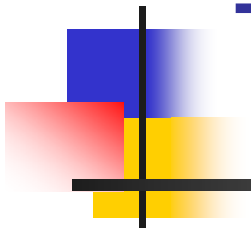
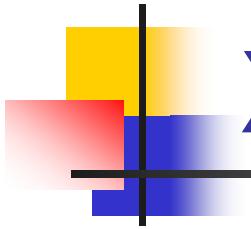


COMP4317: XML & Database Tutorial 6: Xpath Evaluation

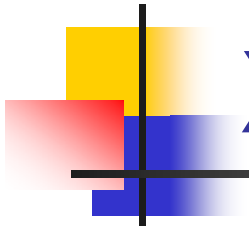
Week 7

Thang Bui @ CSE.UNSW



Xalan XPath

- implements XPath version 1.0 (Java: javax.xml.xpath)

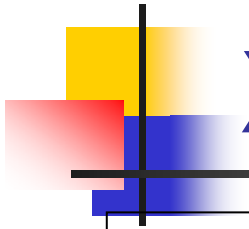


Xalan XPath: How to use

- Instantiate an XPathFactory
- Create an XPath
- (optional) Create an XPathExpression
- Evaluate the XPath expression in a specified context

Xalan XPath - Java

```
// 1. Instantiate an XPathFactory.
javax.xml.xpath.XPathFactory factory =
                    javax.xml.xpath.XPathFactory.newInstance();

// 2. Use the XPathFactory to create a new XPath object
javax.xml.xpath.XPath xpath = factory.newXPath();

// 3. Compile an XPath string into an XPathExpression
javax.xml.xpath.XPathExpression expression = xpath.compile("/doc/name");

// 4. Evaluate the XPath expression on an input document
String result = expression.evaluate(new org.xml.sax.InputSource("foo.xml"));
```



Xalan XPath - Java

```
import javax.xml.xpath.*;
import org.xml.sax.*;
import org.w3c.dom.*;

public class tutor6 {
  public static void main(String[] args)
  {
    InputSource xml = new InputSource(args[0]);
    String expr = args[1];

    // Create a new XPath
    XPathFactory factory = XPathFactory.newInstance();
    XPath xpath = factory.newXPath();
```

Xalan XPath - Java

```
    NodeList result = null;
    try {
      // compile the XPath expression
      XPathExpression xpathExpr = xpath.compile(expr);

      // Evaluate the XPath expression against the input document
      result = (NodeList)xpathExpr.evaluate(xml, XPathConstants.NODESET);

      // Print the result to System.out.
      printResult(result);

    } catch (Exception e) { e.printStackTrace(); }
  }
```

Xalan XPath - Java

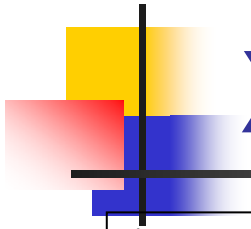
```
  static void printResult(NodeList nodelist) {
    int j = 1;
    for (int i = 0; i < nodelist.getLength(); i++) {
      String str = getNodeString(nodelist.item(i));
      if (!str.equals(""))
        System.out.println((j++) + ":\n" + str);
      }
    }
  }
  static String formatText(String s) {
    String str = s.replaceAll("&","&amp;");
    str = str.replaceAll("'", "&apos;");
    str = str.replaceAll("\"", "&quot;");
    str = str.replaceAll(">","&gt;");
    str = str.replaceAll("<","&lt;");
    return str;
}
```

Xalan XPath - Java

```
static String getNodeListString(NodeList nodelist) {
  String str = "";
  for (int i = 0; i < nodelist.getLength(); i++)
    str += getNodeString(nodelist.item(i));
 }

 static String getNodeString(Node n) {
   switch (n.getNodeType()) {
     case Node.DOCUMENT_NODE:
       return getNodeString((Node)(((Document)n).getDocumentElement()));
     case Node.PROCESSING_INSTRUCTION_NODE:
       return "<?" + n.getNodeName() + " " + n.getNodeValue() + "?>";
     case Node.ATTRIBUTE_NODE:
       return " " + n.getNodeName() + "=\"" + n.getNodeValue() + "\"";
```

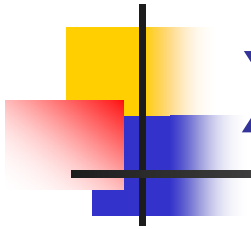
Xalan XPath - Java

```
    case Node.ELEMENT_NODE:
      String str = "<" + n.getNodeName();
      NamedNodeMap attrs = n.getAttributes();
      for (int i = 0; i < attrs.getLength(); i++)
        str += getNodeString(attrs.item(i));
      str += ">\n”;
      str += getNodeListString(n.getChildNodes());
      str += "</" + n.getNodeName() + ">\n";
      return str;
    case Node.TEXT_NODE:
      String t = n.getNodeValue().trim();
      if (!t.equals(""))
        return formatText(t);
      return "";
  } } }
```



