Two Big Challenges on Project 2

Challenge 1: Figuring out how to do basic tasks.

- Drawing a cursor.
- Drawing multiple pieces of text to the screen.

Challenge 2: Figuring out how to design your overall system.

- Not a bad idea to list all your classes and instance variables.
  - Good programmers worry about their data structures.
  - Will determine how efficiently your code runs and how hard your project is to implement.
- Might also list your methods.
- Check to make sure that you can do everything with your design.
- May find that you need to revise later.

HelloWorlding

Example: Want to figure out how to draw a cursor?

- Consider doing it in a file other than editor/Editor.java.
- Much simpler to think about in a limited context.
  - Create a file called DrawCursorInTheMiddleOfTheScreen.java.

Design Tips

Consider filling out a project 2 design worksheet.

- Example of a not-so-great design: String-Based Design

Key runtime constraints:

- Must take constant time to insert, even into middle of the text.
- Must take constant time to do clicks or arrow keys (which is also linear time in the length of the line that the cursor ends up on).
- Must take no more than linear time to re-render the document.

Constant: No matter how long the file is, runtime should be the same.

Linear: Runtime should be proportional to the file length.

Perilous Data Structures

- Using a String to store data.
- Using a java.util.LinkedList to store data typed by the user
    - Why?
    - Cannot support constant time insertions into the middle of the data structure.
- Using any type of java.util.Collection~~<Text> to store data typed by the user (where Text is a JavaFX Text object).~~
    - Why?
    - This was kind of a lame question, sorry, but basically none of the built in collections will do what you want.

Video 2: More on Data Structures and Tricky Cases

Video Link: https://youtu.be/K5nsYVF96HY

Analysis of Linked List Data Structure

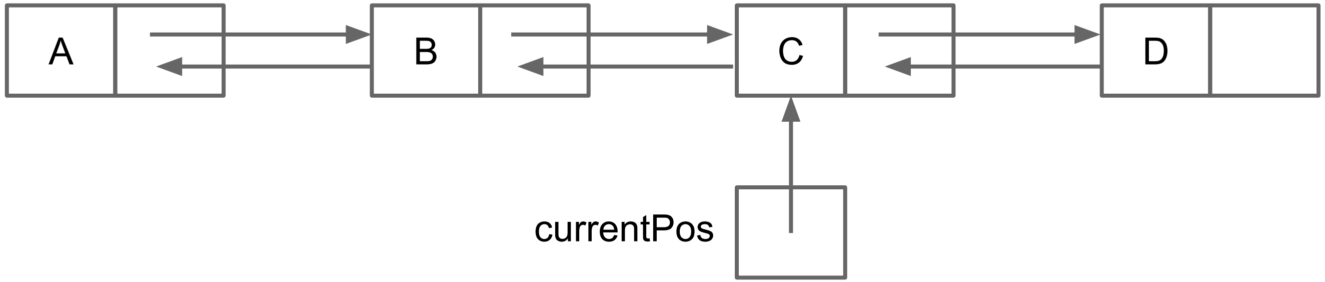
Last video, we discussed why a String isn’t great.

One better solution (alluded to in the spec) is a Linked List of Text.

- Note: Add to the middle of a java.util.LinkedList is linear time. In other words add(1000, new TextNode(‘c’)) is going to be slow, since the java.util.LinkedList needs to scan through the data structure.
- Can get around this by building your own FastLinkedList that keeps track of the current position (see next slide)

Analysis of Linked List Data Structure

FastLinkedList:



Analysis of this approach:

- https://docs.google.com/document/d/1eIuf-7Lznfjq6Vccu-JnmIr8uWXdK1r3ufq2d5o6mIl/edit

Analysis of Linked List Data Structure

Fundamental issue with using a LinkedList alone:

- Impossible to speedily handle clicks.

Two common approaches I've seen:

- Augmentation: Storing information needed for constant time click in a separate data structure.
- Fission: Store each line in a separate list.

