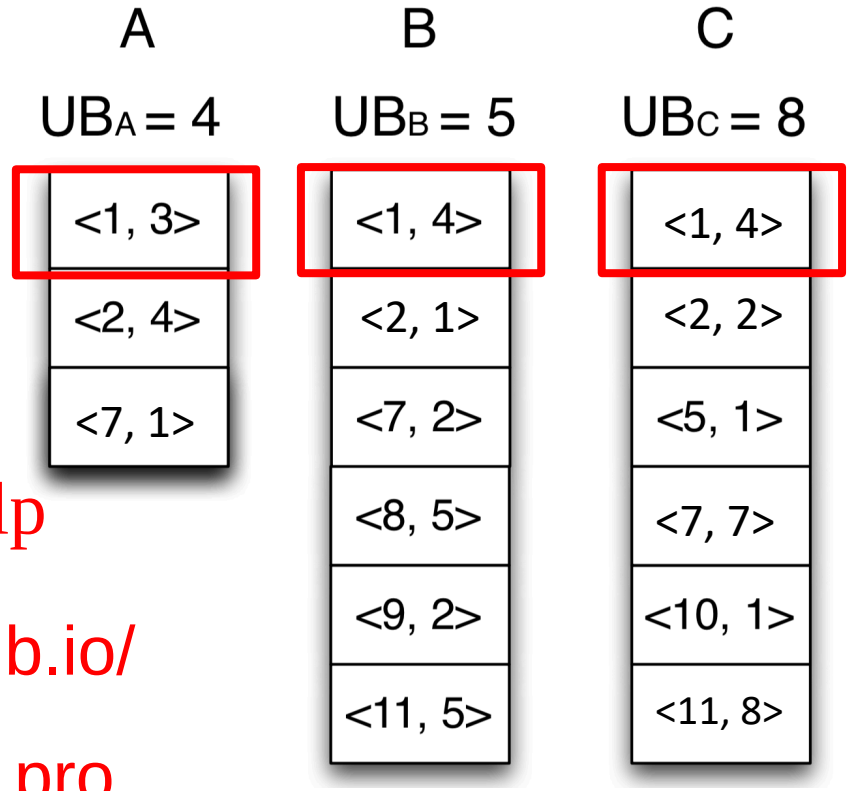
3b. Case I



updatePointers-II

It moves **all** the preceding lists. It is the mWAND optimization for memory-resident index in [Fontoura et al, 2011].

Example (k = 1)

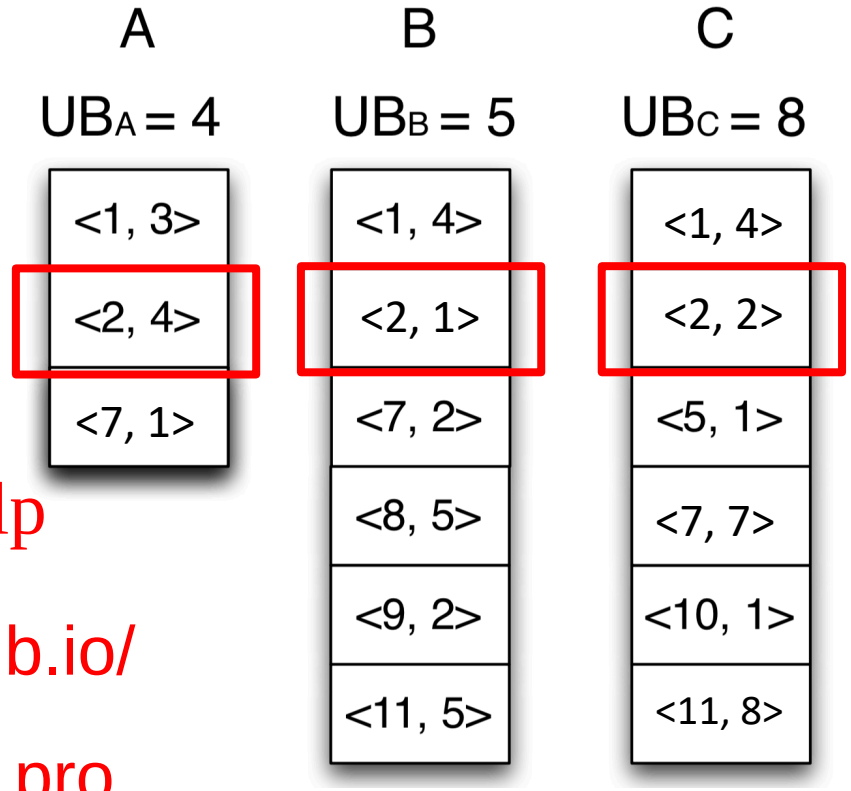


- Step 1:
  - α = 0

Sorted Term	A	B	C
Doc	1		
Cumulative Upper Bound	4		

  - PivotTerm = A
  - PivotDoc = 1
  - Case II
    - fullScore() ➔ d1 = 11, α = 11
  - updatePointers()

Example (k = 1)