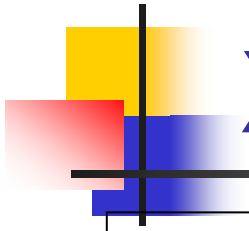
Xalan XPath - Java

```
$ java tutor6 book.xml /book//title[..//price]
1:
<title>
TCP/IP Illustrated
</title>

$ java tutor6 h.xml /a/b
1:
<b>
Hello
</b>
2:
<b>
World
</b>
```



Xalan XPath & DOM

- It returns a list of DOM Nodes
- Just print them all (as in assign1)

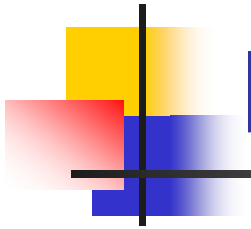


Xalan XPath & DOM Parser

```
DocumentBuilderFactory dfactory =
                        DocumentBuilderFactory.newInstance();
dfactory.setNamespaceAware(true);
Document doc = dfactory.newDocumentBuilder().parse(in);
XPathEvaluator evaluator = new XPathEvaluatorImpl(doc);

XPathNSResolver resolver = evaluator.createNSResolver(doc);

// Evaluate the xpath expression
XPathResult result =
  (XPathResult)evaluator.evaluate(xpath, doc, resolver,
              XPathResult.UNORDERED_NODE_ITERATOR_TYPE, null);

printResult(result);
```

Xalan XPath & DOM Parser

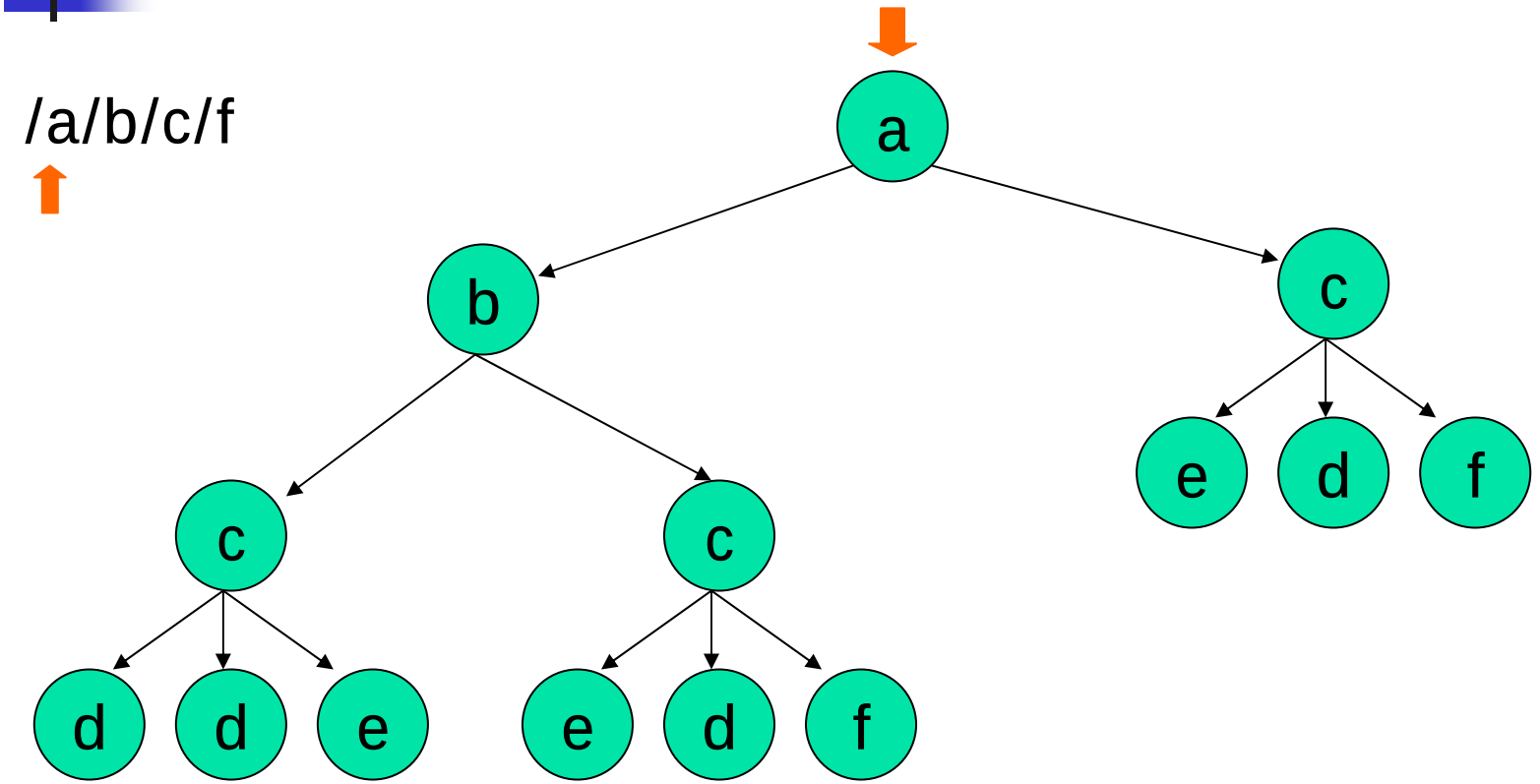
```
void printResult(XPathResult result) {
  Node n;
  int i=1;
  while ((n = result.iterateNext())!= null) {
    String str = getNodeString(n);
    if (!str.equals(""))
      System.out.println((i++) + ":\n" + str);
    }
  }
}
```