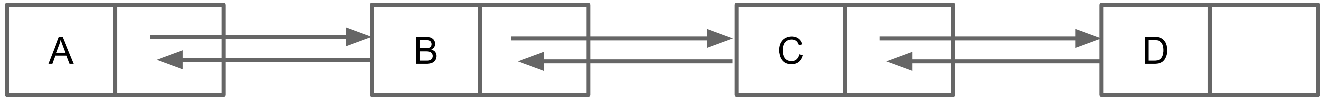
Augmentation approach is likely to be less work!

- Prevents potentially complex propagation issues.

Augmentation vs. Fission

Augmentation:

- DS1: Typed chars:

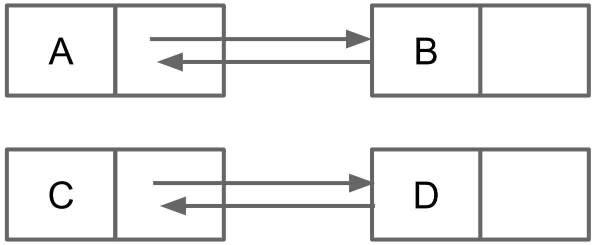


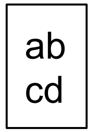
- DS2: Plus some other data structure that keeps track of information useful for supporting fast clicking.

Create a class (object) to record list, cursor, responding coordinate

Fission:

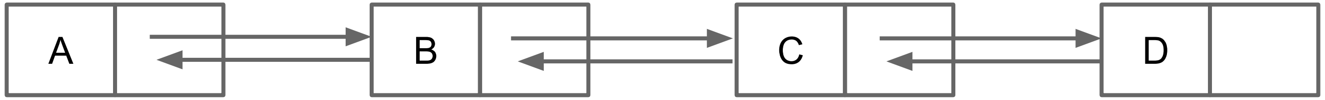
- Typed chars:





More on the Augmentation Approach

Augmentation:

- DS1: Typed chars: 

- DS2: Plus some other data structure that keeps track of information useful for supporting fast clicking.

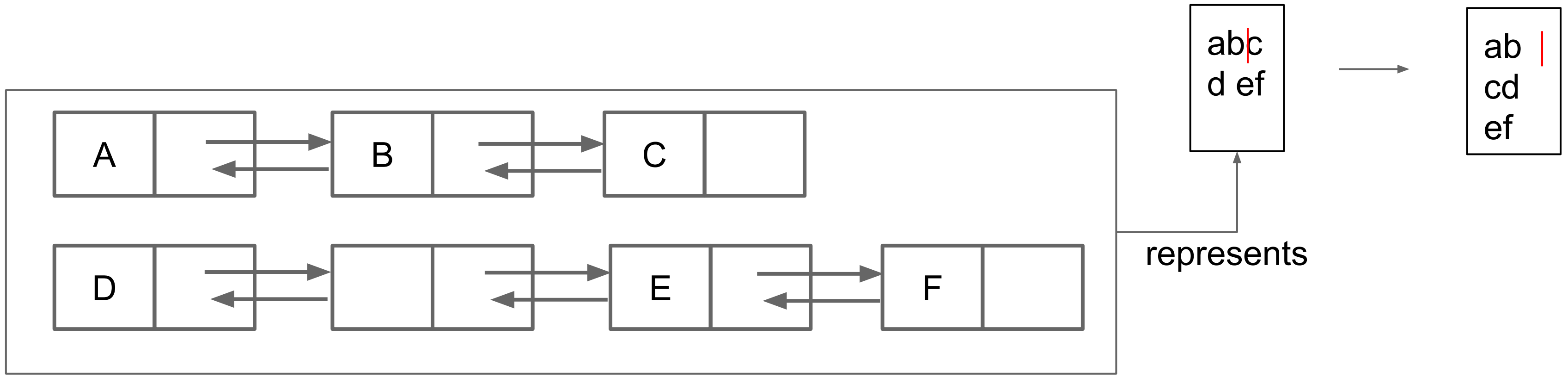
What might this data structure be?

