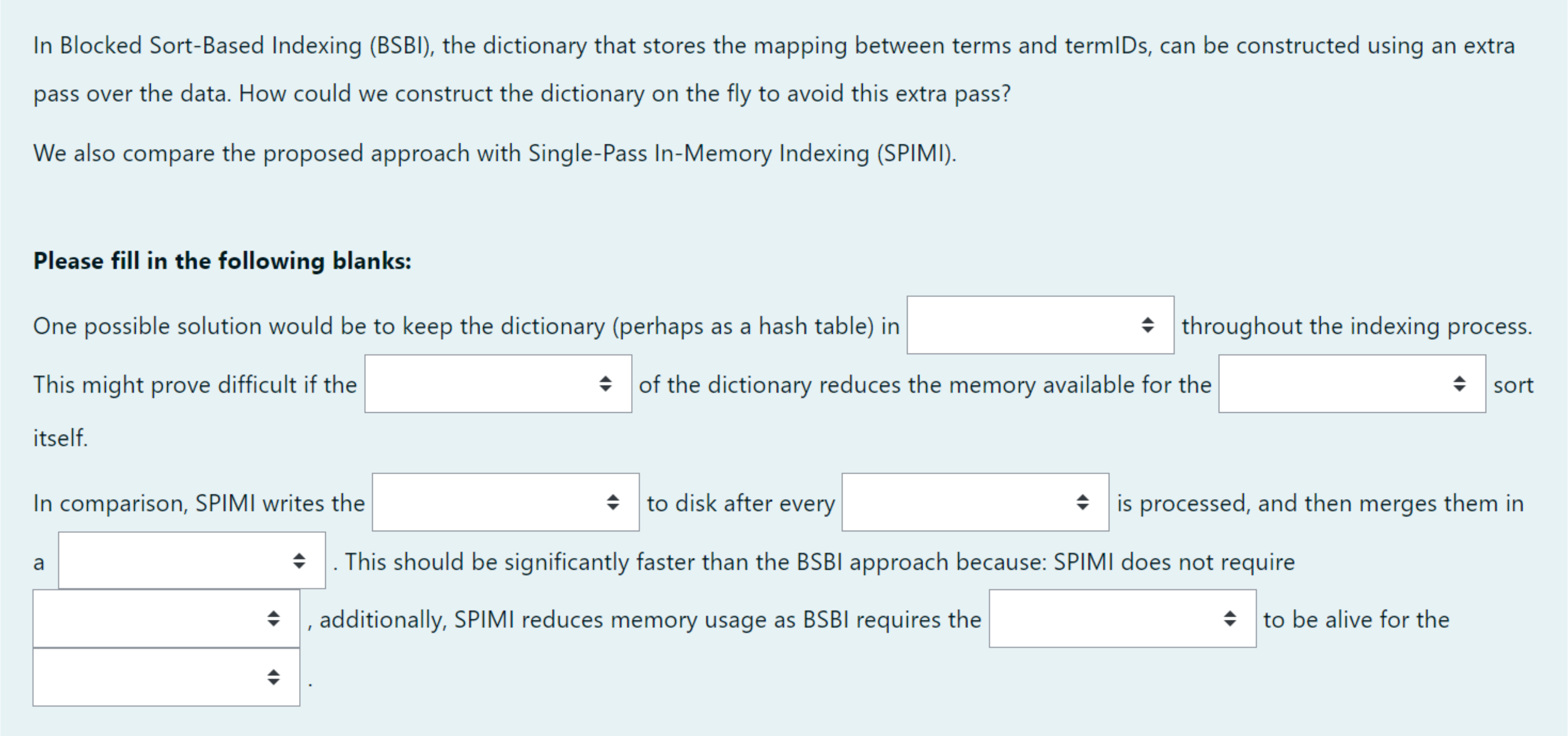
ETH zürich

Information Retrieval Spring 2023

Exercise Session Week 7

Exercise 4: Index construction
Moodle questions: 1



In Blocked Sort-Based Indexing (BSBI), the dictionary that stores the mapping between terms and termIDs, can be constructed using an extra pass over the data. How could we construct the dictionary on the fly to avoid this extra pass?

