

Assignment Project Exam Help

Add WeChat powcoder

# Advanced Safe DAAT

Assignment Project Exam Help

# Algorithms

<https://powcoder.com>

Add WeChat powcoder

Wei Wang

UNSW

# Assignment Project Exam Help

## DAAT Add WeChat powcoder

- Idea

- Access and score each document before moving to the next, based on the inverted index

- Invariant

- All the documents with docID smaller than the curDoc has been processed

Assignment Project Exam Help

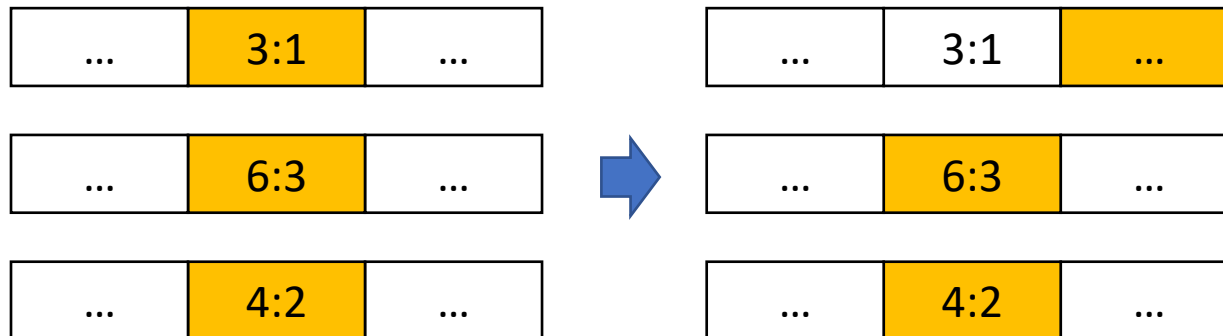
<https://powcoder.com>

#doc to be scored = union of the inverted lists of the query

Add WeChat powcoder

Always move the smaller docID

... : curDoc



# Assignment Project Exam Help

DAAT Add WeChat powcoder

- Top-k optimization
  - Current top-k-th document's score = threshold
- Optimization
  - No need to access/score documents whose score 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,  $UB(w) = \text{idf}(w) * 4$

# Assignment Project Exam Help

## Idea 1 Add WeChat powcoder

- Consider  $Q = A \ B \ C \ D$ 
  - Assume  $|A| \leq |B| \leq |C| \leq |D|$
  - Can we split the query into, e.g.,  $Q1 = A \ OR \ B$ , and  $Q2 = C \ D$
  - Answering  $Q1$  is more efficient (why?), and hopefully only accessing lists in  $Q2$  for the final scoring
- Working out a sufficient condition for the above scheme

Generating the candidates

Scoring the candidates

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

A	<table><tr><td>...</td><td>*.*</td><td>...</td></tr></table>	...	*.*	...	UB = 5
...	*.*	...			
B	<table><tr><td>...</td><td>*.*</td><td>...</td></tr></table>	...	*.*	...	UB = 3
...	*.*	...			
C	<table><tr><td>...</td><td>3:*</td><td>...</td></tr></table>	...	3:*	...	UB = 2
...	3:*	...			

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

➔ \_\_\_\_

If threshold  $\geq 2$ , what can you infer?

## Assignment Project Exam Help

# Determine the Optional 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 decreasing order of their UB values
  - find the largest suffix of the terms such that the accumulative UB values is larger than the threshold (current top-k-th document's score)

# Assignment Project Exam Help

## Simplified MaxScore

- Assume all idf = 1, threshold = 2, and Q1 = A B (required terms)

- Step 1: generate candidates:

- T = Result(Q1)

- Step 2: score candidates and get top-k

- Foreach d in T:
  - Score(d) /\* using lists in Q2 = C \*/
- Keep top-k documents as the answer

Any problem with this alg?

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



Killing two birds  
with one stone!