- Imagine how you might support the operation “go to line #x”, where x is some integer. What data structure would you use?
- How would this same data structure be useful for clicks?

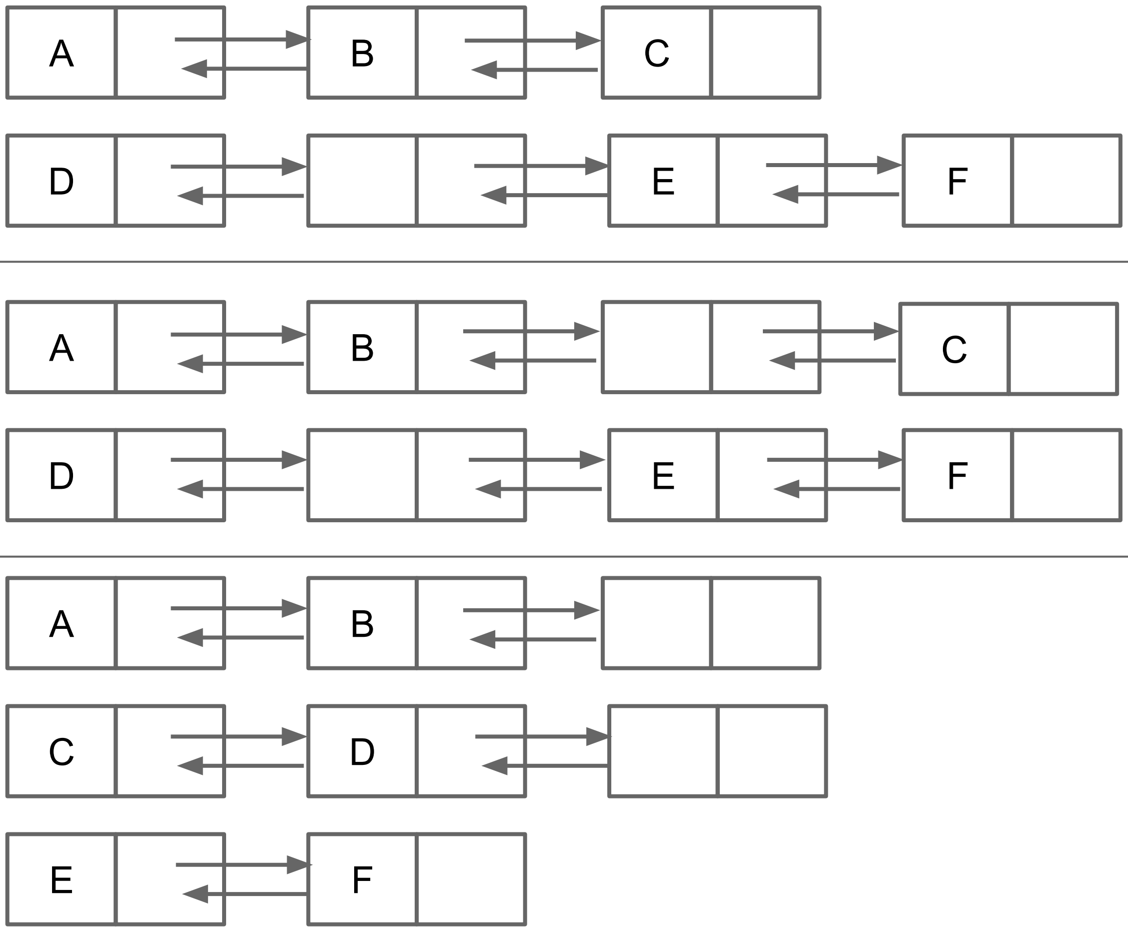
Fission Approach Example

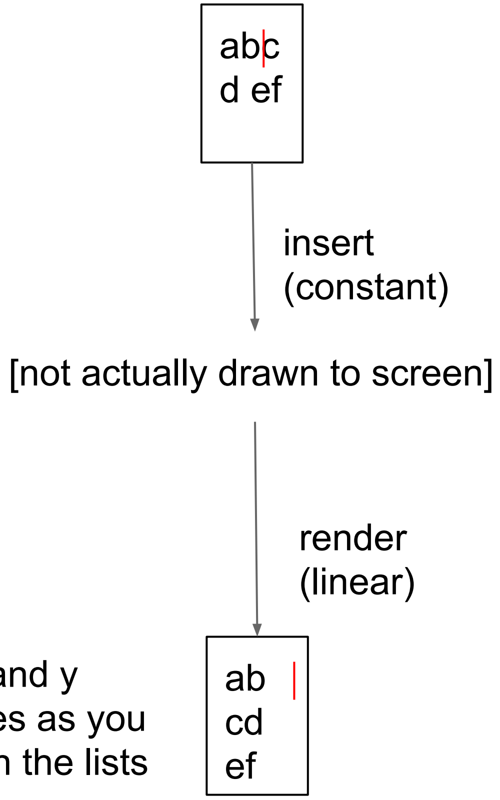
Example, suppose we have the editor window shown, with cursor after the b:

- If we press space, the editor window will change as shown.
- In a fission based approach, changes can cascade.
  - Need to move things between lists.



Fission Approach Example (Two-Stage Update)



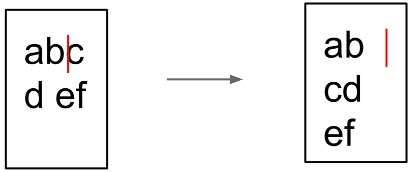


update x and y coordinates as you go through the lists