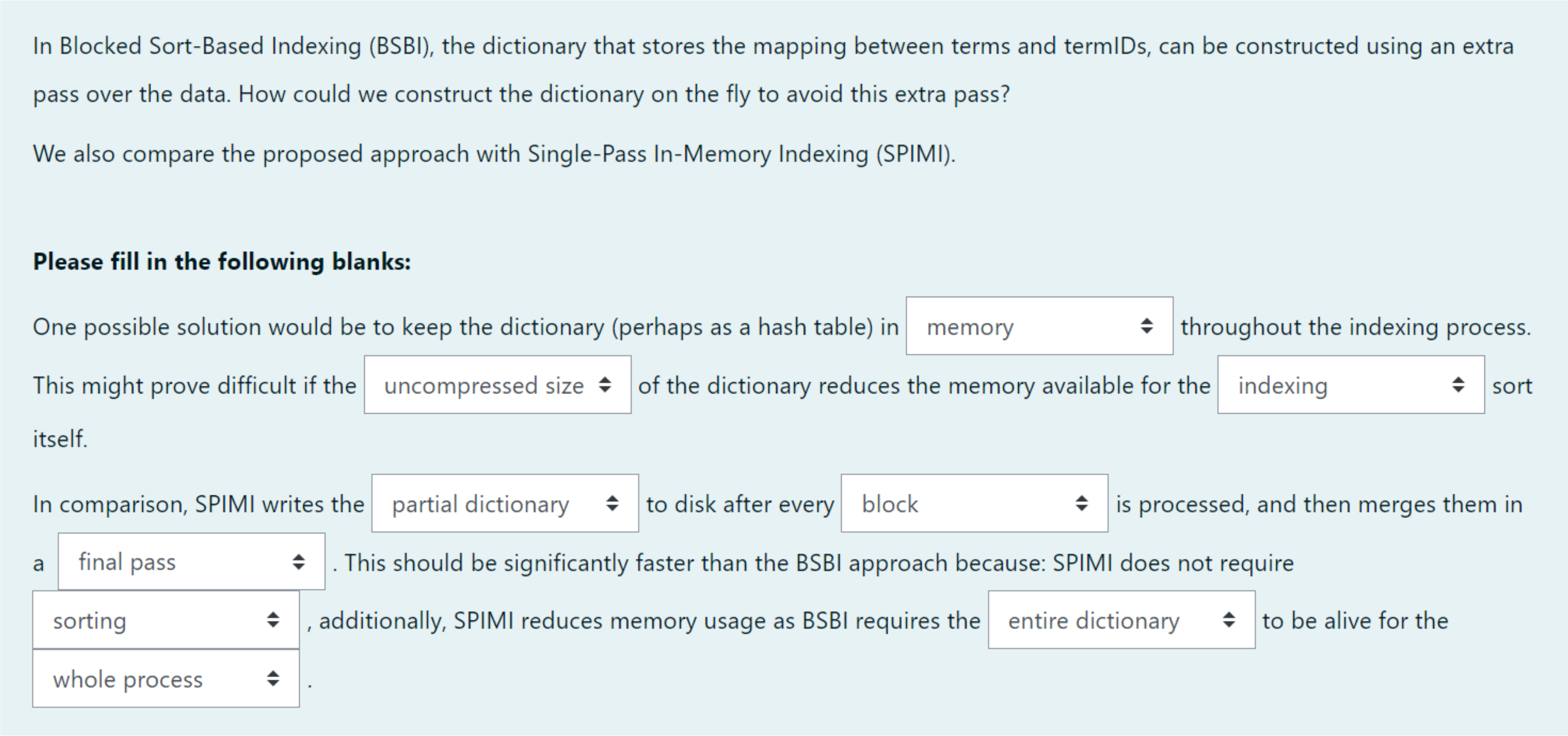
We also compare the proposed approach with Single-Pass In-Memory Indexing (SPIMI).

Please fill in the following blanks:

One possible solution would be to keep the dictionary (perhaps as a hash table) in ⬍ throughout the indexing process. This might prove difficult if the ⬍ of the dictionary reduces the memory available for the ⬍ sort itself.

In comparison, SPIMI writes the ⬍ to disk after every ⬍ is processed, and then merges them in a ⬍ . This should be significantly faster than the BSBI approach because: SPIMI does not require ⬍ , additionally, SPIMI reduces memory usage as BSBI requires the ⬍ to be alive for the ⬍ .

Exercise 4: Index construction
Moodle questions: 1



In Blocked Sort-Based Indexing (BSBI), the dictionary that stores the mapping between terms and termIDs, can be constructed using an extra pass over the data. How could we construct the dictionary on the fly to avoid this extra pass?

We also compare the proposed approach with Single-Pass In-Memory Indexing (SPIMI).

