

WAND Top-K Retrieval

Alfan F. Wicaksono

Fakultas Ilmu Komputer, Universitas Indonesia

Berdasarkan paper "Efficient Query Evaluation using a Two-Level Retrieval Process", by Andrei Z. Broder, D. Carmel, M. Herscovici, A. Soffer, J. Zien



to be or not to be

Ada 22 Milyar hasil yang relevan, namun yang ditampilkan hanya sekitar Top-10 saja.



Login

Semua

Gambar

Video

Buku

Berita

Lainnya

Alat

SafeSearch aktif

Sekitar 22.810.000.000 hasil (0,55 detik)

Kiat: Telusuri hasil dalam bahasa **Indonesia** saja. Anda dapat menentukan bahasa penelusuran di Preferensi

"To be, or not to be" is **the opening phrase of a soliloquy given by Prince Hamlet in the so-called "nunnery scene" of William Shakespeare's play Hamlet, Act 3, Scene 1.** In the speech, Hamlet contemplates death and suicide, bemoaning the pain and unfairness of life but acknowledging that the alternative might be worse.

https://en.wikipedia.org/wiki/To_be,_or_not_to_be

[To be, or not to be - Wikipedia](#)

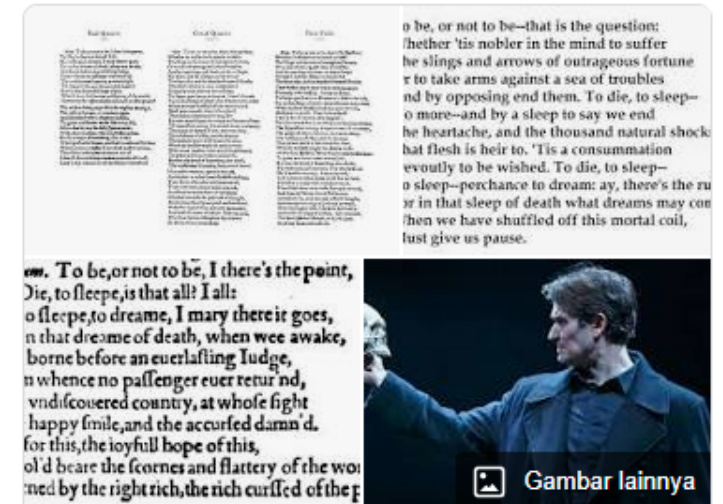
Tentang cuplikan pilihan • Masukan

Orang juga bertanya

What is the full quote To be, or not to be?

Why does Hamlet say To be, or not to be?

What does Shakespeare say To be, or not to be?



To be, or not to be

Diterjemahkan dari bahasa Inggris -

"Menjadi, atau tidak menjadi" adalah frasa pembuka dari solilokui yang diberikan oleh Pangeran Hamlet dalam apa yang disebut "adegan biara" dari drama William Shakespeare Hamlet, Act 3, Scene 1.



to be or not to be



Login

Semua

Gambar

Video

Buku

Berita

Lainnya

Alat

SafeSearch aktif

Sekitar 22.810.000.000 hasil (0,55 detik)

Kiat: Telusuri hasil dalam bahasa **Indonesia** saja. Anda dapat menentukan bahasa penelusuran di Preferensi

"To be, or not to be" is **the opening phrase of a soliloquy given by Prince Hamlet in the so-called "nunnery scene" of William Shakespeare's play Hamlet, Act 3, Scene 1.** In the speech, Hamlet contemplates death and suicide, bemoaning the pain and unfairness of life but acknowledging that the alternative might be worse.

https://en.wikipedia.org/wiki/To_be,_or_not_to_be

[To be, or not to be - Wikipedia](#)

Tentang cuplikan pilihan • Masukan

Orang juga bertanya

What is the full quote To be, or not to be?



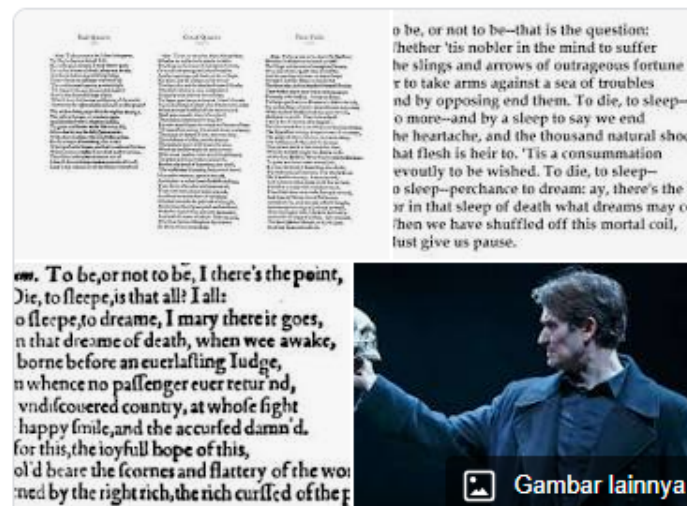
Why does Hamlet say To be, or not to be?



What does Shakespeare say To be, or not to be?



Wow! Hanya 0.55 detik !



To be, or not to be



Diterjemahkan dari bahasa Inggris -

"Menjadi, atau tidak menjadi" adalah frasa pembuka dari solilokui yang diberikan oleh Pangeran Hamlet dalam apa yang disebut "adegan biara" dari drama William Shakespeare Hamlet, Act 3, Scene 1.

Scoring Method

$$\text{score}(Q, D) = \sum_{t \in Q \cap D} \alpha_t w(t, D)$$

Bagian yang tidak
bergantung pada dokumen
(bergantung pada query)

Bagian yang bergantung
pada dokumen D

Scoring Method

$$score(Q, D) = \sum_{t \in Q \cap D} \alpha_t w(t, D)$$

Contoh: **TF-IDF**

$$\alpha_t = tf(t, Q) \cdot \log(N/df(t)) \quad ; \text{ boleh juga dengan } \alpha_t = \log(N/df(t))$$

$$w(t, D) = tf(t, D) \quad ; \text{ boleh juga dengan } w(t, D) = 1 + \log(tf(t, D))$$

Juga dengan varian TF-IDF yang lainnya

Scoring Method

$$score(Q, D) = \sum_{t \in Q \cap D} \alpha_t w(t, D)$$

Contoh: **BM25**

$$\alpha_t = \log(N/df(t))$$

$$w(t, D) = \frac{(k_1 + 1) \cdot tf(t, D)}{k_1 \left((1 - b) + b \frac{dl}{avdl} \right) + tf(t, D)}$$

Juga dengan varian BM25 yang lainnya

Term-at-a-Time (TaaT)

SCORE(q):

1 float $Scores[N] = 0$

2

3 **for each** query term t **do**

4 calculate $w(t,q)$ and fetch postings list for t

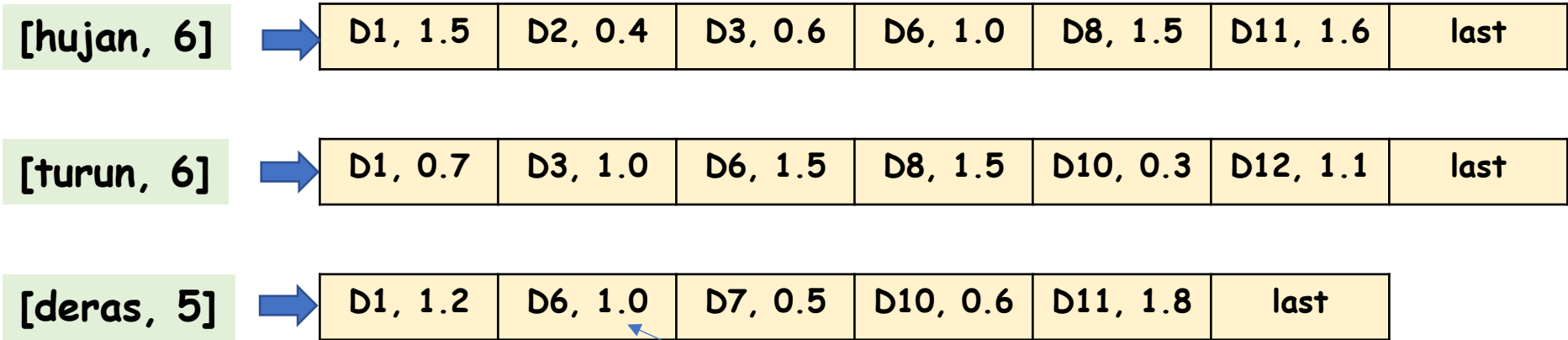
5 **for each** pair($d, tf(t,d)$) in postings list **do**

6 $Scores[d] += w(t,d) \times w(t,q)$

7

8 **return** Top K components of $Scores[]$

Query: hujan turun deras

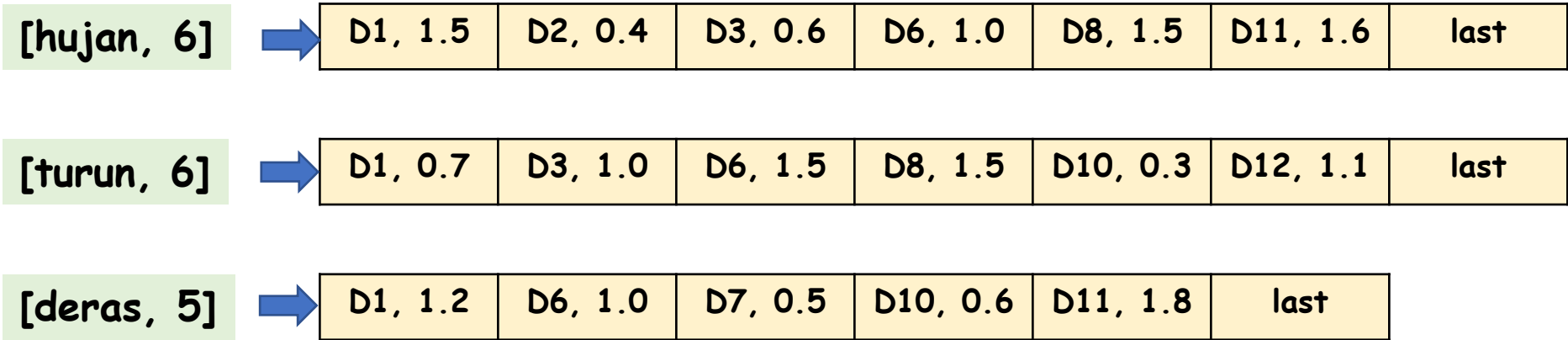


Misal, score $w(t,d)$ sudah dihitung (bisa dengan TF-IDF, BM25 atau yang lainnya). Ini hanya untuk kemudahan saat visualisasi. Kenyataannya, nilai ini bukan sesuatu yang pre-computed dan disimpan di postings lists.