Fission

You have our blessing to do a two-stage fission update, but:

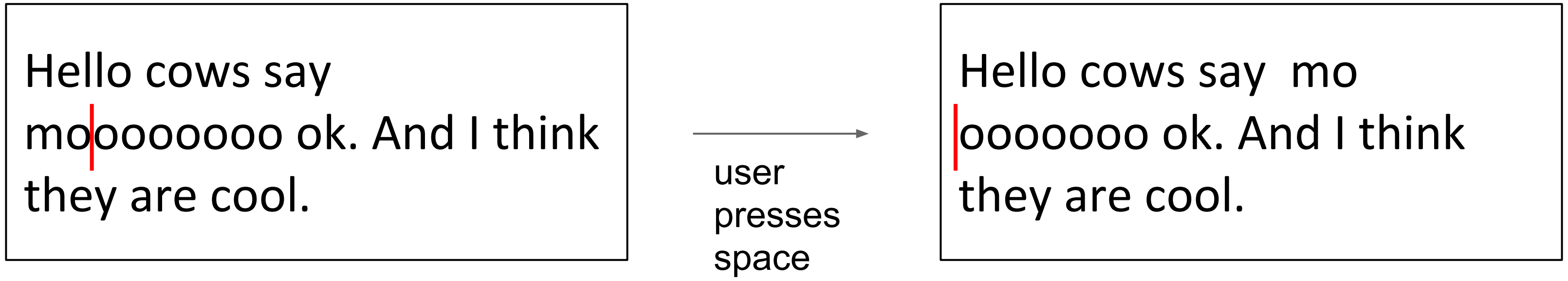
- We suspect it will be more work than an augmentation approach due to cascades, especially example on next slide.
- Make sure your code does not take quadratic time to open a file!



One Case Worth Noting

One issue that popped up at office hours was students not considering the following case where the user presses space in the middle of a word.

- Possible for something to get moved UP a line if space is pressed.



Hello cows say
mooooooooo ok. And I think
they are cool.

user presses space

Hello cows say mo
oooooooo ok. And I think
they are cool.