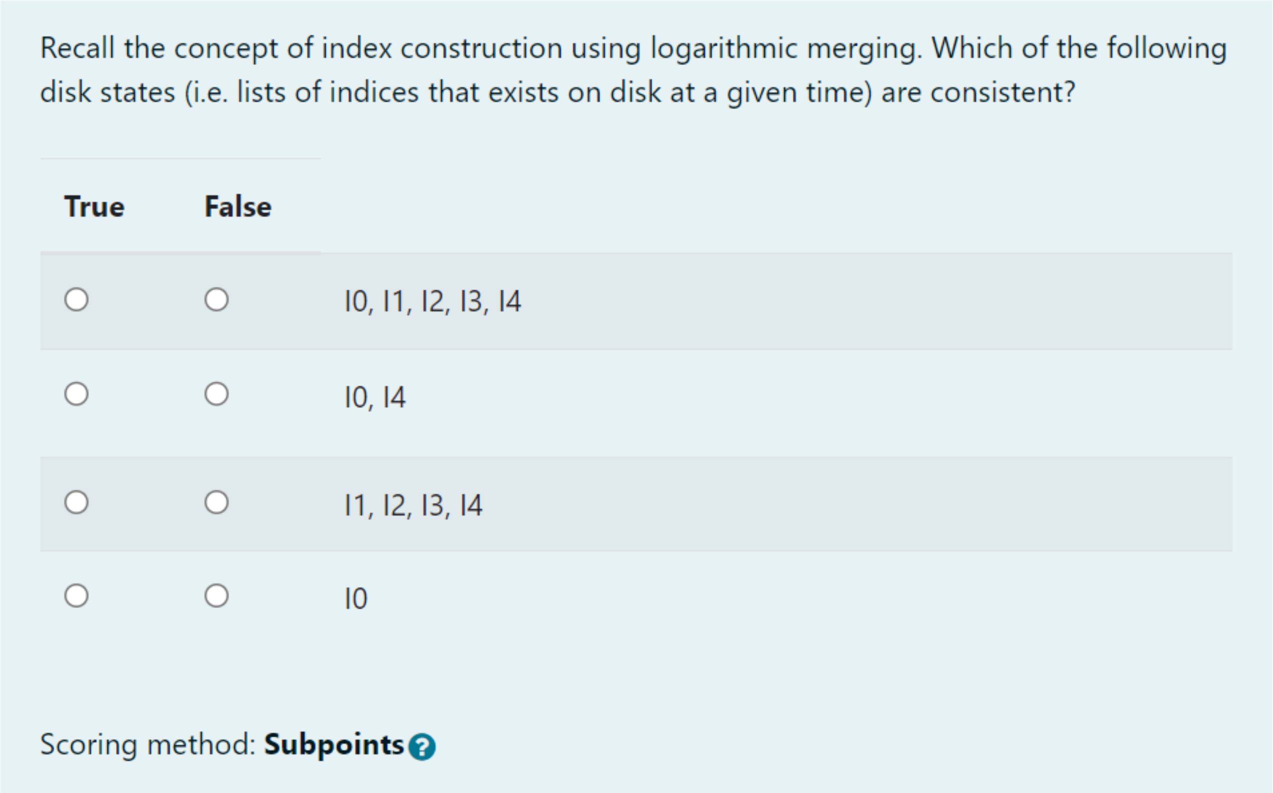
Please fill in the following blanks:

One possible solution would be to keep the dictionary (perhaps as a hash table) in memory throughout the indexing process. This might prove difficult if the uncompressed size of the dictionary reduces the memory available for the indexing sort itself.

In comparison, SPIMI writes the partial dictionary to disk after every block is processed, and then merges them in a final pass . This should be significantly faster than the BSBI approach because: SPIMI does not require sorting , additionally, SPIMI reduces memory usage as BSBI requires the entire dictionary to be alive for the whole process .