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

optional terms

A	...	1:3	...	UB = 5	$T = \{1, 23\}$
B	...	23:2	...	UB = 3	

C	1:1	33:1	55:1
---	-----	------	------

score(d1) = 4  
score(d23) = 2

# Assignment Project Exam Help

## MaxScore

Add WeChat powcoder

- [Step 0] Obtain the initial threshold  $\alpha$

- Update the required terms

Assignment Project Exam Help

- Repeat until the *stopping condition*

<https://powcoder.com>

- [Step 1] • Perform one DAAT step on required terms  $\rightarrow$  obtain  $(d, s1)$ , where  $s1$  is the partial score of  $d$  on required terms

- [Step 2] • Score  $d$  using the optional terms via  $\text{skipTo}(d)$

- [maintenance] • If  $d$ 's final score is larger than  $\alpha$

- Update the top-k results
  - Update  $\alpha$
  - 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  $\leq \alpha$

# Example (k=2)

- Step 0:
  - $d1 = 11, d2 = 7$
  - $\alpha =$
  - $Q1 =$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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>



# Example (k=2)

- $\alpha = 7$

- Iter 1:

- curDoc = d5
- partial score = 1
- “probe” A to get full score = 1
- Nothing to update

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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>

# Assignment Project Exam Help

Fixed order = C B A

## Example (k=2)

- $\alpha = 7$
- Iter 2:
  - curDoc = d7
  - partial score = 9
  - “probe” A to get full score = 10
  - Update
    - $\alpha = 10$
    - Q1 = C

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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>

# Assignment Project Exam Help

Fixed order = C B A

## Example (k=2)

- $\alpha = 10$
- Iter 3:
  - curDoc = d10
  - partial score = 1
  - “probe” A and B to get full score = 1
  - Nothing to update

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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>

# Assignment Project Exam Help

Fixed Order = C B A

## Example (k=2)

- $\alpha = 10$
- Iter 3:
  - curDoc = d11
  - partial score = 8
  - “probe” A and B to get full score = 13
  - Update
    - $\alpha = 11$
    - Q1 = C
- End

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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>

- 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?

# Assignment Project Exam Help

## Idea 2 Add WeChat powcoder

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

A	...	2:*	...	UB = 5
B	...	7:*	...	UB = 3
C	...	4:*	...	UB = 2



### Assignment Project Exam Help

if  $\alpha = 9$ , the first document that can score above  $\alpha$  is from \_\_ B \_\_

Pivot document is the current document for the pivot term

Pivot term = B

Pivot doc = d7

<https://powcoder.com>

Is it possible for any doc < PivotDoc to enter into top-k?

Add WeChat powcoder

Case I: smallest DID  $\neq$  PivotDoc

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

score(d2)  $\leq$  8

→ align preceding lists to PivotDoc:

A.skipTo(d7), C.skipTo(d7)

→ Check again

# Assignment Project Exam Help

## Idea 2 Add WeChat powcoder

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

A	...	2:*	...	UB = 5
B	...	7:*	...	UB = 3
C	...	4:*	...	UB = 2

if $\alpha = 9$ , the first document that can score above $\alpha$ is from __B__
Pivot document is the current document for the pivot term

Pivot term = B

Pivot doc = d7

Is it possible for any doc < PivotDoc to enter into top k?

Add WeChat powcoder

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  $\alpha$

→ fullScore(d7)

→ Adjust  $\alpha$  if necessary

→ all list pointing to d7: next()

# Assignment Project Exam Help

## Idea 2 Add WeChat powcoder



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

A	...	7:*	...	UB = 5
B	...	7:*	...	UB = 3
C	...	7:*	...	UB = 2

if $\alpha = 9$ , the first document that can score above $\alpha$ is from __ B __
Pivot document is the current document for the pivot term

Pivot term = B

Pivot doc = d7

If all term preceding PivotTerm (in the sorted order) agree on 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.

## Assignment Project Exam Help

WAND Add WeChat powcoder

### 1. Initialization

**Algorithm 1** WAND processing. Assignment Project Exam Help

```
function WAND( $q, \mathcal{I}, k$ )  
  for  $t \leftarrow 0$  to  $|q| - 1$  do  
     $U[t] \leftarrow \max_d \{w_d \mid (d, w_d) \in \mathcal{I}_t\}$   
     $(c_t, w_t) \leftarrow \text{first\_posting}(\mathcal{I}_t)$   
5:  end for  
     $\theta \leftarrow -\infty$  // current threshold  
     $Ans \leftarrow \{\}$  //  $k$ -set of  $(d, s_d)$  values
```