FA-FAQ, Warnings, and Recommendations

- Recommended that you store Text objects not Characters or Strings.
- It is possible to modify a Text object's X and Y coordinate, e.g. using setX.
  - Don't need to destroy all your text objects and recreate them.
- **If you're stuck, consider implementing a slow version of a feature instead.**
- Don't need to add/remove children every time you draw to the screen.
- It is OK to scan from one line to the next to handle up/down arrows (since each line is constant in length).
- Expected timing tests:
  - We will click at beginning / end of file and ensure runtime is constant.
  - We will open a large file and ensure opening is linear time (not quadratic).
  - Will not be directly testing constant insertion time, since we have no way of isolating this part of your code, but opening a large file must still be linear time!
- Inserting into the middle of an ArrayList is linear time!

Project Late Policy

Due 3/7 at 11:59 PM

- 3/8: Lose up to 2%
- 3/9 and on: Lose 10% each day

Basics Autograder due 3/2 2016 @ 11:59 PM

- Ross informs me our client side AG will be available late tonight.

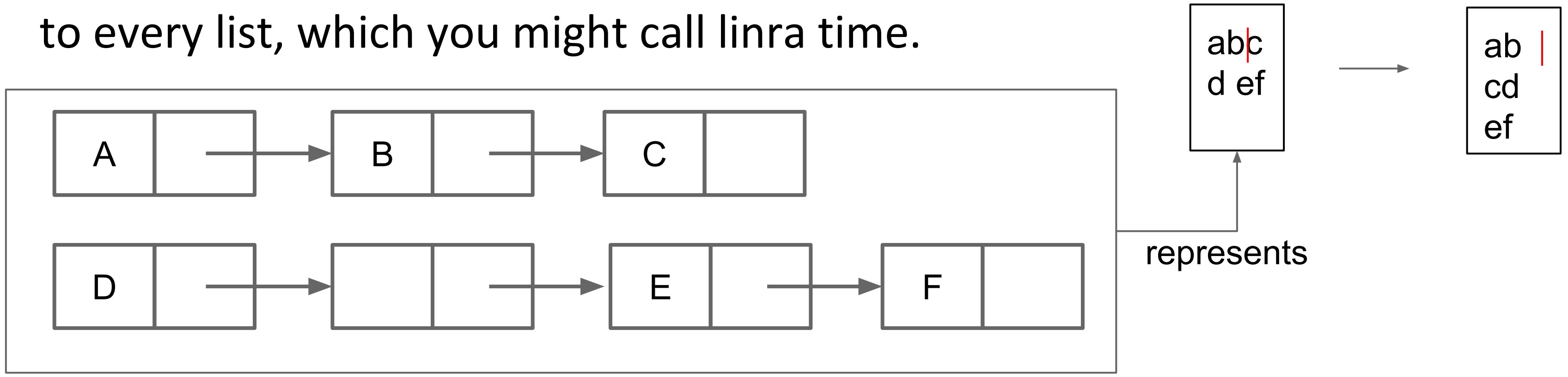
Extra Slides

Runtime Ambiguity Update

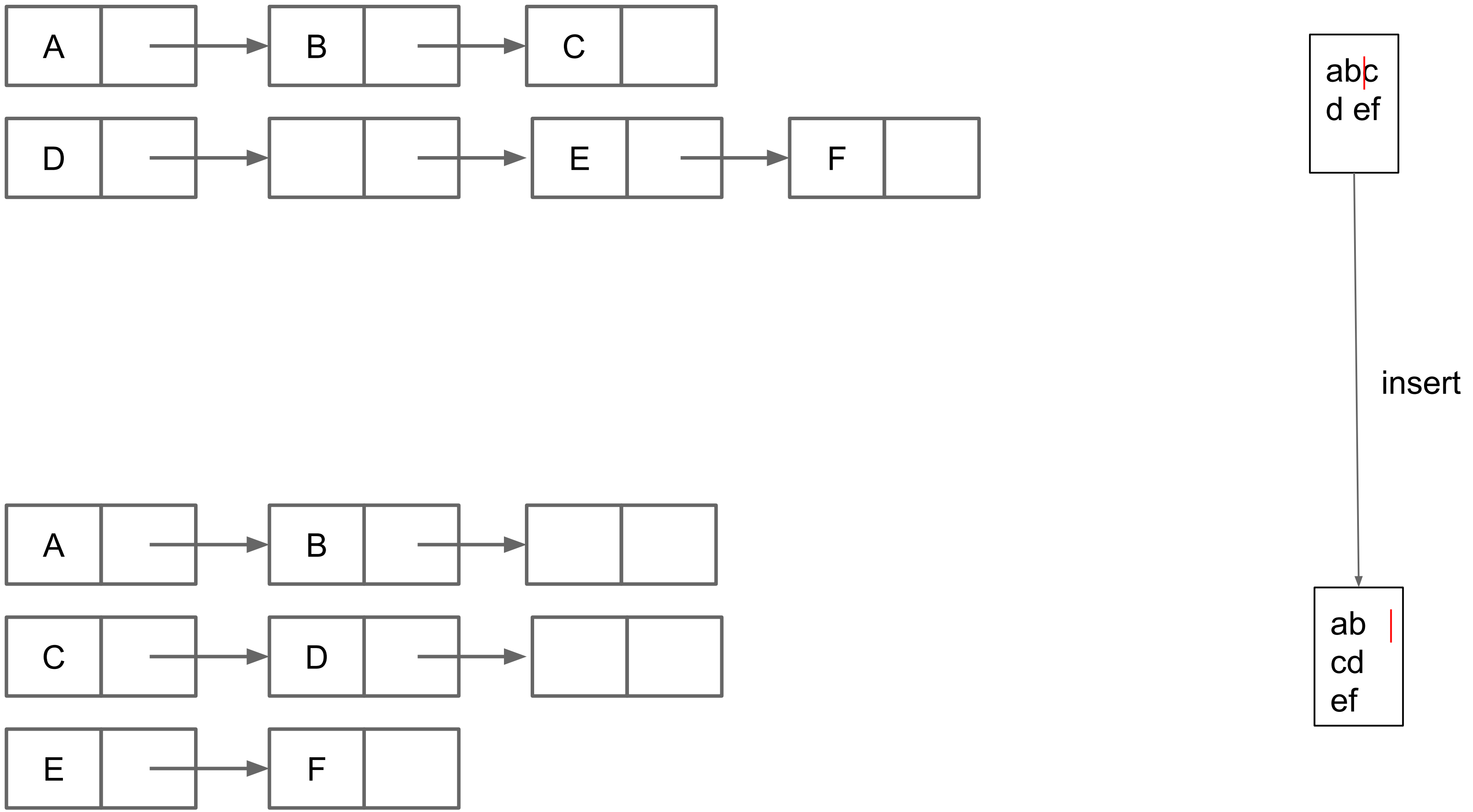
One could argue that Fission based approaches have linear time insert and are thus disallowed. However, we consider the Fission approach to be constant time. These slides are a detailed discussion of this line of issue.

Example, suppose we have the editor window shown, with cursor after the b:

- If we press space, the editor window will change as shown.
- In a fission based approach, that could potentially mean a cascading change to every list, which you might call linra time.



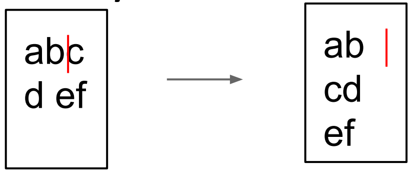
Fission Example (looks linear time)