ETH zürich

Exercise 4: Index construction
Moodle questions: 2

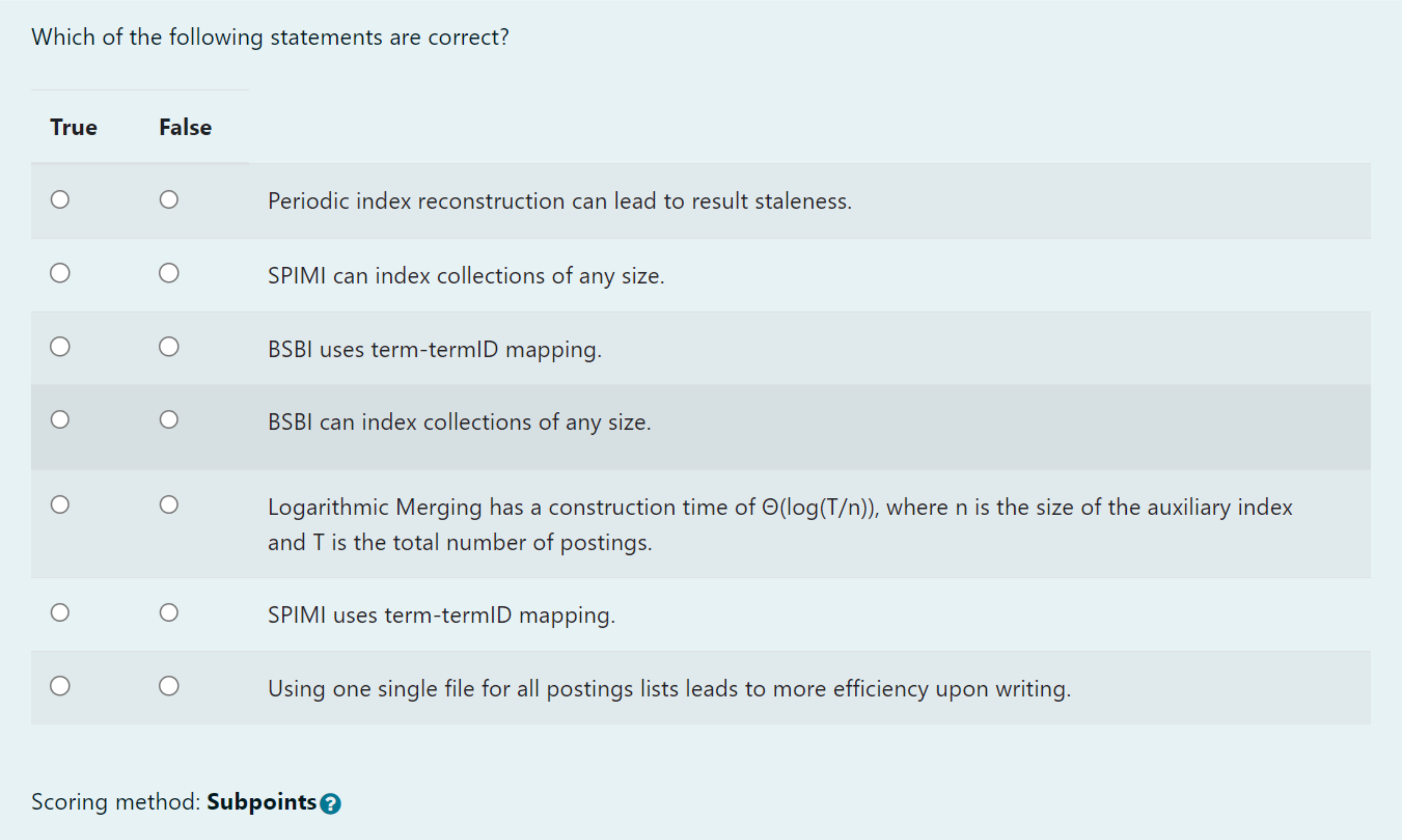


Recall the concept of index construction using logarithmic merging. Which of the following disk states (i.e. lists of indices that exists on disk at a given time) are consistent?

True	False	
○	○	I0, I1, I2, I3, I4
○	○	I0, I4
○	○	I1, I2, I3, I4
○	○	I0

Scoring method: **Subpoints**

Exercise 4: Index construction
Moodle questions: 3



Which of the following statements are correct?

True	False	
○	○	Periodic index reconstruction can lead to result staleness.
○	○	SPIMI can index collections of any size.
○	○	BSBI uses term-termID mapping.
○	○	BSBI can index collections of any size.
○	○	Logarithmic Merging has a construction time of Θ(log(T/n)), where n is the size of the auxiliary index and T is the total number of postings.
○	○	SPIMI uses term-termID mapping.
○	○	Using one single file for all postings lists leads to more efficiency upon writing.

Scoring method: **Subpoints** ?