Yang pre-computed dan simpan biasanya adalah $tf(t,d)$ yang berupa integer.

Query: hujan turun deras

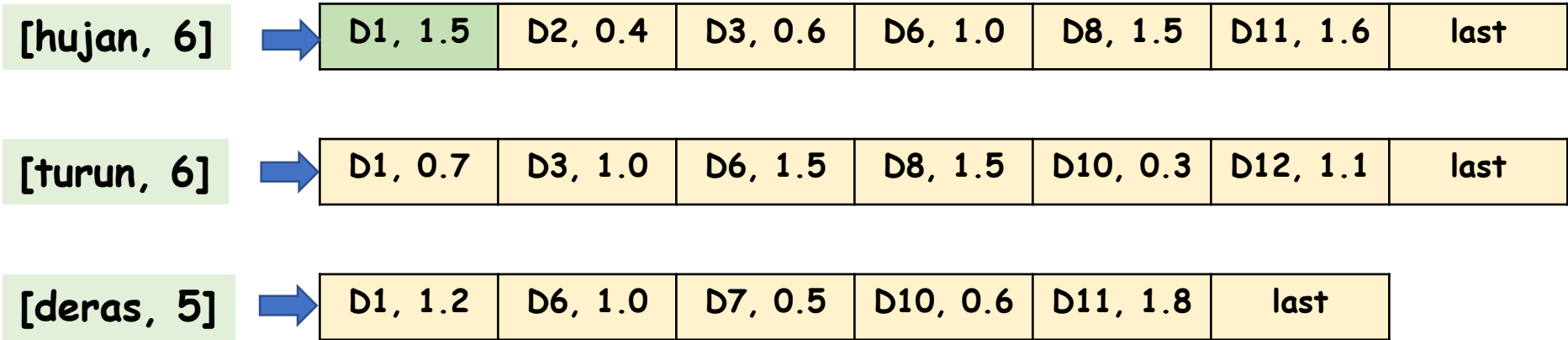
TaaT



Scores = []

Query: hujan turun deras

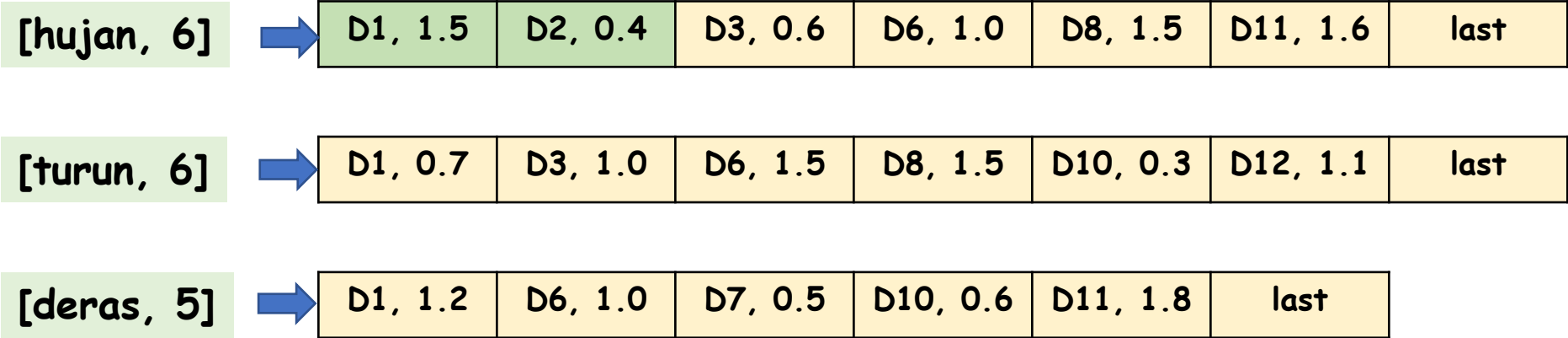
TaaT



Scores = [(D1, 1.5)]

Query: hujan turun deras

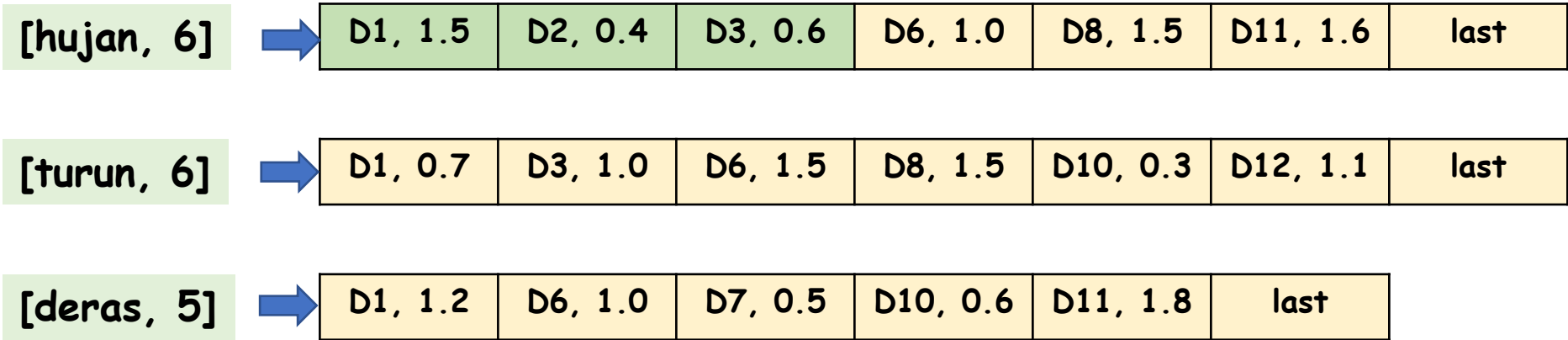
TaaT



Scores = [(D1, 1.5), (D2, 0.4),]

Query: hujan turun deras

TaaT



Scores = [(D1, 1.5), (D2, 0.4), (D3, 0.6)]

Query: hujan turun deras

TaaT

[hujan, 6]	➡	D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
------------	---	---------	---------	---------	---------	---------	----------	------

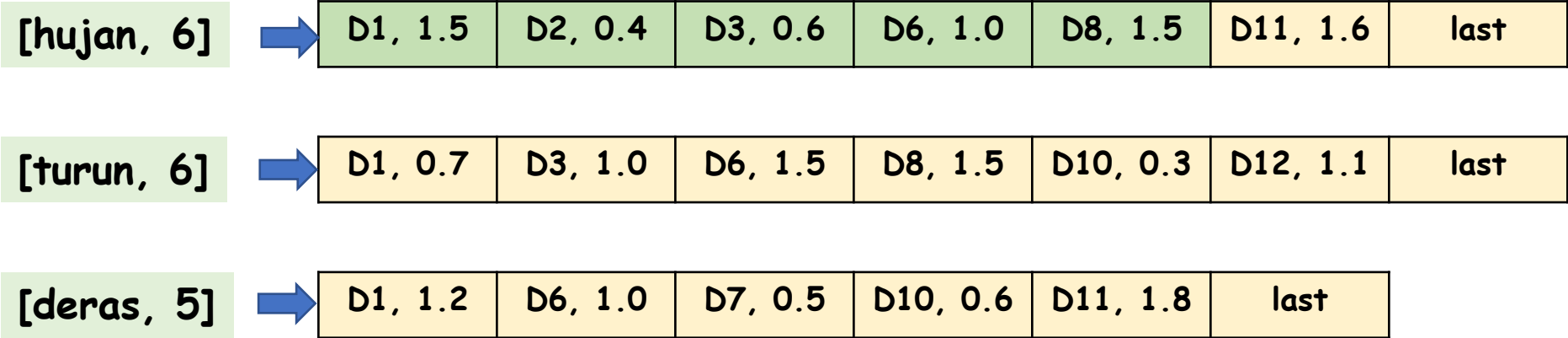
[turun, 6]	➡	D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
------------	---	---------	---------	---------	---------	----------	----------	------

[deras, 5]	➡	D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
------------	---	---------	---------	---------	----------	----------	------

Scores = [(D1, 1.5), (D2, 0.4), (D3, 0.6), (D6, 1.0)]

Query: hujan turun deras

TaaT



Scores = [(D1, 1.0), (D2, 0.4), (D3, 0.6), (D6, 1.0), (D8, 1.5)]

Query: hujan turun deras

TaaT

[hujan, 6]	➡	D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
------------	---	---------	---------	---------	---------	---------	----------	------

[turun, 6]	➡	D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
------------	---	---------	---------	---------	---------	----------	----------	------

[deras, 5]	➡	D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
------------	---	---------	---------	---------	----------	----------	------

Scores = [(D1, 1.5), (D2, 0.4), (D3, 0.6), (D6, 1.0), (D8, 1.5), (D11, 1.6)]

Query: hujan turun deras

TaaT

[hujan, 6]	➡	D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
------------	---	---------	---------	---------	---------	---------	----------	------