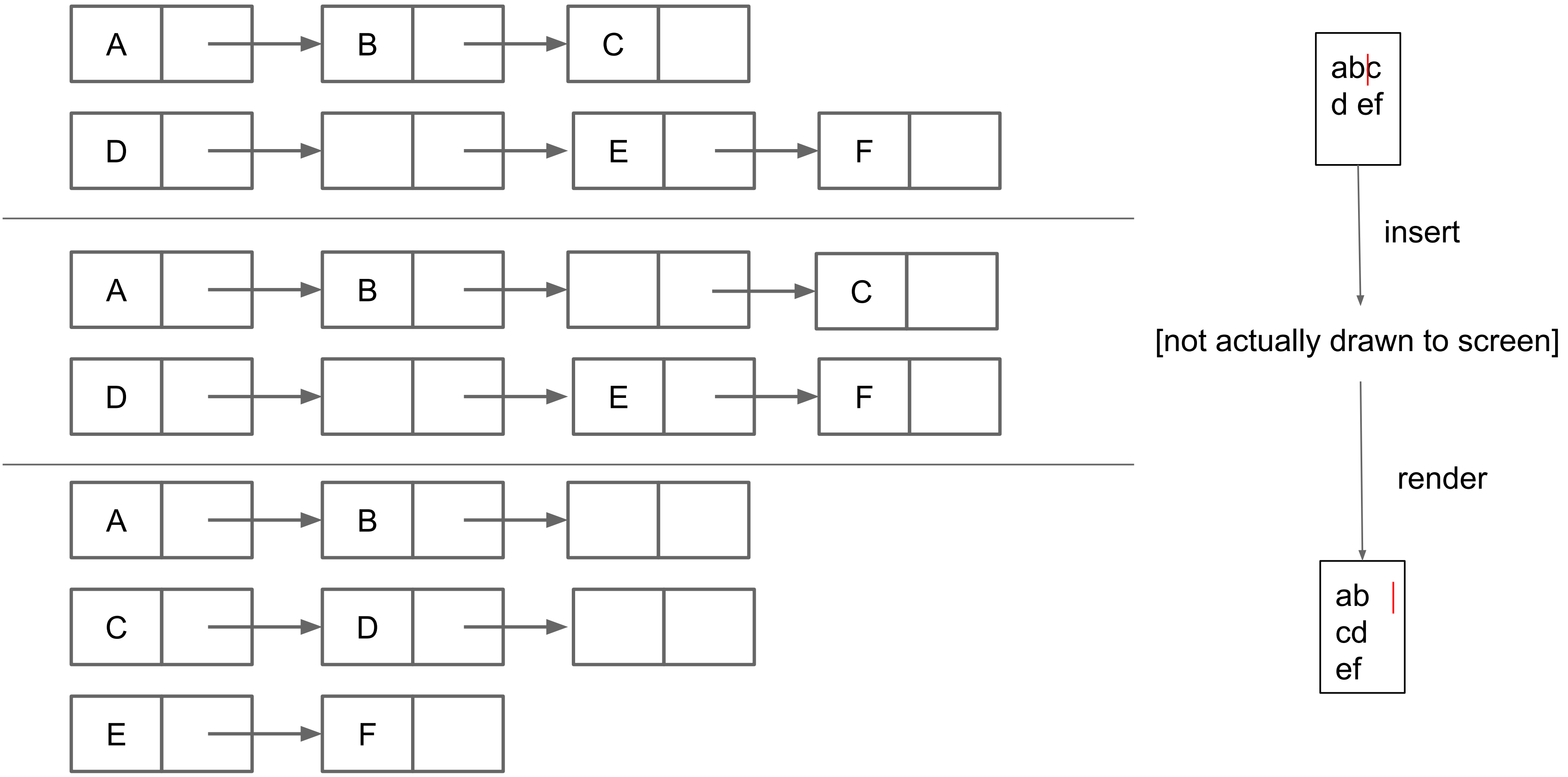
Runtime Ambiguity Update

Example, suppose we have the editor window shown, with cursor after the b:

- In a fission based approach, pressing space could potentially mean a cascading change to every list (not constant time).
- However, if we think of processing a new keypress list as a two-phase process, things are fine.
  - When inserting, add to the current list, even if this makes the list too long to fit on a line (constant time).
  - Then during rendering, perform the fission operation (which can be linear time).
  - As noted, you have our blessing to do this, but make sure your code does not take quadratic time to open a file!



Fission Example, Treating as a Constant then Linear Time Process


A
B
C
D
E
F
A
B
C
D
E
F
A
B
C
D
E
F
abc
d ef
insert
[not actually drawn to screen]
render
ab
cd
ef