Lecture this week: Index compression

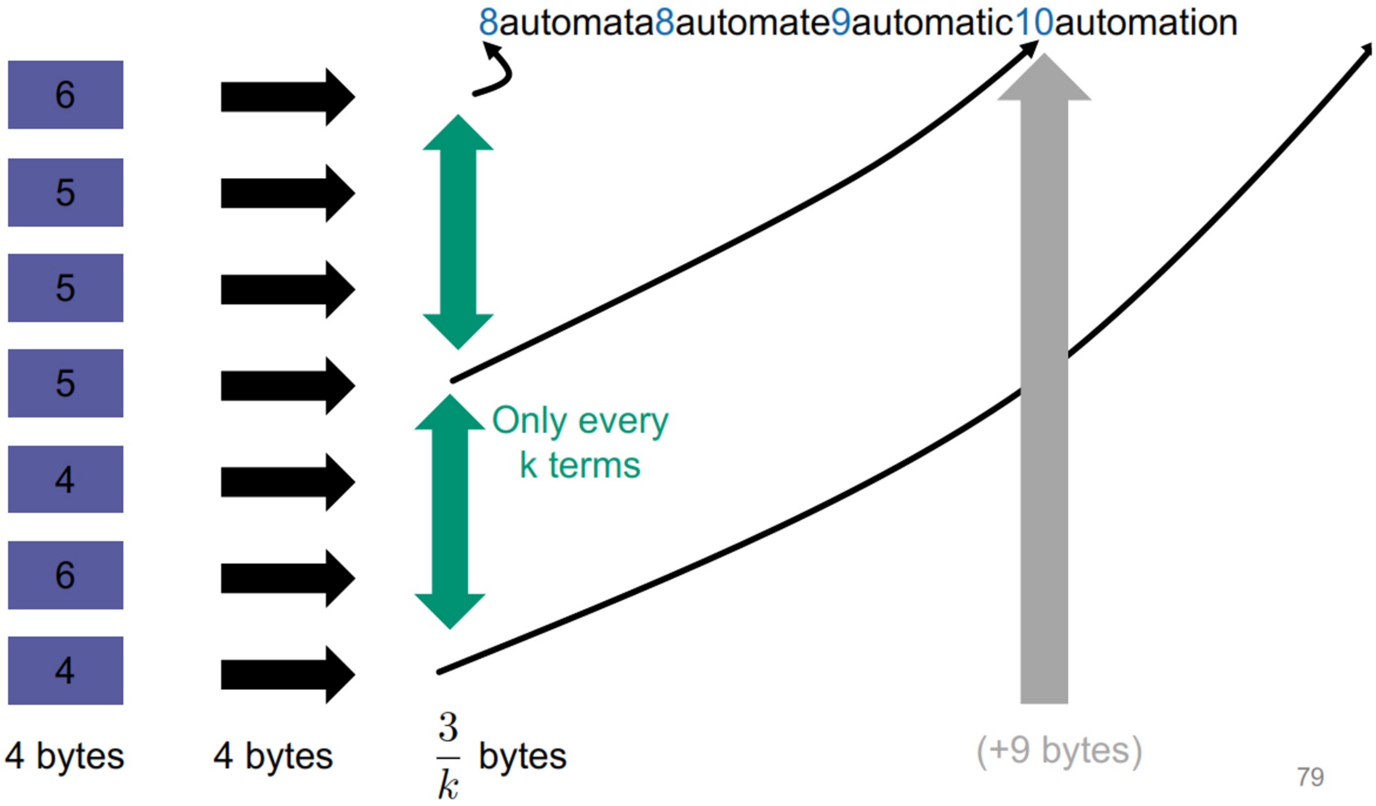
- Heaps' Law: $\#terms = k\sqrt{\#tokens}$ $30 \leq k \leq 100$
- Zipf's Law: $Frequency = \frac{c}{Rank}$

Lecture this week: Index compression

computer.....	6	→
CPU..........	5	→
data.........	5	→
ETH..........	5	→
information..	4	→
retrieval....	6	→
Zürich.......	4	→
20 bytes	4 bytes	4 bytes