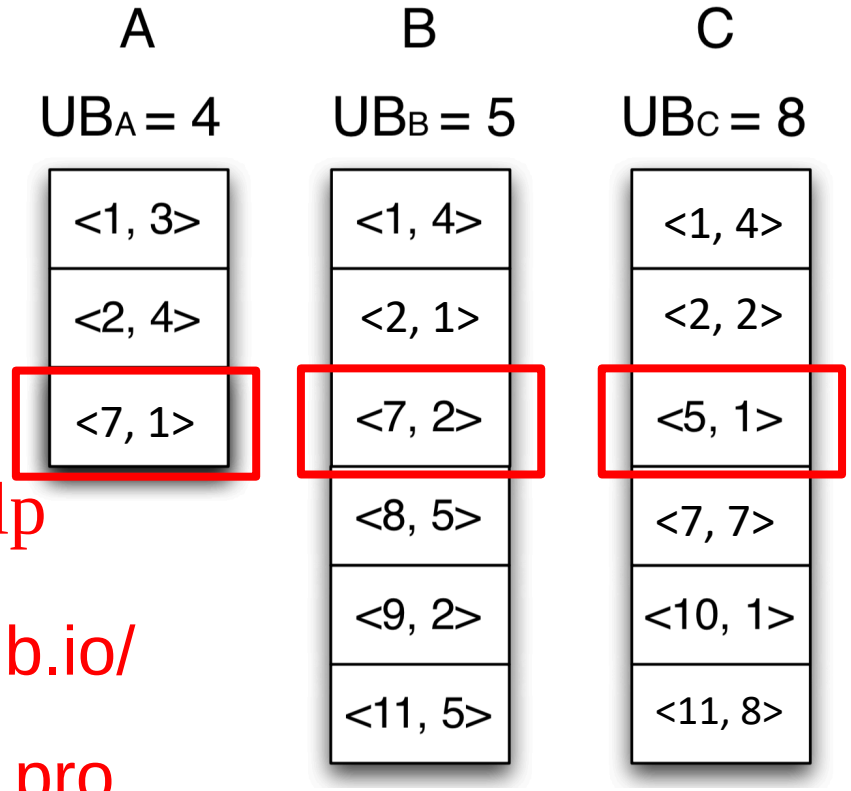


- Step 2:
  - α = 11

Sorted Term	A	B	C
Doc	2		
Cumulative Upper Bound	4		

  - PivotTerm = C
  - PivotDoc = 2
  - Case II
    - fullScore() ➔ d2 = 7, α = 11
  - updatePointers()

Example (k = 1)

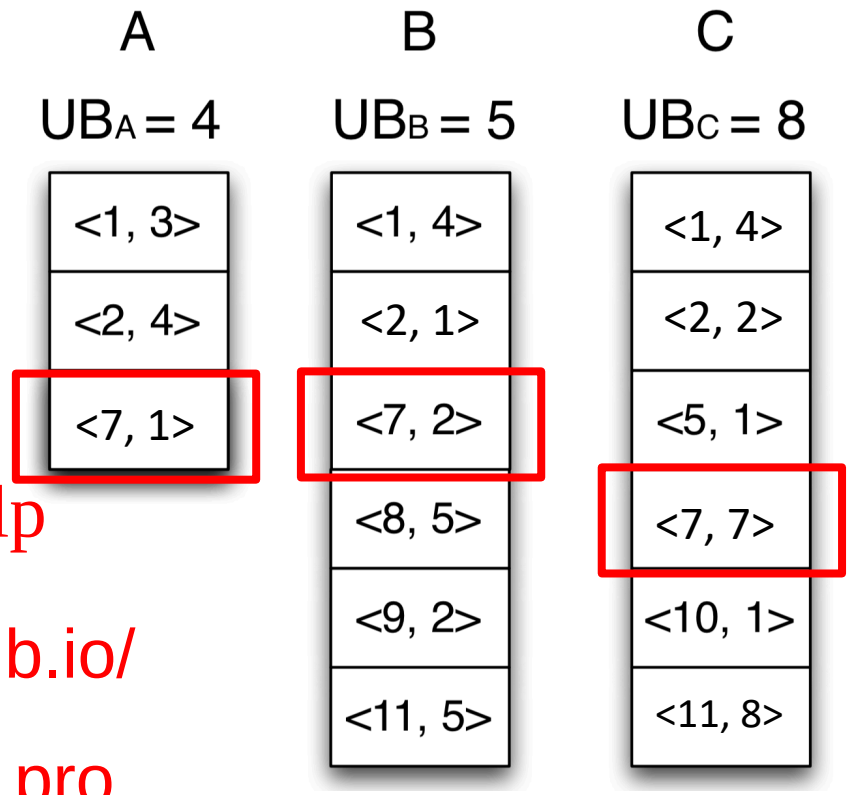


- Step 3:
  - α = 11

Sorted Term	C	B	A
Doc	5		
Cumulative Upper Bound	8		

  - PivotTerm = B
  - PivotDoc = 7
  - Case I
    - updatePointers()

Example (k = 1)



- Step 4:
  - α = 11

Sorted Term	C	B	A
Doc	7		
Cumulative Upper Bound	8		

  - PivotTerm = B
  - PivotDoc = 7
  - Case II
    - fullScore() ➔ d7 = 10, α = 11
  - updatePointers()
- Step 5: …

Comparisons

	MaxScore	(m)WAND
Pruning Strategy	UB based on fixed ordering of terms ➔ "proactive" pruning	UB based on variable ordering of terms ➔ "passive" pruning
Performance		Better for long queries
Applicability	DAAT	DAAT

Hybrid TAAT DA

- Idea
  - Find a good α (with little cost) and run optimized DAAT algorithm
- Q = {A B C D}, in inc length
  - Q1 = {A, B}, Q2 = Q
  - α(Q1), L(Q1) = ProcessQuery(Q1)
    - L(Q1): documents scored for Q1
  - ProcessQueryDAAT(Q2; α(Q1), L(Q1))
    - Treat L(Q1) as another inverted list, UB(L(Q1)) = α(Q1)

References

- Efficient Query Evaluation using a Two-Level Retrieval. CIKM 2003.
- Exploring the Magic of WAND. ADCS 2013.
- [Fontoura et al, 2011] or Top-k Queries over Memory-Resident Inv 11.
- Howard R. Turtle, James Flood. Quer on Strategies and Optimizations. Inf. Process. Manag. 31(6): 831-850 (1995)