Complexity of Merging Ordered Documents

Antoine Amarilli[1] M. Lamine Ba[1]
Daniel Deutch[2] Pierre Senellart[1]

[1]Télécom ParisTech, Paris, France

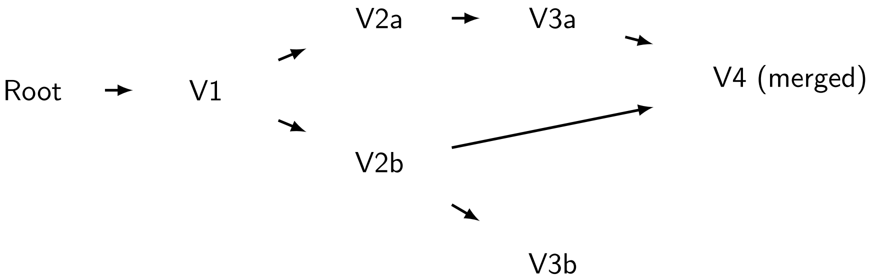
[2]Tel Aviv University, Tel Aviv, Israel

November 20, 2013

Version control

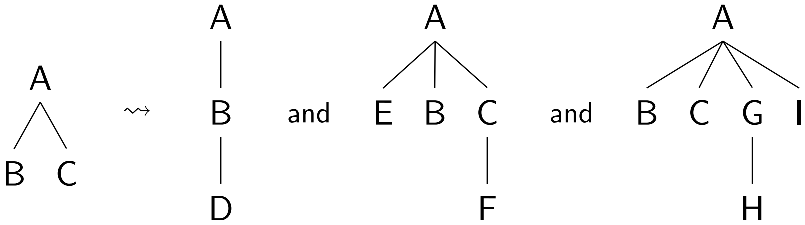
- Version control software (VCS) for line-based textual data.
- Also, tree-shaped documents:
  - ⇒ Texts (sections, paragraphs)
  - ⇒ XML
  - ⇒ Code
  - ⇒ ...
- Main problem: solving conflicts.

DAG of versions



Conflict resolution

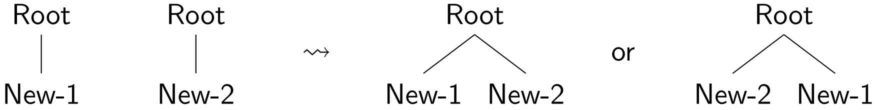
We must merge conflicting versions:



⇒ How can such conflicts be resolved?

Importance of order

- Use of probabilistic XML for versioning already investigated:
  - ⇒ M. Lamine Ba, T. Abdessalem, P. Senellart.
    Uncertain Version Control in Open Collaborative Editing of Tree-Structured Documents.
    DocEng'13, Florence, Italy.
  - ⇒ M. Lamine Ba, T. Abdessalem, P. Senellart.
    Merging Uncertain Multi-Version XML Documents.
    DChanges'13, Florence, Italy.
- Does not take order into account on child nodes.
- Important source of uncertainty!



Representing the merge

- The actual merge is chosen by the user.
- Computing all possible merges is unreasonable.
- Ideally, derive a strong representation system to represent the possible merges, merges of merges, etc.
- Here: study the complexity of a decision problem:

  Input.
  - Set of documents D to merge.
  - Possible world W of the merge.

  Output. Is W really a possible merge of D?
- We restrict the study to an ordered list for now. (List of children for a tree of height 1.)

Table of contents

Edition

- Base version B with items and unique IDs.
- Insert operation to insert a new node with a fresh ID.

Edition

- Base version B with items and unique IDs.
- Insert operation to insert a new node with a fresh ID.

#	Section
1	Introduction
2	Problem
3	Conclusion

Edition

- Base version B with items and unique IDs.
- Insert operation to insert a new node with a fresh ID.