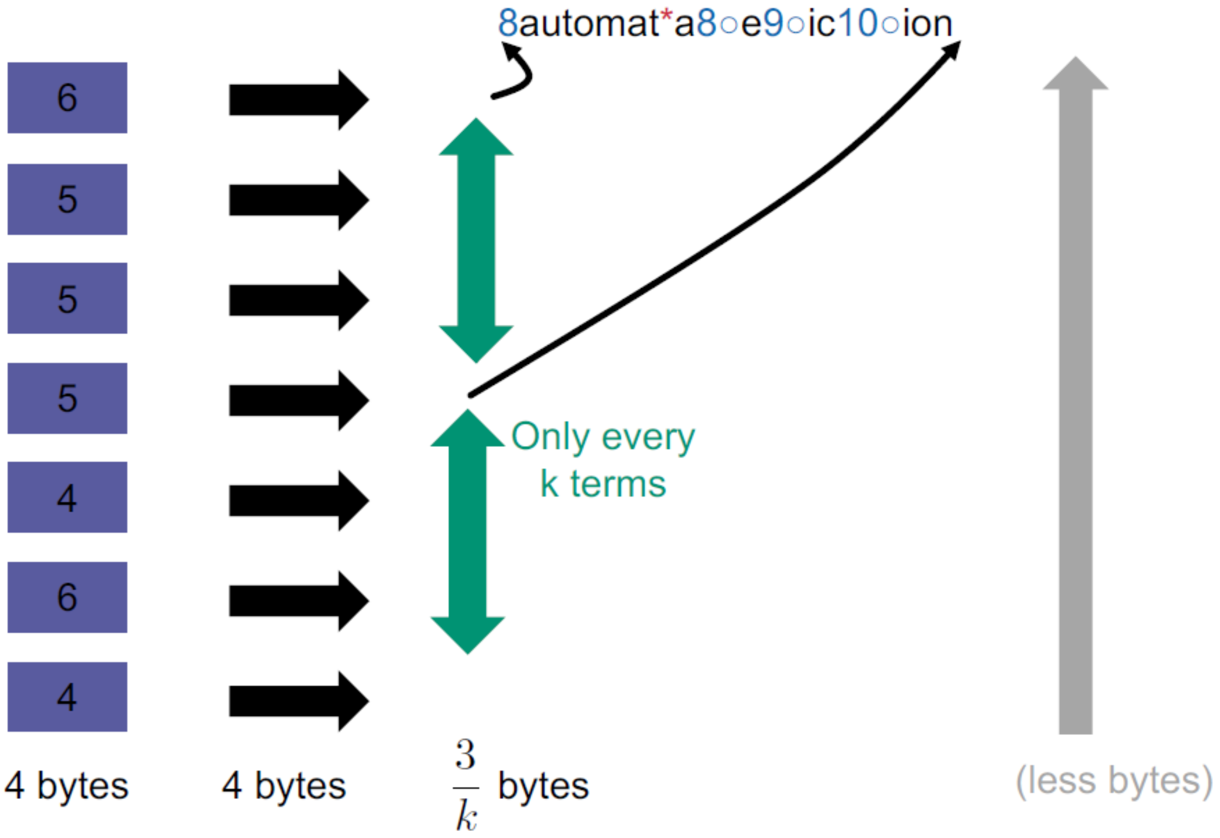
Lecture this week: Index compression Dictionary compression