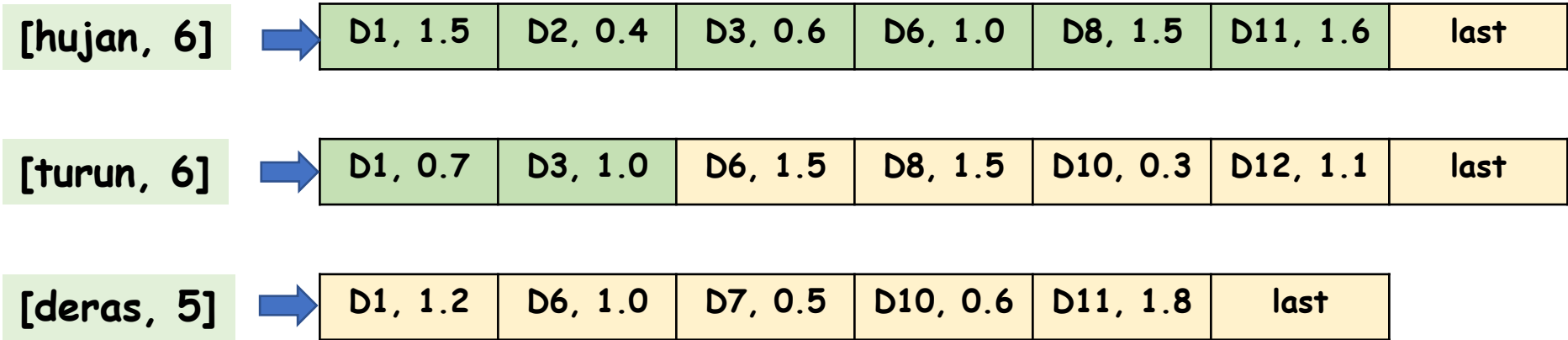
[turun, 6]	➡	D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
------------	---	---------	---------	---------	---------	----------	----------	------

[deras, 5]	➡	D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
------------	---	---------	---------	---------	----------	----------	------

Scores = [(D1, 2.2), (D2, 0.4), (D3, 0.6), (D6, 1.0), (D8, 1.5), (D11, 1.6)]

Query: hujan turun deras

TaaT



Scores = [(D1, 2.2), (D2, 0.4), (D3, 1.6), (D6, 1.0), (D8, 1.5), (D11, 1.6)]

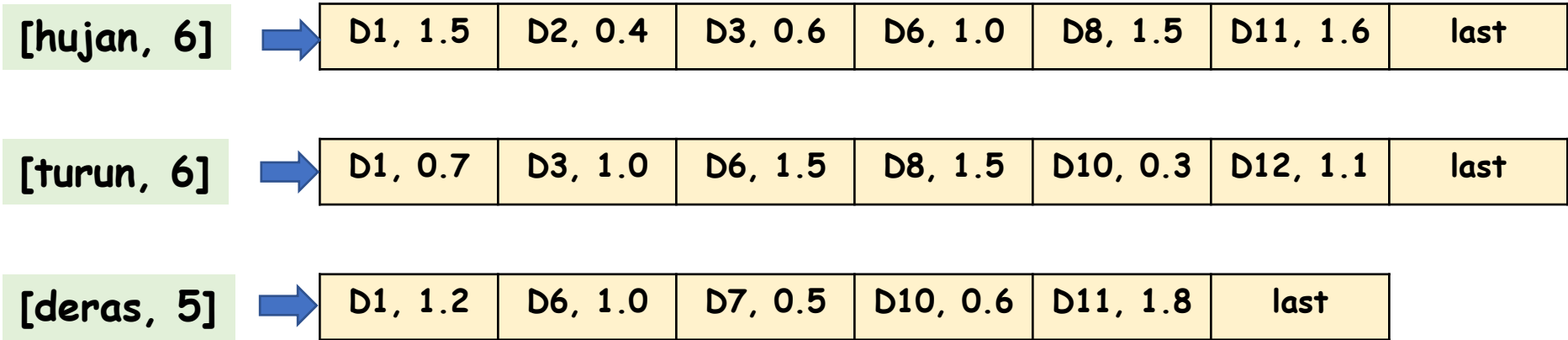
Dan, seterusnya, silakan lanjutkan sendiri ...
Jika score semua dokumen sudah dihitung, gunakan **HEAP** untuk cari **Top-K**

Document-at-a-Time (DaaT)

- Semua postings lists yang terkait query diproses secara paralel.
- Score dari sebuah dokumen dihitung secara penuh (fully evaluated) sebelum berpindah ke dokumen yang lain.
- Di setiap list ada pointer ke current Doc ID. Selalu majukan pointer pada list dengan current Doc ID paling kecil.

Query: hujan turun deras

DaaT

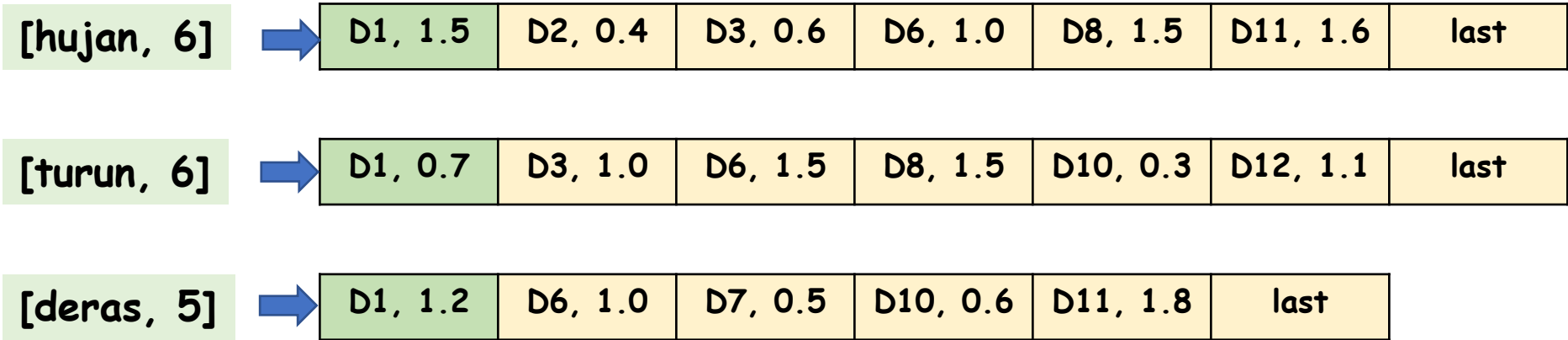


Scores = []

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 1

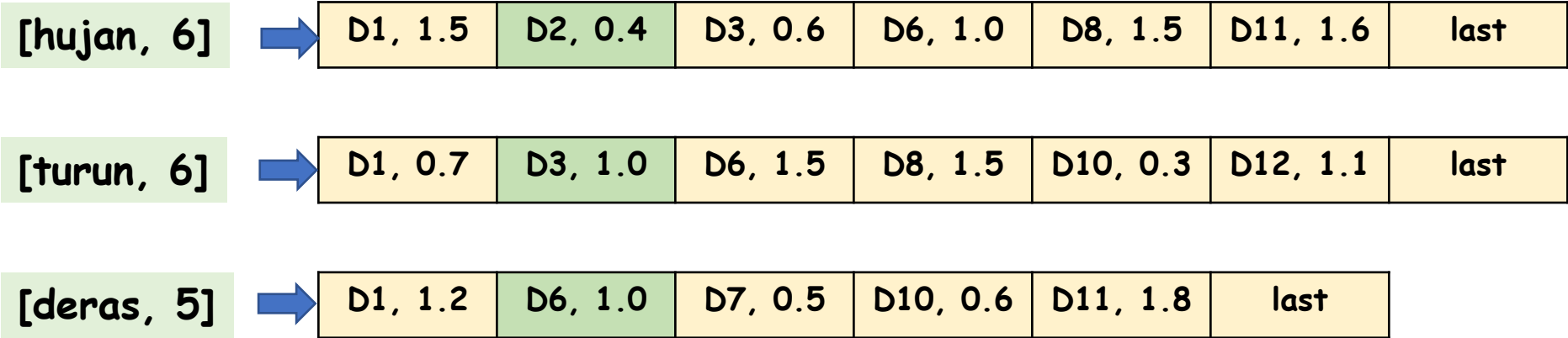


Scores = [(D1, 3.4),]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 2

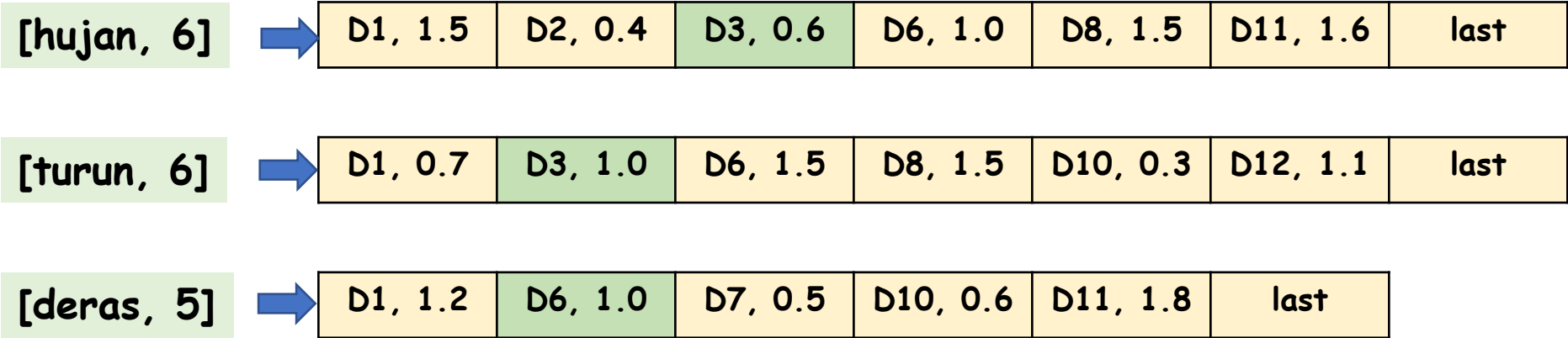


Scores = [(D1, 3.4), (D2, 0.4)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 3