#	Section
1	Introduction
2	Problem
3	Conclusion

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

Edition

- Base version B with items and unique IDs.
- Insert operation to insert a new node with a fresh ID.

#	Section
1	Introduction
2	Problem
3	Conclusion

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

Merge

- Union of the nodes: $D_1 \cup D_2$.
- Consistent order with the individual documents.

Merge

- Union of the nodes: $D_1 \cup D_2$.
- Consistent order with the individual documents.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

Merge

- Union of the nodes: $D_1 \cup D_2$.
- Consistent order with the individual documents.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

Merge

- Union of the nodes: $D_1 \cup D_2$.
- Consistent order with the individual documents.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
4	Extension
7	Extension
5	Related work
3	Conclusion

Possible worlds

- Exponentially many merges $\binom{|D_1|+|D_2|}{|D_1|}$
- PTIME algorithm to check if a world is a possible merge.
- Given possible world W with IDs, do the following:
  - Verify the domain.
  - For every pair $x < y$,

    For every document D containing both x and y,

    Check if $x < y$ in D.

Possible worlds

- Exponentially many merges $\binom{|D_1|+|D_2|}{|D_1|}$
- PTIME algorithm to check if a world is a possible merge.
- Given possible world W with IDs, do the following:
  - Verify the domain.
  - For every pair $x < y$,

    For every document D containing both x and y,

    Check if $x < y$ in D.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

Possible worlds

- Exponentially many merges $\binom{|D_1|+|D_2|}{|D_1|}$
- PTIME algorithm to check if a world is a possible merge.
- Given possible world W with IDs, do the following:
  - Verify the domain.
  - For every pair $x < y$,

    For every document D containing both x and y,

    Check if $x < y$ in D.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

Possible worlds

- Exponentially many merges $\binom{|D_1|+|D_2|}{|D_1|}$
- PTIME algorithm to check if a world is a possible merge.
- Given possible world W with IDs, do the following:
  - Verify the domain.
  - For every pair $x < y$,
    - For every document D containing both x and y,
      - Check if $x < y$ in D.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
5	Related work
4	Extension
3	Conclusion

Possible worlds

- Exponentially many merges $\binom{|D_1|+|D_2|}{|D_1|}$
- PTIME algorithm to check if a world is a possible merge.
- Given possible world W with IDs, do the following:
  - Verify the domain.
  - For every pair $x < y$,

    For every document D containing both x and y,

    Check if $x < y$ in D.

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
5	**Related work**
4	**Extension**
3	Conclusion

Table of contents

Possible worlds, without IDs

- We may not have the IDs in W (e.g., user input).
- Ambiguity about how to match elements!

Possible worlds, without IDs

- We may not have the IDs in W (e.g., user input).
- Ambiguity about how to match elements!

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

Possible worlds, without IDs

- We may not have the IDs in W (e.g., user input).
- Ambiguity about how to match elements!

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

Possible worlds, without IDs

- We may not have the IDs in W (e.g., user input).
- Ambiguity about how to match elements!

#	Section
1	Introduction
2	Problem
4	Extension
5	Related work
3	Conclusion

#	Section
1	Introduction
6	Preliminaries
2	Problem
7	Extension
3	Conclusion

Section
Introduction
Preliminaries
Problem
Extension
Related work
Extension
Conclusion

Dynamic algorithm

- Ambiguity makes it look like the problem is hard
- In fact a dynamic algorithm solves it in $O(n^2)$.
- Show it on an example.

Dynamic algorithm

- Ambiguity makes it look like the problem is hard
- In fact a dynamic algorithm solves it in $O(n^2)$.
- Show it on an example.

#	Label
1	A
3	A
4	B
2	B

Dynamic algorithm

- Ambiguity makes it look like the problem is hard
- In fact a dynamic algorithm solves it in $O(n^2)$.
- Show it on an example.

#	Label
1	A
3	A
4	B
2	B

#	Label
1	A
5	A
2	B
6	A

Dynamic algorithm

- Ambiguity makes it look like the problem is hard
- In fact a dynamic algorithm solves it in $O(n^2)$.
- Show it on an example.