- Blocked storage

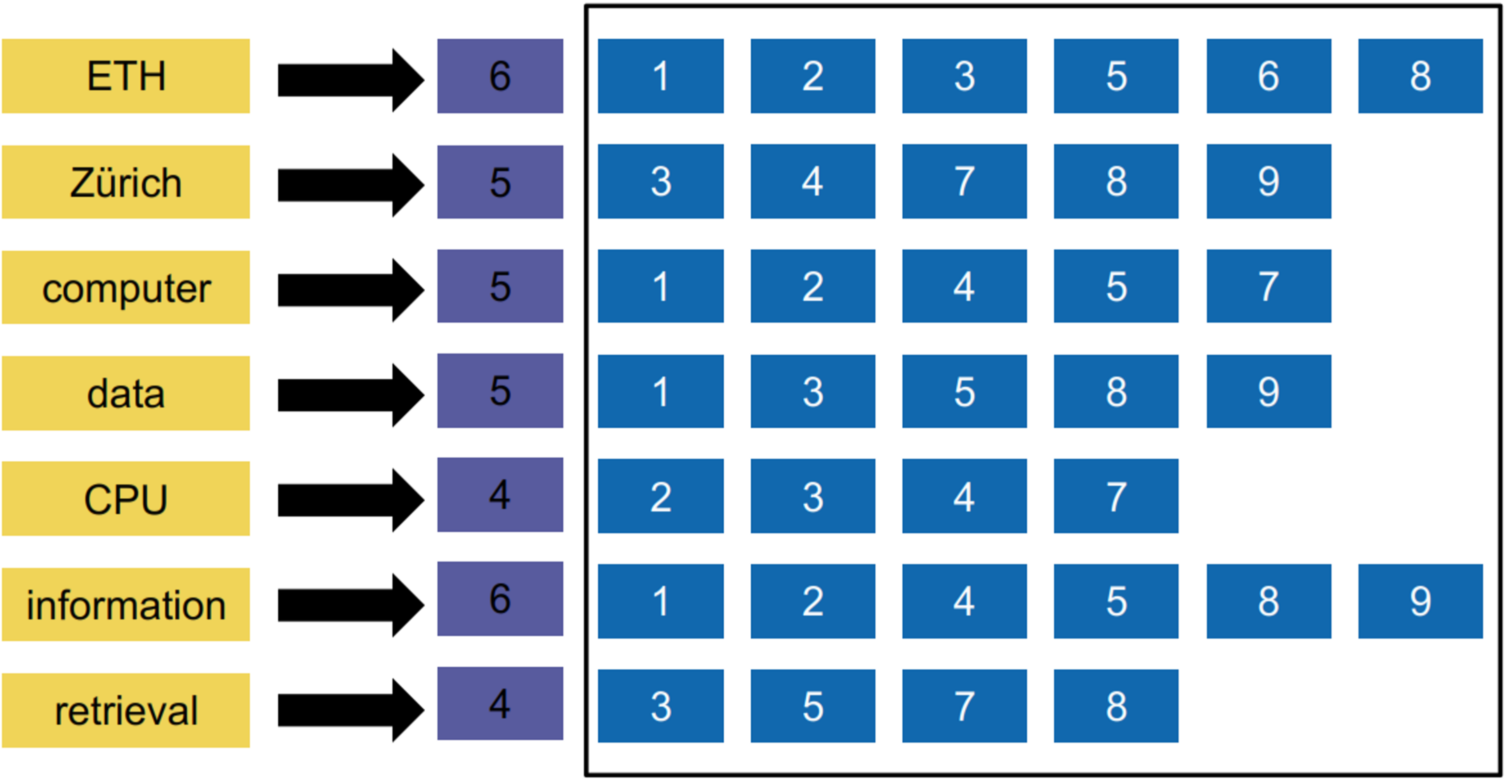


Lecture this week: Index compression
Dictionary compression

- Front coding



Lecture this week: Index compression
Posting List Compression

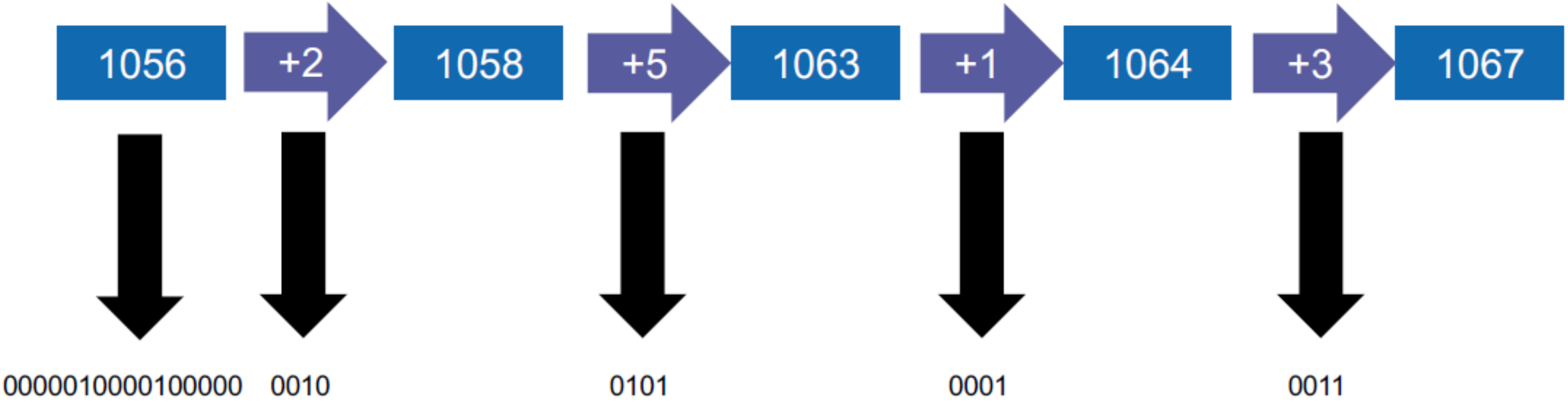


ETH	6	1	2	3	5	6	8
Zürich	5	3	4	7	8	9	
computer	5	1	2	4	5	7	
data	5	1	3	5	8	9	
CPU	4	2	3	4	7		
information	6	1	2	4	5	8	9
retrieval	4	3	5	7	8		

Now, we want to compress this.