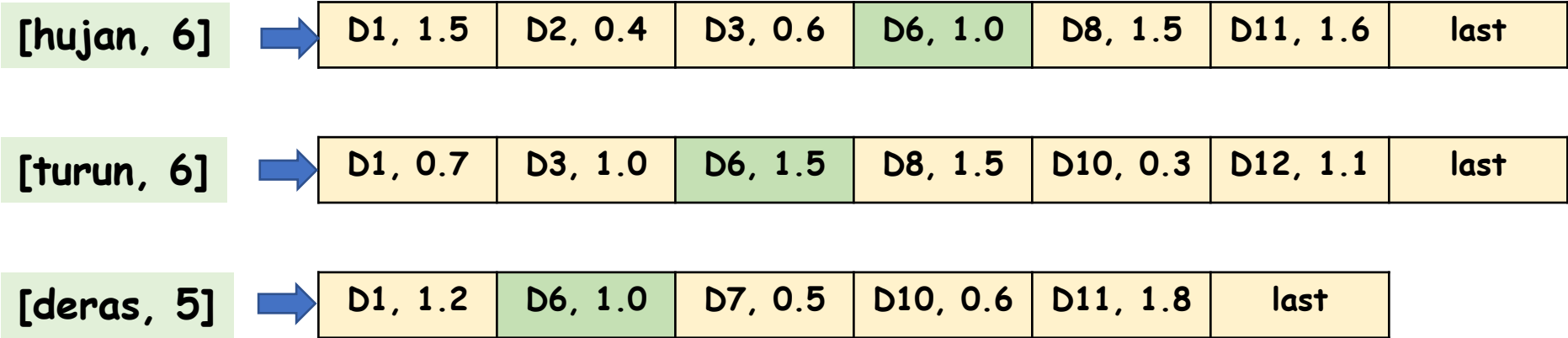


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 4

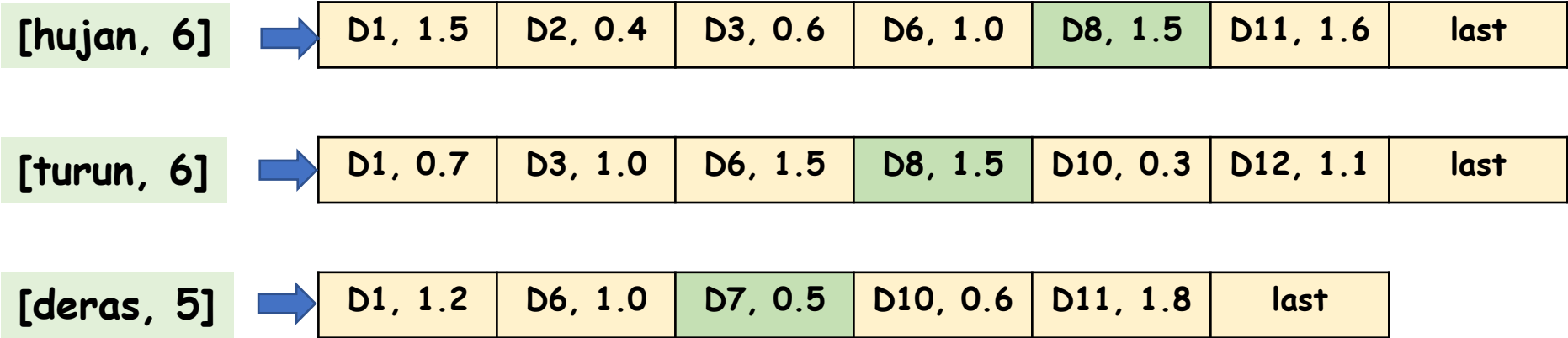


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 5

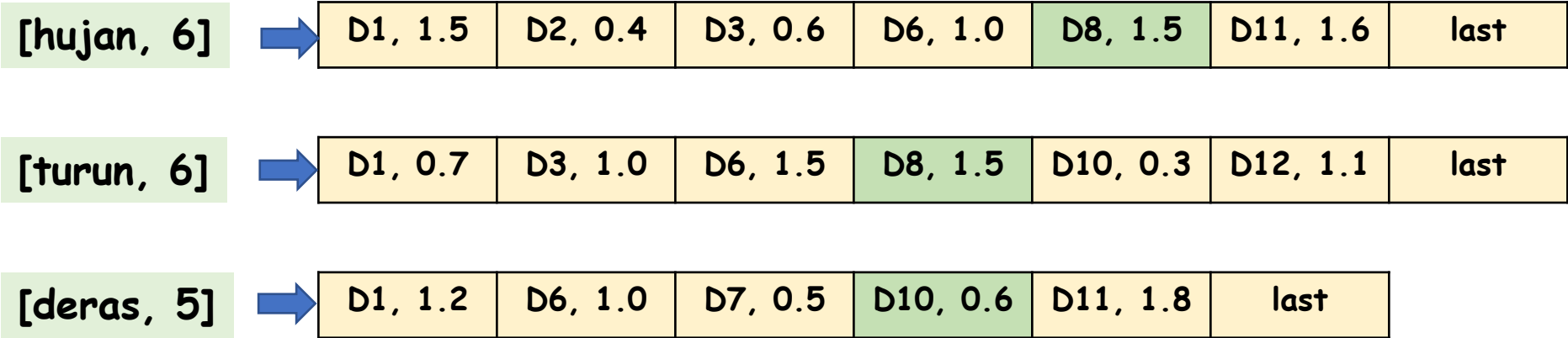


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5), (D7, 0.5)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 6

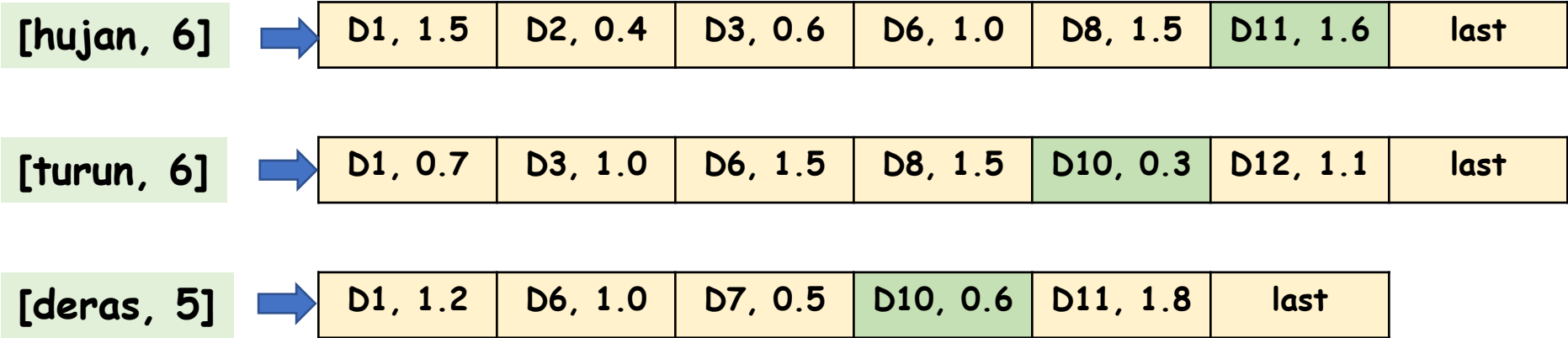


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5), (D7, 0.5), (D8, 3.0)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 7

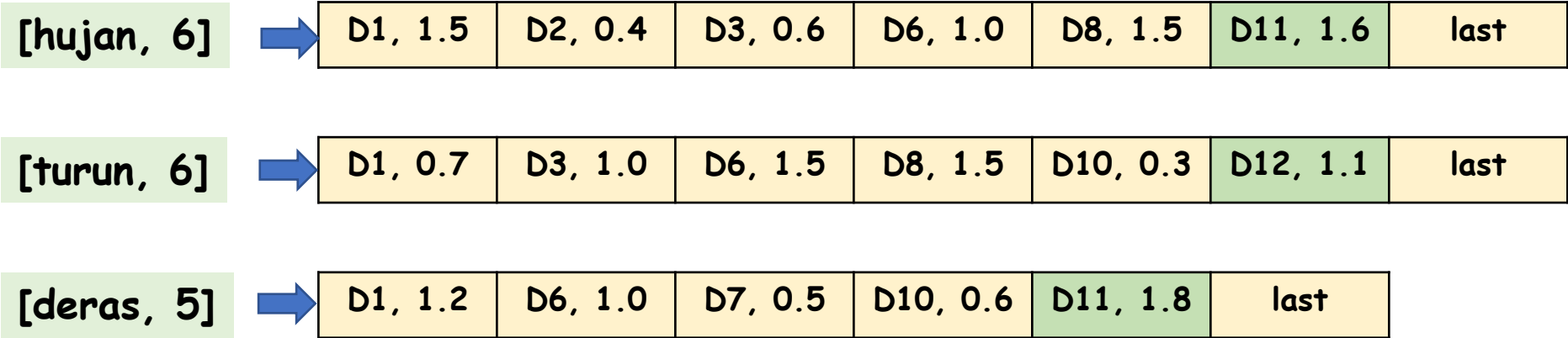


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5), (D7, 0.5), (D8, 3.0), (D10, 0.9)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 8

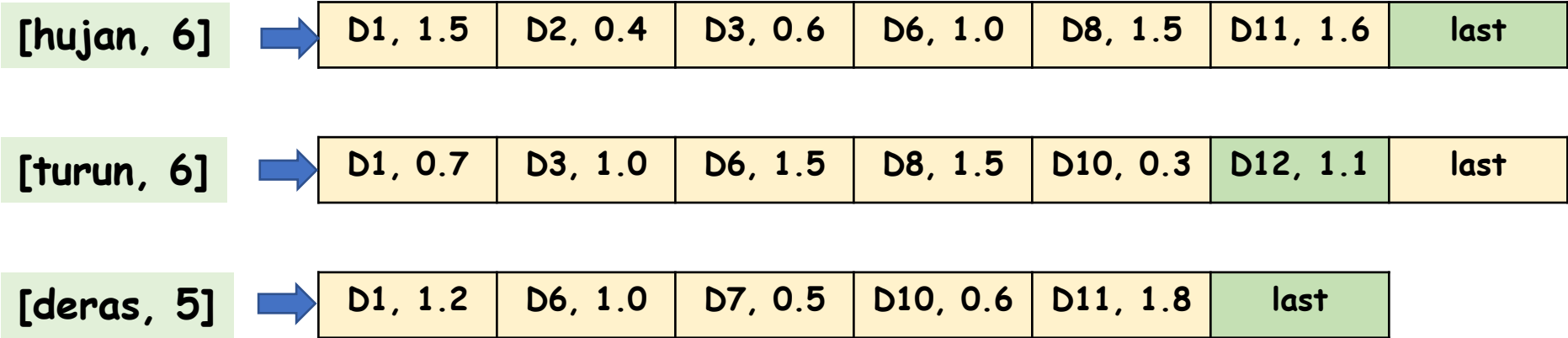


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5), (D7, 0.5), (D8, 3.0), (D10, 0.9), (D11, 3.4)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 9