#	Label
1	A
3	A
4	B
2	B

#	Label
1	A
5	A
2	B
6	A

Label
A
A
A
B
B
A

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	.
A_1					
A_5					
B_2					
A_6					
.					

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	.
A_1					
A_5					
B_2					
A_6					
.					

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	.
A_1	A				
A_5		A	A	B	
B_2		A	B	B	
A_6					A
.					.

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		A	A	B	
B_2		A	B	B	
A_6					A
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - $A_1\ A_3\ B_4\ B_2$.
  - $A_1\ A_5\ B_2\ A_6$.
- Possible world W: $A\ A\ A\ B\ B\ A$.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		A	A	B	
B_2		A	B	B	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		A	A	B	
B_2		A	B	$B - \searrow$	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		A	A	B	
B_2		A	$B - \rightarrow$	$B - \searrow$	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		A	A	B	
B_2		$A - \rightarrow$	$B - \rightarrow$	$B - \searrow$	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		A	A	$B - \bullet$	
B_2		$A - \rightarrow$	$B - \rightarrow$	$B - \searrow$	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$.$
A_1	A				
A_5		A	$A - \downarrow$	$B - \bullet$	
B_2		$A - \rightarrow$	$B - \rightarrow$	$B - \searrow$	
A_6					$A - \downarrow$
$.$					$. - .$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	A				
A_5		$A - \not\searrow$	$A - \downarrow$	$B - \bullet$	
B_2		$A - \rightarrow$	$B - \rightarrow$	$B - \searrow$	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

Dynamic algorithm (example)

- Set of documents D:
  - A_1 A_3 B_4 B_2.
  - A_1 A_5 B_2 A_6.
- Possible world W: A A A B B A.

	A_1	A_3	B_4	B_2	$\cdot$
A_1	$A - \searrow$				
A_5		$A - \nrightarrow\!\!\!\downarrow$	$A - \downarrow$	$B - \bullet$	
B_2		$A - \rightarrow$	$B - \rightarrow$	$B - \searrow$	
A_6					$A - \downarrow$
$\cdot$					$\cdot - \cdot$

More documents

- If we have multiple documents to merge (say k).
- Generalized dynamic algorithm is in $O(n^k)$.
- Hence, PTIME for fixed k.
- What if k is not fixed?
- Certainly it is still in NP.
- In fact, it is NP-hard (reduction from MinSAT).
- Thanks: `http://cstheory.stackexchange.com/a/19081/`

Table of contents

Conflicts

- Move operation to move arbitrary nodes.
- Documents can disagree.

Conflicts

- Move operation to move arbitrary nodes.
- Documents can disagree.

#	Section
1	Problem A
2	Problem B

Conflicts

- Move operation to move arbitrary nodes.
- Documents can disagree.

#	Section
1	Problem A
2	Problem B

#	Section
1	Problem A
2	Problem B
3	Conclusion

Conflicts

- Move operation to move arbitrary nodes.
- Documents can disagree.

#	Section
1	Problem A
2	Problem B

#	Section
1	Problem A
2	Problem B
3	Conclusion

#	Section
4	Intro
2	Problem B
1	Problem A

Merge

- Order is arbitrary if documents disagree.
- However, try to maintain order within documents.

Merge

- Order is arbitrary if documents disagree.
- However, try to maintain order within documents.

#	Section
1	Problem A
2	Problem B
3	Conclusion

Merge

- Order is arbitrary if documents disagree.
- However, try to maintain order within documents.

#	Section
1	Problem A
2	Problem B
3	Conclusion

#	Section
4	Intro
2	Problem B
1	Problem A

Merge

- Order is arbitrary if documents disagree.
- However, try to maintain order within documents.

