Wednesday, December 7, 2022 1:04 PM

Scenario #2: With all the online work going on these days, you've decided to create a webbased document editor (like a simplified Google Docs. Users of your tool will edit documents through a web client, with the actual document contents stored on a server. To keep the project manageable, you'll focus on four operations: adding text, deleting text, undoing edits, and searching for words in the document. The editor should allow documents to have basic styling, such as boldface words/phrases and section headings. In particular, searching for words needs to be fast (because you imagine people writing large documents in your tool once it becomes popular).

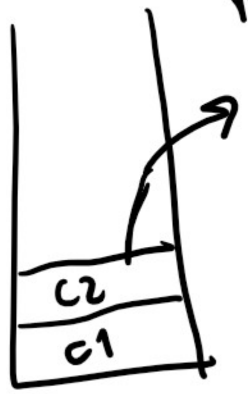
Question 1: What data do you need to manage for this problem?

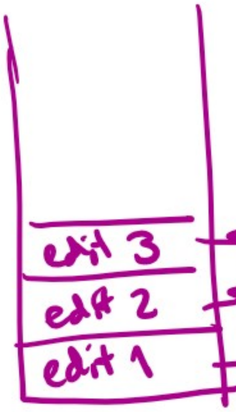
Question 2: What data structure(s) do you propose? Be sure to include details such as the type of elements in lists, the types of keys/values in hashmaps, whether objects are mutable or immutable, etc. Justify your data structure choices

↳ representation of doc.

undoing

new copy of doc every time user makes action?





→ delete char. at loc. 3 ("s")

→ bold the "a" at loc. 1

→ put "cats" in loc. 0 of document

Edit class:

- the action taken
- the info necessary to undo the action

Format:

X dict words → format

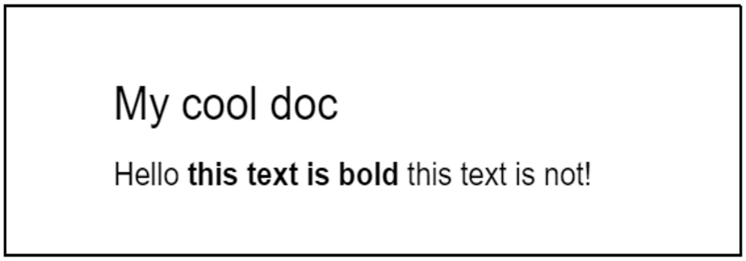
X dict format → words

X dict locations → format

dict format → ranges of locs.

HTML

Wednesday, December 7, 2022 1:11 PM

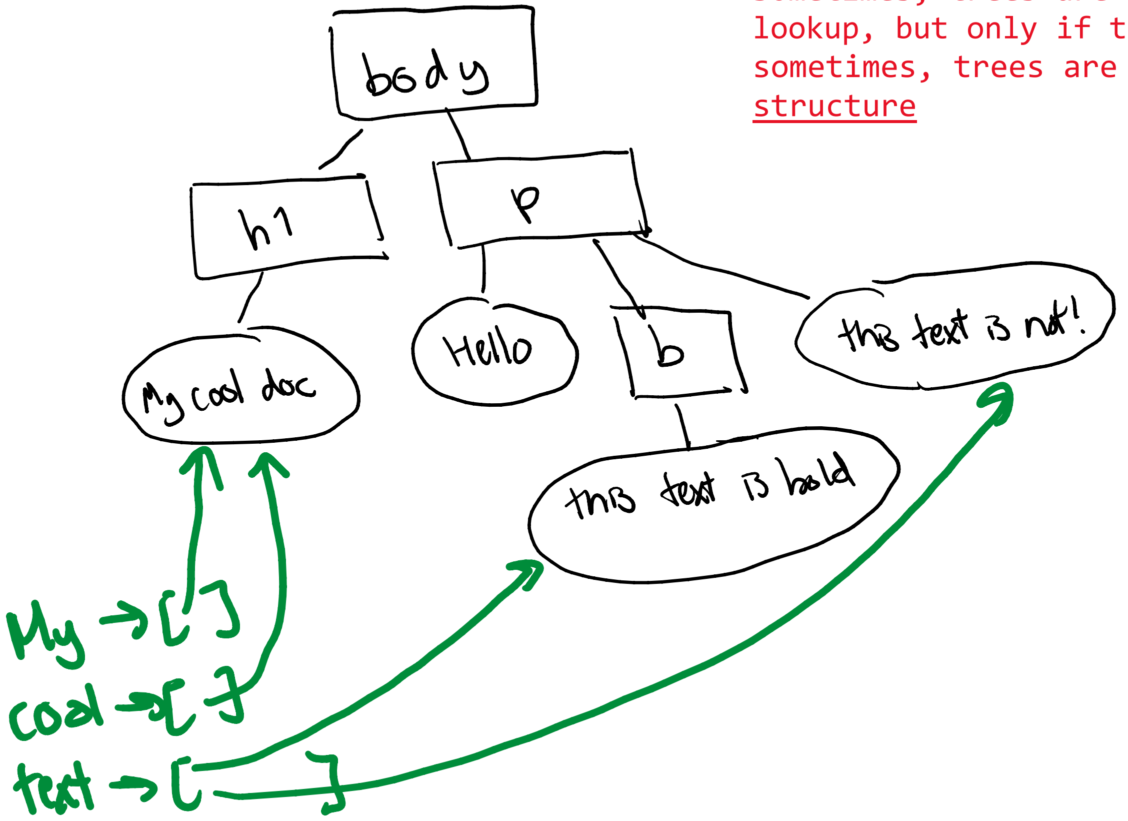


My cool doc

Hello **this text is bold** this text is not!

```
<html>
   <body>
      <h1>My cool doc</h1>
      <p>Hello <b>this text is bold</b> this text is not!</p>
   </body>
</html>
```

Sometimes, trees are useful (log N) for lookup, but only if there's an order to them
sometimes, trees are useful to describe structure


body
h1
p
My cool doc
Hello
b
this text is not!
this text is bold
My → [ ]
cool → [ ]
text → [ ]