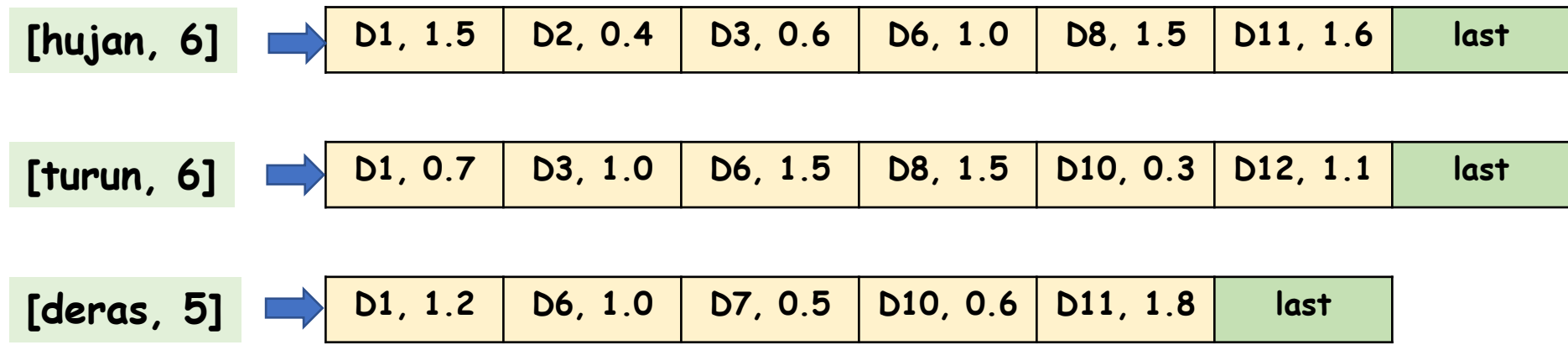


Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5), (D7, 0.5), (D8, 3.0), (D10, 0.9),
(D11, 3.4), (D12, 1.1)]

Query: hujan turun deras

DaaT

Jumlah dokumen yang fully-evaluated: 9



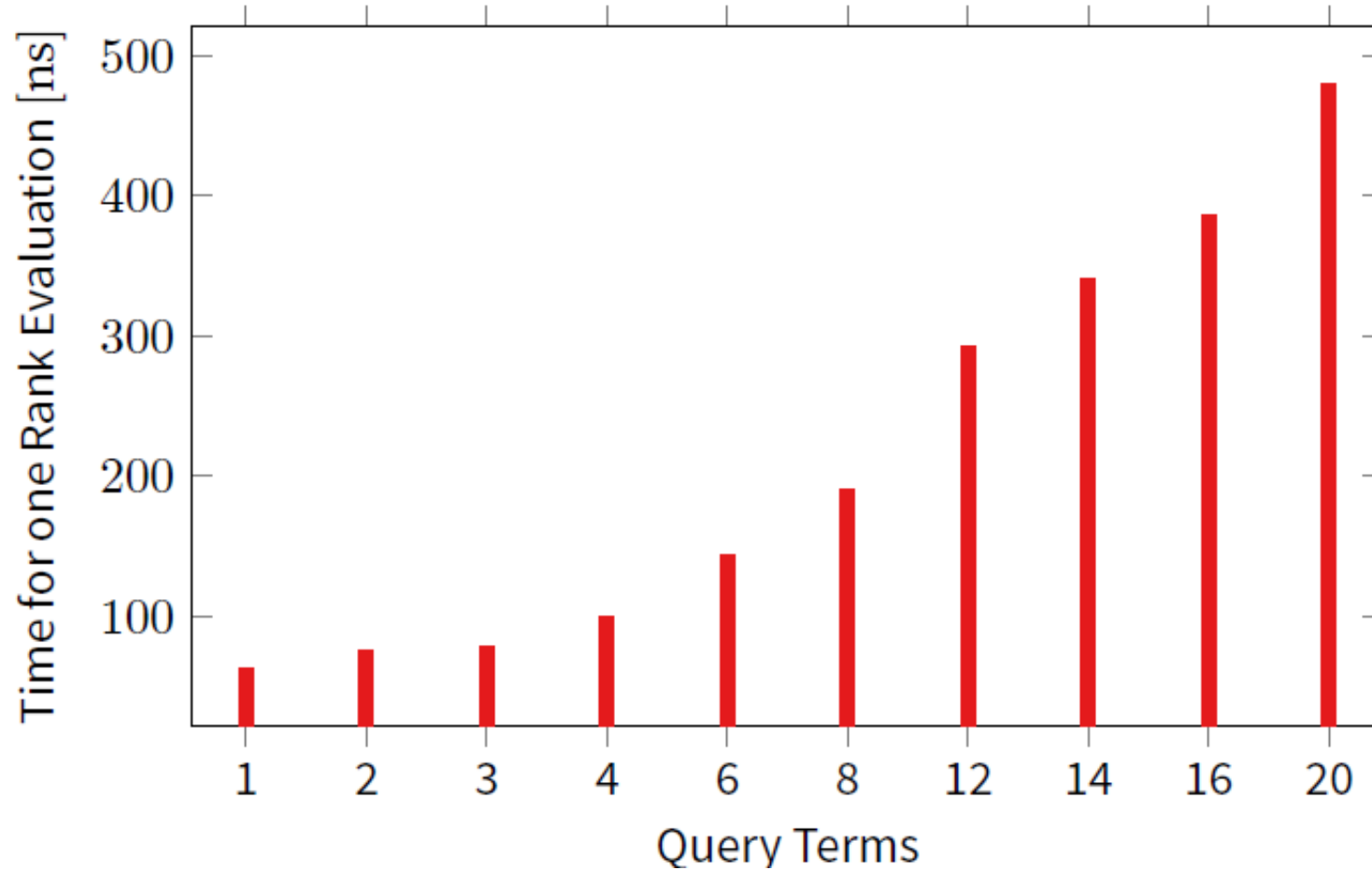
Scores = [(D1, 3.4), (D2, 0.4), (D3, 1.6), (D6, 3.5), (D7, 0.5), (D8, 3.0), (D10, 0.9), (D11, 3.4), (D12, 1.1)]

➡ Top-K HEAP

Masalah dengan DaaT sebelumnya

- Semua dokumen harus dihitung score-nya (fully-evaluated)
- Menghitung score dari sebuah dokumen (misal dengan BM25) sangat mahal! (lihat slide berikutnya)
- Bisakah kita mencari Top-K documents **tanpa harus menghitung score dari semua dokumen** pada postings lists terkait?

BM25 Computation Speed



BM25 Computation Speed

- Menghitung BM25 untuk sebuah pasangan query dan dokumen memerlukan kira-kira **100 nanoseconds**.
- Misal, query "the student" cocok dengan **25M dokumen**. Kita tahu bahwa "the" bisa muncul di hampir setiap dokumen.

$$25.000.000 \times 100 \text{ nanoseconds} = 2.5 \text{ seconds}$$

- Apakah user mau menunggu **2.5 detik** untuk dapat jawaban?

A/B Testing Experiment by Google

- Average revenue per user turun 4% ketika presentasi hasil search diperlambat.
- User sadar atau sensitive terhadap search system yang diperlambat minimal 50 ms.

Bing Search Engine

- Mengurangi waktu proses sebesar 100 ms meningkatkan annual revenue sebesar 0.6%.
- Di tahun 2015, 1% kenaikan revenue per user setara dengan kenaikan sekitar \$10.000.000 per tahun.

Strategi

Tambah server (horizontal scaling)

- Server bertambah -> perlu usaha maintenance lebih, system makin kompleks, power cost yang makin naik.

Ganti spesifikasi hardware yang lebih canggih (vertical scaling)

- Harga mahal & tidak scaling terus-menerus (diminishing returns)

Perbaikan masih bisa di level Software/Program

- Usulkan algoritma query processing yang lebih cepat atau efisien

Solusi: Top-K Retrieval

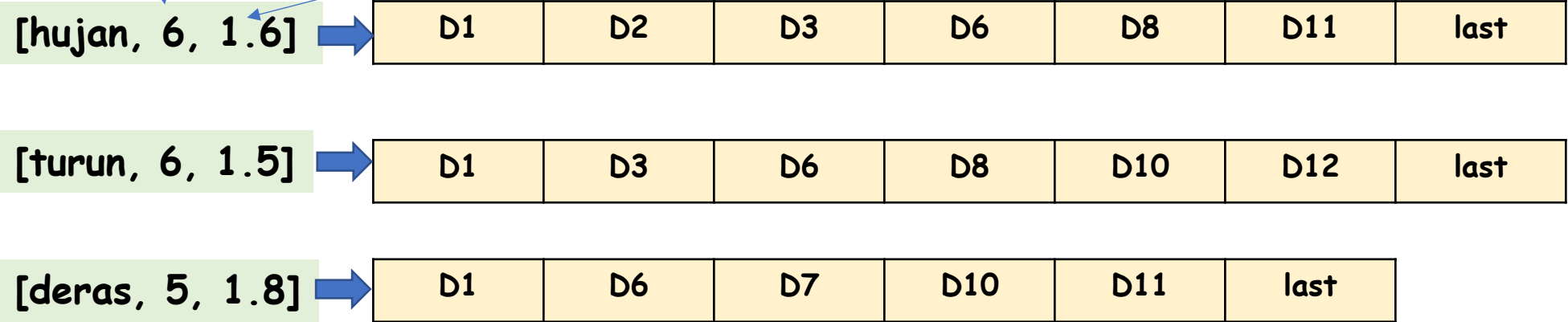
- Secara umum, search engine memang tidak menampilkan semua dokumen yang relevan; biasanya hanya Top-K saja (dengan K biasanya 10).
- For a given query, it's possible to produce a "complete" ranking without scoring all documents in the postings lists.
- Avoid scoring documents that we know will not appear in the Top-K results.