#	Section
1	Problem A
2	Problem B
3	Conclusion

#	Section
4	Intro
2	Problem B
1	Problem A

#	Section
4	Intro
1	Problem A
2	Problem B
3	Conclusion

Merge

- Order is arbitrary if documents disagree.
- However, try to maintain order within documents.

#	Section
1	Problem A
2	Problem B
3	Conclusion

#	Section
4	Intro
2	Problem B
1	Problem A

#	Section
4	Intro
1	Problem A
2	Problem B
3	Conclusion

$\Rightarrow$ Only one thing to decide here.

Weird cases

- Sometimes weird!

Weird cases

- Sometimes weird!

#	Section
1	Problem A
3	Transition A-B
2	Problem B

Weird cases

- Sometimes weird!

#	Section
1	Problem A
3	Transition A-B
2	Problem B

#	Section
2	Problem B
4	Transition B-A
1	Problem A

Weird cases

- Sometimes weird!

#	Section
1	Problem A
3	Transition A-B
2	Problem B

#	Section
2	Problem B
4	Transition B-A
1	Problem A

#	Section
2	Problem B
4	Transition B-A
1	Problem A
3	Transition A-B

Weird cases

- Sometimes weird!

#	Section
1	Problem A
3	Transition A-B
2	Problem B

#	Section
2	Problem B
4	Transition B-A
1	Problem A

#	Section
2	Problem B
4	Transition B-A
1	Problem A
3	Transition A-B

$\Rightarrow$ Satisfy a maximal subset of the original constraints.

Weird cases

- Sometimes weird!

#	Section
1	Problem A
3	Transition A-B
2	Problem B

#	Section
2	Problem B
4	Transition B-A
1	Problem A

#	Section
2	Problem B
4	Transition B-A
1	Problem A
3	Transition A-B

$\Rightarrow$ Satisfy a maximal subset of the original constraints.

$\Rightarrow$ It is still in NP to decide if something is a possible world.

$\Rightarrow$ It is NP-hard even for 2 documents (from Unary 3-partition).

$\Rightarrow$ Thanks: `http://cstheory.stackexchange.com/a/19415/`

Table of contents

Nested hierarchies

- Necessary for XML.
- Hardness results still holds.
- Dynamic algorithm can be extended:
  - When solving the problem for two forests:
    - Solve for each pair of children.
    - Use the dynamic algorithm to solve the child sequence.
  - When solving the problem for two trees:
    - Check equality of the root labels.
    - Solve the forests of children.

Relational algebra

- Add order to relational databases.
- Allow arbitrary relational operations.
- Track provenance of the data.
- Use provenance for order uncertainty.
- Connected with provenance for aggregates.

Thanks!

Thanks for your attention!

(Work in progress, questions and feedback welcome.)

Hardness of multiple documents

- Reduction from MinSAT:

  Input. n clauses $\pm x_i \vee \pm x_j$, k variables, integer d

  Output. can we avoid satisfying $> d$ clauses?

- IDs:
  - e_i^+ for $+x_i$, e_i^- for $-x_i$
  - c_i for clause i
- Labels:
  - Clause, for clauses,
  - Var_i for variable i
- Document set D:
  - $e_i^\pm c_j$ if $\pm x_i$ occurs in c_j
- Possible world W:
  - Var_i from 1 to k
  - Clause $n-d$ times
  - Var_i from 1 to k
  - Clause d times

Hardness of multiple documents (example)

- MinSAT instance:
  - $C_1 : x \vee y$
  - $C_2 : \neg x \vee \neg y$
  - $d = 0$
- Encoding (D, and W):

#	Label
e_x^+	Var_x
c_1	Clause

#	Label
e_y^+	Var_y
c_1	Clause

#	Label
e_x^-	Var_x
c_2	Clause

#	Label
e_y^-	Var_y
c_2	Clause

Label
Var_x
Var_y
Clause
Clause
Var_x
Var_y

Hardness of multiple documents (example)