Lecture this week: Index compression
Postings file compression

- Encode gaps. But how?



Lecture this week: Index compression
Postings file compression

- Prefix codes:
  - Phone numbers
  - UTF-8
- Variable byte encoding
- Gamma encoding

Lecture this week: Index compression
Postings file compression

- Variable byte encoding
  - Principle for 8 bit packets:

00000000

continuation bit   encoding on n-1 bits (here 7)

  - Numbers 0-64 with 4 bit packets

  - Trade-off big - small

fits on 3 bits
fits on 6 bits

decimal	binary	variable byte encoding
0	0	1000
1	1	1001
2	10	1010
3	11	1011
4	100	1100
5	101	1101
6	110	1110
7	111	1111
8	1000	0001 1000
9	1001	0001 1001
10	1010	0001 1010
11	1011	0001 1011
12	1100	0001 1100
13	1101	0001 1101
14	1110	0001 1110
15	1111	0001 1111
16	10000	0010 1000
17	10001	0010 1001
18	10010	0010 1010
19	10011	0010 1011
20	10100	0010 1100
21	10101	0010 1101
22	10110	0010 1110
23	10111	0010 1111
...	...	...
64	1000000	0001 0000 1000

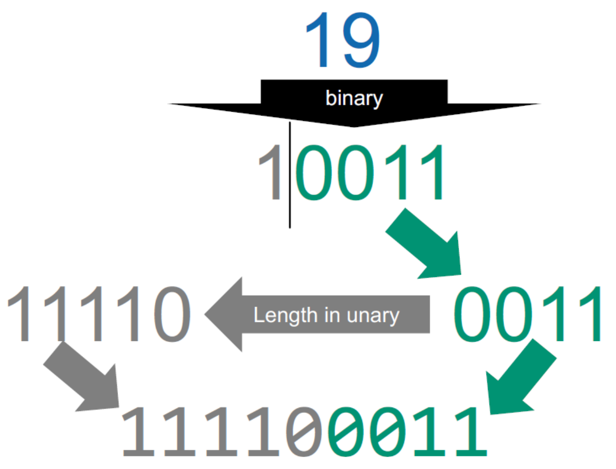
50%

less space

Lecture this week: Index compression
Postings file compression

- Gamma encoding

decimal	binary	binary without leading 1	length	length (unary)	gamma code
0	0	-	0		N/A
1	1	-	0		0
2	10	0	1	10	100
3	11	1	1	10	101
4	100	00	2	110	11000
5	101	01	2	110	11001
6	110	10	2	110	11010
7	111	11	2	110	11011
8	1000	000	3	1110	1110000
9	1001	001	3	1110	1110001
10	1010	010	3	1110	1110010
11	1011	011	3	1110	1110011
12	1100	100	3	1110	1110100
13	1101	101	3	1110	1110101
14	1110	110	3	1110	1110110
15	1111	111	3	1110	1110111
16	10000	0000	4	11110	111100000
17	10001	0001	4	11110	111100001
18	10010	0010	4	11110	111100010
19	10011	0011	4	11110	111100011
20	10100	0100	4	11110	111100100
21	10101	0101	4	11110	111100101
22	10110	0110	4	11110	111100110
23	10111	0111	4	11110	111100111
...	...	...	...	...	...
64	1000000	000000	6	1111110	1111110000000



Exercise 5: Index compression

BONUS TIME

Exercise 5: Index compression

- Moodle-based
- Start: Apr 19, 15:00
- Deadline: Apr 26, 14:59
- 6/9 required to pass
- Some theoretical questions
- You can use your code to get results for some questions (even theoretical ones)
- Some require you to do the coding exercises
- Important: You have only one try
  - Do not submit unless you are finished!

Exercise 5: Index compression

- Get yourself familiar with bit manipulation in Python
- Implement `get_len_unary()`, `gamma_encode()`, `extract_len_from_gcode()`, `gamma_decode()` to encode/decode a single number with gamma code
- Implement `gamma_encode_stream()` and `gamma_decode_stream()` to encode/decode a stream of numbers

Exercise 5: Index compression

- Special case: gamma code of 1
- How should the encoder/decoder handle the case?

Exercise 5: Index compression

- Take it a step further: construct postings lists that are compressed with gamma code