Query: hujan turun deras

Upper Bound Score per Term
Term's maximum contribution to any document score in its postings list

DF

$$UB_t \geq \alpha_t \max(w(t, d_1), w(t, d_2), \dots)$$



WAND Operator

Digunakan untuk menentukan apakah sebuah dokumen punya kemungkinan untuk berada di Top-K.

$$WAND(x_1, UB_1, x_2, UB_2, x_3, UB_3, \dots, x_k, UB_k, \theta)$$

The minimum score in the current Top-K documents

$$= \text{True jika } UB(d, q) = \sum_{1 \leq i \leq k} x_i UB_i \geq \theta$$

x_i bernilai 1 jika term i ada di dokumen; dan 0 jika sebaliknya.

Jika **True**, artinya dokumen tersebut punya chance untuk berada di Top-K.
Jika **False**, sudah tidak ada chance. **Jangan dievaluasi! Skip saja!**

Two Level Approach

Tahap 1 (WAND Condition)

- Cari kumpulan kandidat dokumen dengan menghitung $UB(d,q)$
- Jika $UB(d,q) \geq \text{minimum score}$, ke tahap 2
- Jika tidak, di-skip

Tahap 2 (Full Evaluation)

- Hitung exact full score dari para kandidat yang lolos tahap 1
- Periksa apakah bisa mengalahkan minimum score

Tahap Akhir (Top-K Heap)

TOP-1 :
Threshold :
Pivot :
curDoc :

[hujan, 6, 1.6]	➡	D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
-----------------	---	---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]	➡	D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
-----------------	---	---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]	➡	D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
-----------------	---	---------	---------	---------	----------	----------	------

TOP-1 : []
Threshold : 0
Pivot : 0
curDoc : 0

1. **Function** init(queryTerms)
2. terms \leftarrow queryTerms
3. curDoc \leftarrow 0
4. **for each** t \in terms
5. posting[t] \leftarrow t.iterator.next(0)

Inisialisasi

- Sets the current document (curDoc) to be considered to zero
- For each query term, t, initializes its current posting to be the first posting item in the list

[hujan, 6, 1.6] \rightarrow

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5] \rightarrow

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] \rightarrow

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------


```

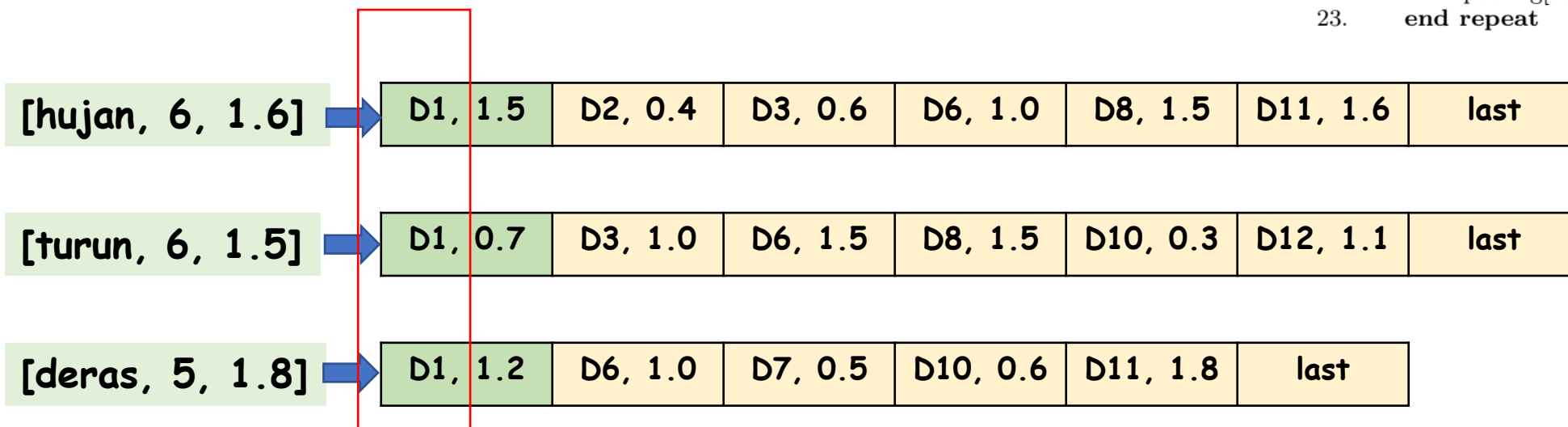
1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
        one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
          to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.      else
20.        /* not enough mass yet on pivot, advance
          one of the preceding terms */
21.        aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.        posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat

```

TOP-1 : []
 Threshold : 0
 Pivot : 0
 curDoc : 0

#documents that are fully evaluated: 0

Sudah terurut berdasarkan Doc ID



```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

TOP-1 : []
 Threshold : 0
 Pivot : 0
 curDoc : 0

#documents that are fully evaluated: 0

hujan adalah pivot term

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

TOP-1 : []
 Threshold : 0
 Pivot : 1
 curDoc : 0

#documents that are fully evaluated: 0

D1 adalah pivot

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

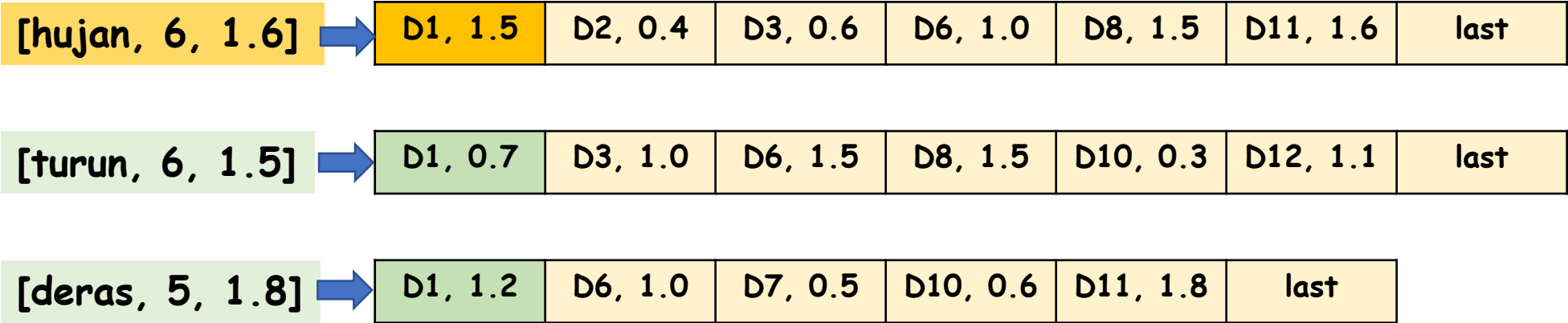
```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
        $UB \geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 1
 curDoc : 1 #documents that are fully evaluated: 1

D1 dikembalikan, dan dihitung score full-nya!
 Lalu disimpan di daftar TOP-1 kita, dan jangan lupa update nilai threshold terbaru

WAND(D1) = WAND(hujan, 1.6, turun, 1.5, deras, 1.8; threshold) = True



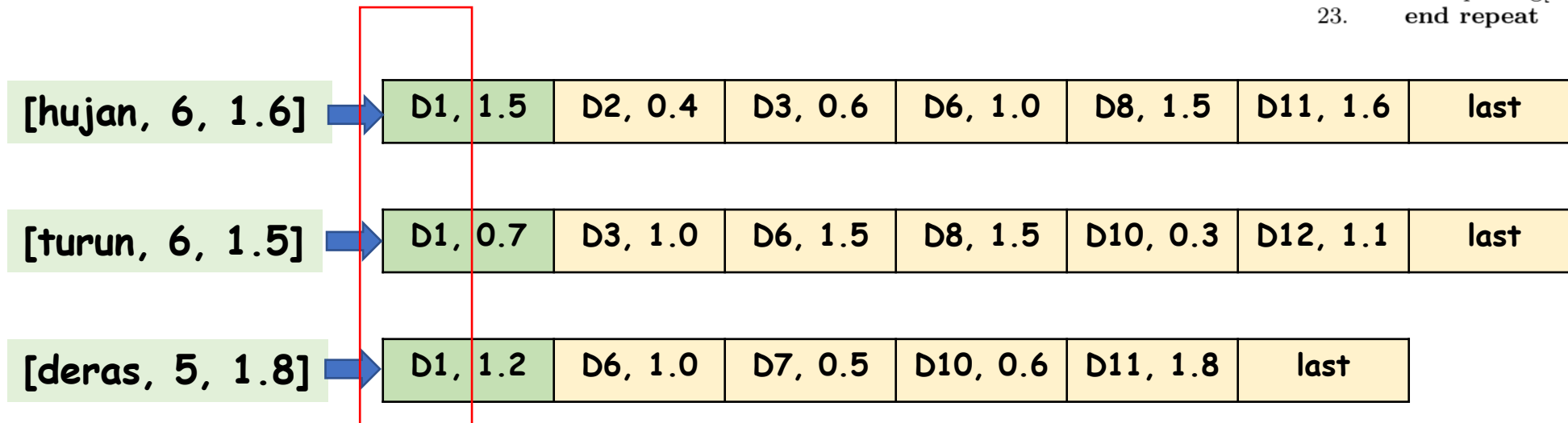
```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.      else
20.        /* not enough mass yet on pivot, advance one of the preceding terms */
21.        aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.        posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat
  