<https://powcoder.com>

Add WeChat powcoder



# Assignment Project Exam Help

## WAND Add WeChat powcoder

### 2. Finding the Pivot

```
while the set of candidates  $(c_t, w_t)$  is non-empty do  
    permute the candidates so that  $c_0 \leq c_1 \leq \dots c_{|q|-1}$   
10:     $score\_limit \leftarrow 0$   
     $pivot \leftarrow 0$   
    while  $pivot < |q| - 1$  do  
         $tmp\_s\_lim \leftarrow score\_limit + U[pivot]$   
        if  $tmp\_s\_lim > \theta$  then  
15:            break, and continue from step 20  
        end if  
         $score\_limit \leftarrow tmp\_score\_lim$   
         $pivot \leftarrow pivot + 1$   
    end while
```

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## WAND Add WeChat powcoder

3a. Case II

fullScore()

update

**while** the set of candidates  $(c_t, w_t)$  is non-empty **do**

20:       **if**  $c_0 = c_{pivot}$  **then**

$s \leftarrow 0$   
           $t \leftarrow 0$

// score document  $c_{pivot}$

**while**  $t < |q|$  **and**  $c_t = c_{pivot}$  **do**

$s \leftarrow s + w_t$  // add contribution to score

25:        $(c_t, w_t) \leftarrow next\_posting(\mathcal{I}_t)$

$t \leftarrow t + 1$

**end while** //  $s$  is the score of document  $c_{pivot}$

**if**  $s > \theta$  **then** // and is a possible top- $k$  answer

$Ans \leftarrow insert(Ans, (c_{pivot}, s))$

**if**  $|Ans| > k$  **then**

$Ans \leftarrow delete\_smallest(Ans)$

$\theta \leftarrow minimum(Ans)$

**end if**

**end if**

updatePointers-I

## Assignment Project Exam Help

# WAND Add WeChat powcoder

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].

```
while the set of candidates  $(c_t, w_t)$  is non-empty do
  35:   else  $c_0 \neq c_{pivot}$  // can't score  $c_{pivot}$  (yet)
        for  $i \leftarrow 0$  to  $pivot - 1$  do
             $(c_t, w_t) \leftarrow seek\_to\_document(\mathcal{I}_t, c_{pivot})$ 
        end for // all pointers are now at  $c_{pivot}$  or greater
    end if
40: end while
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## Example (k=1)

- Step 1:

- $\alpha = 0$

Sorted Term	A	B	C
Doc	1	1	1
Cumulative Upper Bound	4	9	17

- PivotTerm = A
- PivotDoc = 1
- Case II
  - fullScore()  $\rightarrow$  d1 = 11,  $\alpha = 11$
- updatePointers()

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>

# Assignment Project Exam Help

## Example (k=1)

- Step 2:

- $\alpha = 11$

Sorted Term	A	B	C
Doc	2	2	2
Cumulative Upper Bound	4	9	17

- PivotTerm = C
- PivotDoc = 2
- Case II
  - fullScore()  $\rightarrow$  d2 = 7,  $\alpha = 11$
- updatePointers()

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>

# Assignment Project Exam Help

## Example (k=1)

- Step 3:

- $\alpha = 11$

Sorted Term	C	B	A
Doc	5	7	7
Cumulative Upper Bound	8	13	17

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

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>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## Example (k=1)

- Step 4:

- $\alpha = 11$

Sorted Term	C	B	A
Doc	7	7	7
Cumulative Upper Bound	8	13	17

- PivotTerm = B

- PivotDoc = 7

- Case II

- $\text{fullScore}() \rightarrow d7 = 10, \alpha = 11$

- $\text{updatePointers}()$

- Step 5: ...

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>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

## 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 short queries	Better for long queries
Applicability	DAAT & TAAI	DAAT

SI	SQ	LQ
Naive DAAT	193.0	4,554.6
mWAND	200.0	2,104.6
DAAT <i>max_score</i>	169.0	2,685.6

LI	SQ	LQ
Naive DAAT	3,581.3	26,778.3
mWAND	1,867.0	7,556.3
DAAT <i>max_score</i>	1,572.6	9,321.3

Table 5: Latency results for naive DAAT, mWAND and DAAT *max\_score*.



# Assignment Project Exam Help

## Hybrid TAAT DAAT

- Idea

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

# Assignment Project Exam Help

## References

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