- MinSAT instance:
  - $C_1 : x \vee y$
  - $C_2 : \neg x \vee \neg y$
  - $d = 0$
- Encoding (D, and W):

#	Label
e_x^+	Var_x
c_1	Clause

#	Label
e_y^+	Var_y
c_1	Clause

#	Label
e_x^-	Var_x
c_2	Clause

#	Label
e_y^-	Var_y
c_2	Clause

Label
Var_x
Var_y
Clause
Clause
Var_x
Var_y

Hardness of multiple documents (example)

- MinSAT instance:
  - $C_1 : x \vee y$
  - $C_2 : \neg x \vee \neg y$
  - $d = 0$
- Encoding (D, and W):

#	Label
e_x^+	Var_x
c_1	Clause

#	Label
e_y^+	Var_y
c_1	Clause

#	Label
e_x^-	Var_x
c_2	Clause

#	Label
e_y^-	Var_y
c_2	Clause

Label
Var_x
Var_y
Clause
Clause
Var_x
Var_y

Hardness of multiple documents (example)

- MinSAT instance:
  - $C_1 : x \vee y$
  - $C_2 : \neg x \vee \neg y$
  - $d = 0$
- Encoding (D, and W):

#	Label
e_x^+	Var_x
c_1	Clause

#	Label
e_y^+	Var_y
c_1	Clause

#	Label
e_x^-	Var_x
c_2	Clause

#	Label
e_y^-	Var_y
c_2	Clause

Label
Var_x
Var_y
Clause
Clause
Var_x
Var_y

Hardness of multiple documents (example)

- MinSAT instance:
  - $C_1 : x \vee y$
  - $C_2 : \neg x \vee \neg y$
  - $d = 0$
- Encoding (D, and W):

#	Label
e_x^+	Var_x
c_1	Clause

#	Label
e_y^+	Var_y
c_1	Clause

#	Label
e_x^-	Var_x
c_2	Clause

#	Label
e_y^-	Var_y
c_2	Clause

Label
Var_x
Var_y
Clause
Clause
Var_x
Var_y

Hardness of multiple documents (example)

- MinSAT instance:
  - $C_1 : x \vee y$
  - $C_2 : \neg x \vee \neg y$
  - $d = 0$
- Encoding (D, and W):

#	Label
e_x^+	Var_x
c_1	Clause

#	Label
e_y^+	Var_y
c_1	Clause

#	Label
e_x^-	Var_x
c_2	Clause

#	Label
e_y^-	Var_y
c_2	Clause

Label
Var_x
Var_y
Clause
Clause
Var_x
Var_y

- Thanks: `http://cstheory.stackexchange.com/a/19081/`

Hardness

- Merging two versions is already NP-complete.
- Reduction from Unary 3-partition:

  Input. $3m$ integers n_i in unary, and integer B.

  Output. can we partition in triples with sum always B?
- Create two blocks per integer n_i:
  - Both start with an element labeled Open.
  - Both contain n_i elements labeled Item:
    - ⇒ First block in one order.
    - ⇒ Second block in reverse order.
  - Both end with an element labeled Close.
- Document set D:
  - D_1: concatenate all blocks in one order.
  - D_2: concatenate all blocks in the reverse order.
- Possible world W:
  - Open^3, Item^B, Close^3.
  - Repeat m times.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

$\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.

Hardness (intuition and example)

- Induced order constraints:
  - All Item elements are incomparable.
  - Open and Close precede and follow all of their Item's.
  - Open, Item, Close blocks are incomparable among them.
- 3-partition instance: $\{1, 1, 1, 2, 3, 4\}$, $B = 6$.
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item, Close
  - Open, Item^2, Close
  - Open, Item^3, Close
  - Open, Item^4, Close

  $\Rightarrow$ Open^3, Item^6, Close^3, Open^3, Item^6, Close^3.
- Thanks: `http://cstheory.stackexchange.com/a/19415/`