```

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 1
 curDoc : 1

#documents that are fully evaluated: 1

Mulai dari awal lagi, dan lakukan sort terhadap Doc ID



```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 1
 curDoc : 1

#documents that are fully evaluated: 1

“deras” adalah pivot term, dengan D1 adalah pivot

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 1
 curDoc : 1 #documents that are fully evaluated: 1

Advance one of the preceding terms

pickTerm() → banyak cara untuk memilih; yang paling simple adalah "pilih yang pertama"

Pointer dari term yang terpilih dimajukan hingga bertemu DID pertama yang $\geq 1 + 1 = 2$

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```


TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 1
 curDoc : 1

#documents that are fully evaluated: 1

Mulai lagi dari awal. Sort berdasarkan Doc ID

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
Threshold : 3.4
Pivot : 1
curDoc : 1

#documents that are fully evaluated: 1

Mulai lagi dari awal. Sort berdasarkan Doc ID

```
1. Function next( $\theta$ )
2. repeat
3.   /* Sort the terms in non decreasing order of
   DID */
4.   sort(terms, posting)
5.   /* Find pivot term - the first one with accumulated
    $UB \geq \theta$  */
6.   pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.   if (pTerm = null) return (NoMoreDocs)
8.   pivot  $\leftarrow$  posting[pTerm].DID
9.   if (pivot = lastID) return (NoMoreDocs)
10.  if (pivot  $\leq$  curDoc)
11.    /* pivot has already been considered, advance
    one of the preceding terms */
12.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.    posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.  else /* pivot > curDoc */
15.    if (posting[0].DID = pivot)
16.      /* Success, all terms preceding pTerm belong
      to the pivot */
17.      curDoc  $\leftarrow$  pivot
18.      return (curDoc, posting)
19.  else
20.    /* not enough mass yet on pivot, advance
    one of the preceding terms */
21.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.    posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Mulai lagi dari awal. Sort berdasarkan Doc ID

$$1.5 + 1.8 + 1.6 \geq 3.4$$

Pivot term = hujan, pivot = D2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

TOP-1 : [(D1,3.4)]
Threshold : 3.4
Pivot : 2
curDoc : 1

#documents that are fully evaluated: 1

Not enough mass yet on pivot, advance one of the preceding terms.

pickTerm() → kita pilih yang pertama

```
1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
```

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Not enough mass yet on pivot, advance one of the preceding terms.

pickTerm() → kita pilih yang pertama
 Lalu kita pindahkan ke posisi pertama yang Doc ID \geq pivot

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat
  
```

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Mulai lagi dari awal, lakukan sorting

```

1. Function next( $\theta$ )
2. repeat
3.   /* Sort the terms in non decreasing order of
      DID */
4.   sort(terms, posting)
5.   /* Find pivot term - the first one with accumulated
      UB  $\geq \theta$  */
6.   pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.   if (pTerm = null) return (NoMoreDocs)
8.   pivot  $\leftarrow$  posting[pTerm].DID
9.   if (pivot = lastID) return (NoMoreDocs)
10.  if (pivot  $\leq$  curDoc)
11.    /* pivot has already been considered, advance
       one of the preceding terms */
12.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.    posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.  else /* pivot > curDoc */
15.    if (posting[0].DID = pivot)
16.      /* Success, all terms preceding pTerm belong
         to the pivot */
17.      curDoc  $\leftarrow$  pivot
18.      return (curDoc, posting)
19.  else
20.    /* not enough mass yet on pivot, advance
       one of the preceding terms */
21.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.    posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Mulai lagi dari awal, lakukan sorting

```

1. Function next( $\theta$ )
2. repeat
3.   /* Sort the terms in non decreasing order of
      DID */
4.   sort(terms, posting)
5.   /* Find pivot term - the first one with accumulated
      UB  $\geq \theta$  */
6.   pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.   if (pTerm = null) return (NoMoreDocs)
8.   pivot  $\leftarrow$  posting[pTerm].DID
9.   if (pivot = lastID) return (NoMoreDocs)
10.  if (pivot  $\leq$  curDoc)
11.    /* pivot has already been considered, advance
       one of the preceding terms */
12.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.    posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.  else /* pivot > curDoc */
15.    if (posting[0].DID = pivot)
16.      /* Success, all terms preceding pTerm belong
         to the pivot */
17.      curDoc  $\leftarrow$  pivot
18.      return (curDoc, posting)
19.  else
20.    /* not enough mass yet on pivot, advance
       one of the preceding terms */
21.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.    posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[deras, 5, 1.8]

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Pivot term = hujan dengan pivot = D2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Kita majukan pointer di deras ke Doc ID pertama yang \geq pivot.

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat
  
```

[deras, 5, 1.8]

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Kita majukan pointer di deras ke Doc ID pertama yang \geq pivot.

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat
  
```

[deras, 5, 1.8]

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 2
 curDoc : 1

#documents that are fully evaluated: 1

Sort berdasarkan Doc ID

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
        $UB \geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

Pivot term = deras, dengan pivot = D6

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

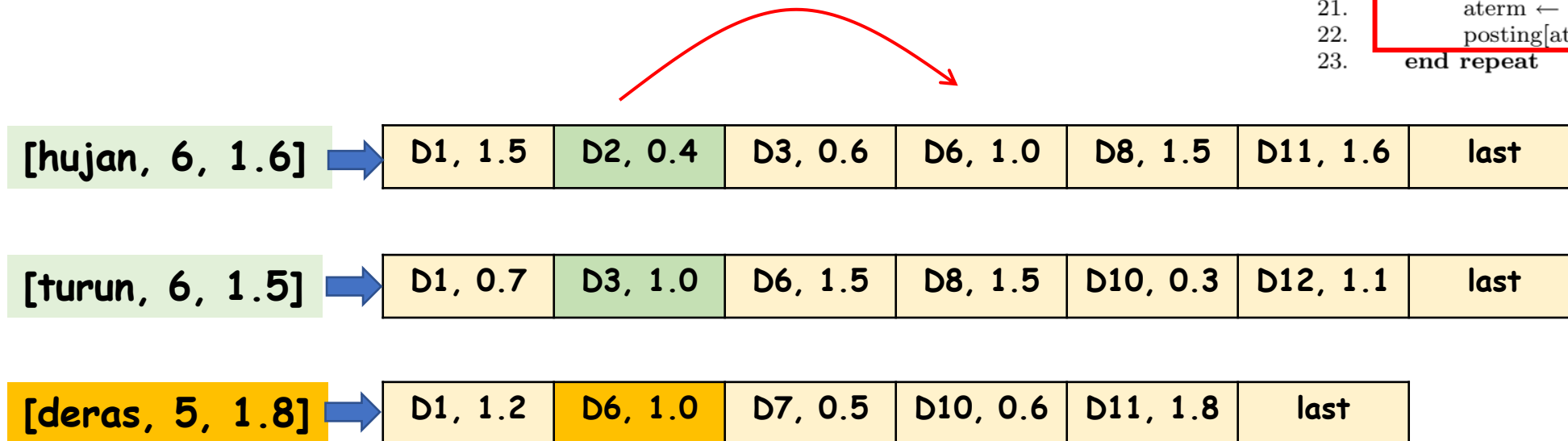
#documents that are fully evaluated: 1

Not enough mass yet on pivot, advance one of the preceding terms.

pickTerm() → kita pilih yang pertama
 Lalu kita pindahkan ke posisi pertama yang Doc ID >= pivot

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```



TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

D3 di-skip karena dijamin tidak akan memberikan score yang > 3.4 . Mengapa? 😊

Catat bahwa posting list terakhir berada pada posisi D6. Seandainya dua posting list pertama berada pada posisi D3, paling tinggi hanya $1.6 + 1.5 = 3.1$ yang < 3.4 .

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

Sort dokumen berdasarkan Doc ID

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

Pivot term = deras, dengan pivot = 6

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

Majukan Doc ID di term turun ke Doc ID pertama yang \geq pivot

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
Threshold : 3.4
Pivot : 6
curDoc : 1

#documents that are fully evaluated: 1

Majukan Doc ID di term turun ke Doc ID pertama yang \geq pivot

```
1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
```

[turun, 6, 1.5]



D1, 0.7

D3, 1.0

D6, 1.5

D8, 1.5

D10, 0.3

D12, 1.1

last

[hujan, 6, 1.6]



D1, 1.5

D2, 0.4

D3, 0.6

D6, 1.0

D8, 1.5

D11, 1.6

last

[deras, 5, 1.8]



D1, 1.2

D6, 1.0

D7, 0.5

D10, 0.6

D11, 1.8

last

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D1,3.4)]
 Threshold : 3.4
 Pivot : 6
 curDoc : 1

#documents that are fully evaluated: 1

Artinya pivot ID muncul di semua preceding lists!

Dokumen pivot dievaluasi full = 1.5 + 1.0 + 1.0 = 3.5

3.5 > 3.4!

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7

D3, 1.0

D6, 1.5

D8, 1.5

D10, 0.3

D12, 1.1

last

[hujan, 6, 1.6]



D1, 1.5

D2, 0.4

D3, 0.6

D6, 1.0

D8, 1.5

D11, 1.6

last

[deras, 5, 1.8]



D1, 1.2

D6, 1.0

D7, 0.5

D10, 0.6

D11, 1.8

last

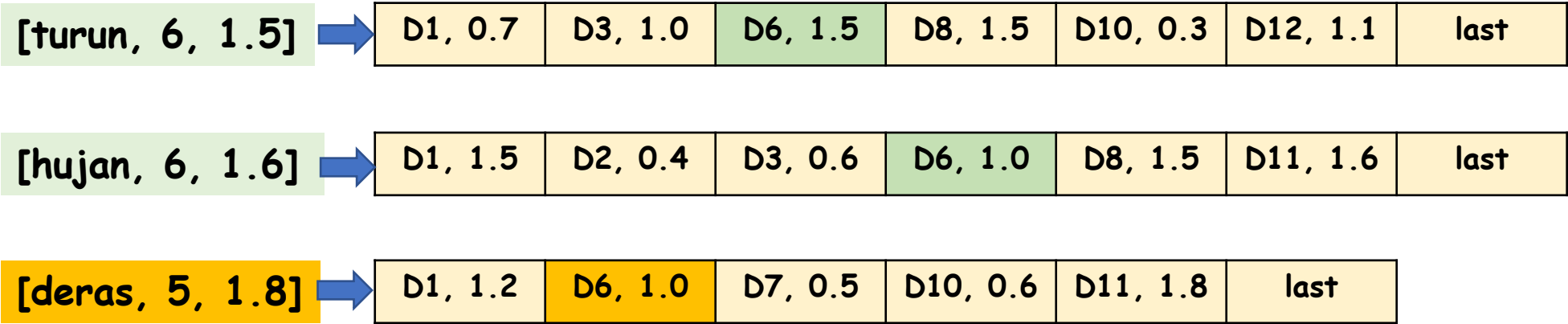
TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 6
 curDoc : 6 #documents that are fully evaluated: 2

Artinya pivot ID muncul di semua preceding lists!

Dokumen pivot dievaluasi full = 1.5 + 1.0 + 1.0 = 3.5

3.5 > 3.4!

WAND(D6) = WAND(hujan, 1.6, turun, 1.5, deras, 1.8; threshold) = True



```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 6
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7

D3, 1.0

D6, 1.5

D8, 1.5

D10, 0.3

D12, 1.1

last

[hujan, 6, 1.6]



D1, 1.5

D2, 0.4

D3, 0.6

D6, 1.0

D8, 1.5

D11, 1.6

last

[deras, 5, 1.8]



D1, 1.2

D6, 1.0

D7, 0.5

D10, 0.6

D11, 1.8

last

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 6
 curDoc : 6

#documents that are fully evaluated: 2

Pointer dari term yang terpilih dimajukan hingga bertemu DID pertama yang $\geq 6 + 1 = 7$

Yaitu D8

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[turun, 6, 1.5]



D1, 0.7

D3, 1.0

D6, 1.5

D8, 1.5

D10, 0.3

D12, 1.1

last

[hujan, 6, 1.6]



D1, 1.5

D2, 0.4

D3, 0.6

D6, 1.0

D8, 1.5

D11, 1.6

last

[deras, 5, 1.8]



D1, 1.2

D6, 1.0

D7, 0.5

D10, 0.6

D11, 1.8

last

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 8
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5

D2, 0.4

D3, 0.6

D6, 1.0

D8, 1.5

D11, 1.6

last

[deras, 5, 1.8]



D1, 1.2

D6, 1.0

D7, 0.5

D10, 0.6

D11, 1.8

last

[turun, 6, 1.5]



D1, 0.7

D3, 1.0

D6, 1.5

D8, 1.5

D10, 0.3

D12, 1.1

last

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 8
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.    end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 8
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D6, 3.5)]
 Threshold : 3.5
 Pivot : 8
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[deras, 5, 1.8] → D1, 1.2 D6, 1.0 D7, 0.5 D10, 0.6 D11, 1.8 last

[hujan, 6, 1.6] → D1, 1.5 D2, 0.4 D3, 0.6 D6, 1.0 D8, 1.5 D11, 1.6 last

[turun, 6, 1.5] → D1, 0.7 D3, 1.0 D6, 1.5 D8, 1.5 D10, 0.3 D12, 1.1 last

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 10
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2. repeat
3.   /* Sort the terms in non decreasing order of
      DID */
4.   sort(terms, posting)
5.   /* Find pivot term - the first one with accumulated
      UB  $\geq \theta$  */
6.   pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.   if (pTerm = null) return (NoMoreDocs)
8.   pivot  $\leftarrow$  posting[pTerm].DID
9.   if (pivot = lastID) return (NoMoreDocs)
10.  if (pivot  $\leq$  curDoc)
11.    /* pivot has already been considered, advance
       one of the preceding terms */
12.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.    posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.  else /* pivot > curDoc */
15.    if (posting[0].DID = pivot)
16.      /* Success, all terms preceding pTerm belong
         to the pivot */
17.      curDoc  $\leftarrow$  pivot
18.      return (curDoc, posting)
19.  else
20.    /* not enough mass yet on pivot, advance
       one of the preceding terms */
21.    aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.    posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5

D2, 0.4

D3, 0.6

D6, 1.0

D8, 1.5

D11, 1.6

last

[turun, 6, 1.5]



D1, 0.7

D3, 1.0

D6, 1.5

D8, 1.5

D10, 0.3

D12, 1.1

last

[deras, 5, 1.8]



D1, 1.2

D6, 1.0

D7, 0.5

D10, 0.6

D11, 1.8

last

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 10
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8]



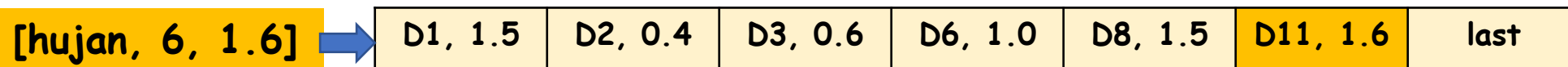
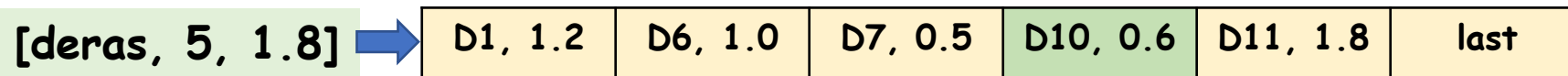
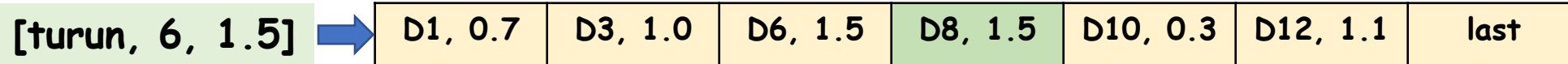
D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 11
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

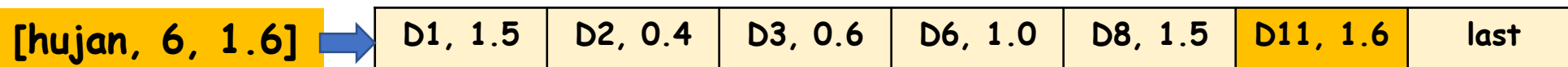
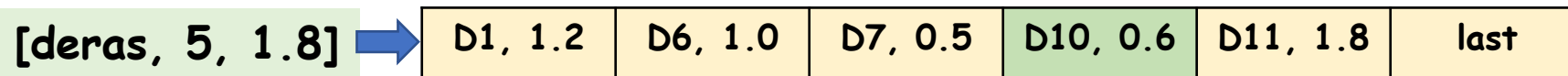
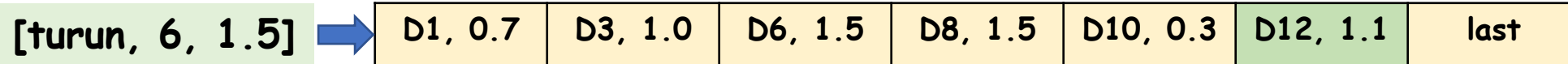


TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 11
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```



TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : 12
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[deras, 5, 1.8]



D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

[hujan, 6, 1.6]



D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5]



D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

TOP-1 : [(D6, 3.5)]
 Threshold : 3.5
 Pivot : 12
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[deras, 5, 1.8] → D1, 1.2 D6, 1.0 D7, 0.5 D10, 0.6 D11, 1.8 last

[hujan, 6, 1.6] → D1, 1.5 D2, 0.4 D3, 0.6 D6, 1.0 D8, 1.5 D11, 1.6 last

[turun, 6, 1.5] → D1, 0.7 D3, 1.0 D6, 1.5 D8, 1.5 D10, 0.3 D12, 1.1 last

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : last
 curDoc : 6

#documents that are fully evaluated: 2

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-1 : [(D6,3.5)]
 Threshold : 3.5
 Pivot : last
 curDoc : 6

#documents that are fully evaluated: 2

STOP !

```

1. Function next( $\theta$ )
2.   repeat
3.     /* Sort the terms in non decreasing order of
       DID */
4.     sort(terms, posting)
5.     /* Find pivot term - the first one with accumulated
       UB  $\geq \theta$  */
6.     pTerm  $\leftarrow$  findPivotTerm(terms,  $\theta$ )
7.     if (pTerm = null) return (NoMoreDocs)
8.     pivot  $\leftarrow$  posting[pTerm].DID
9.     if (pivot = lastID) return (NoMoreDocs)
10.    if (pivot  $\leq$  curDoc)
11.      /* pivot has already been considered, advance
         one of the preceding terms */
12.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
13.      posting[aterm]  $\leftarrow$  aterm.iterator.next(curDoc+1)
14.    else /* pivot > curDoc */
15.      if (posting[0].DID = pivot)
16.        /* Success, all terms preceding pTerm belong
           to the pivot */
17.        curDoc  $\leftarrow$  pivot
18.        return (curDoc, posting)
19.    else
20.      /* not enough mass yet on pivot, advance
         one of the preceding terms */
21.      aterm  $\leftarrow$  pickTerm(terms[0..pTerm])
22.      posting[aterm]  $\leftarrow$  aterm.iterator.next(pivot)
23.  end repeat
  
```

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------

TOP-2 :
Threshold :
Pivot :
curDoc :

Latihan

Lakukan hal yang sama dengan sebelumnya. Namun sekarang **Top-2** yang diambil!

[hujan, 6, 1.6] →

D1, 1.5	D2, 0.4	D3, 0.6	D6, 1.0	D8, 1.5	D11, 1.6	last
---------	---------	---------	---------	---------	----------	------

[turun, 6, 1.5] →

D1, 0.7	D3, 1.0	D6, 1.5	D8, 1.5	D10, 0.3	D12, 1.1	last
---------	---------	---------	---------	----------	----------	------

[deras, 5, 1.8] →

D1, 1.2	D6, 1.0	D7, 0.5	D10, 0.6	D11, 1.8	last
---------	---------	---------	----------	----------	------