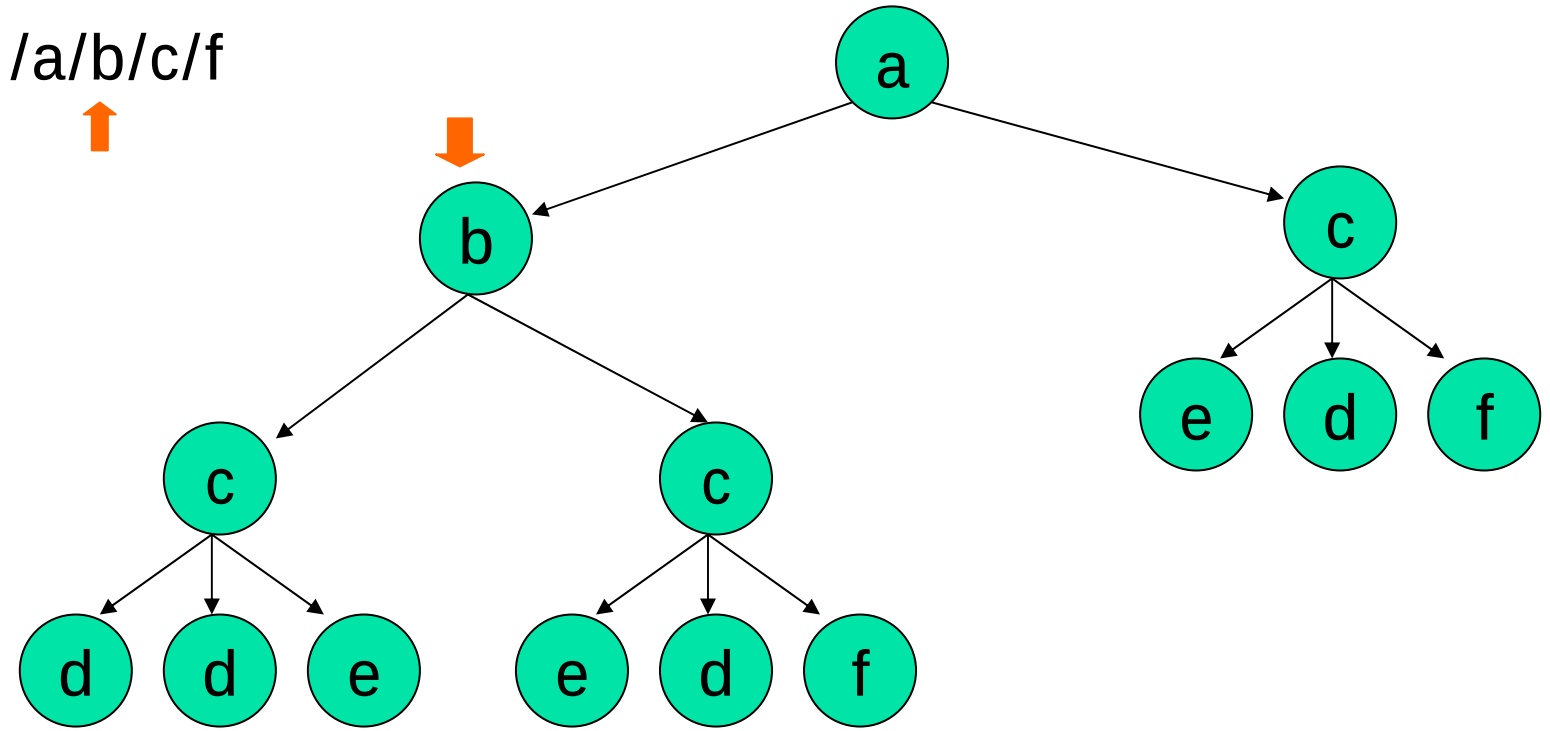
For assignment 4

- Use tutor6.java as a tester
- Write yourself a XPath evaluator
- Compare your result with tutor6.java
- References
  - http://xml.apache.org/xalan-j/
  - http://xml.apache.org/xalan-c/

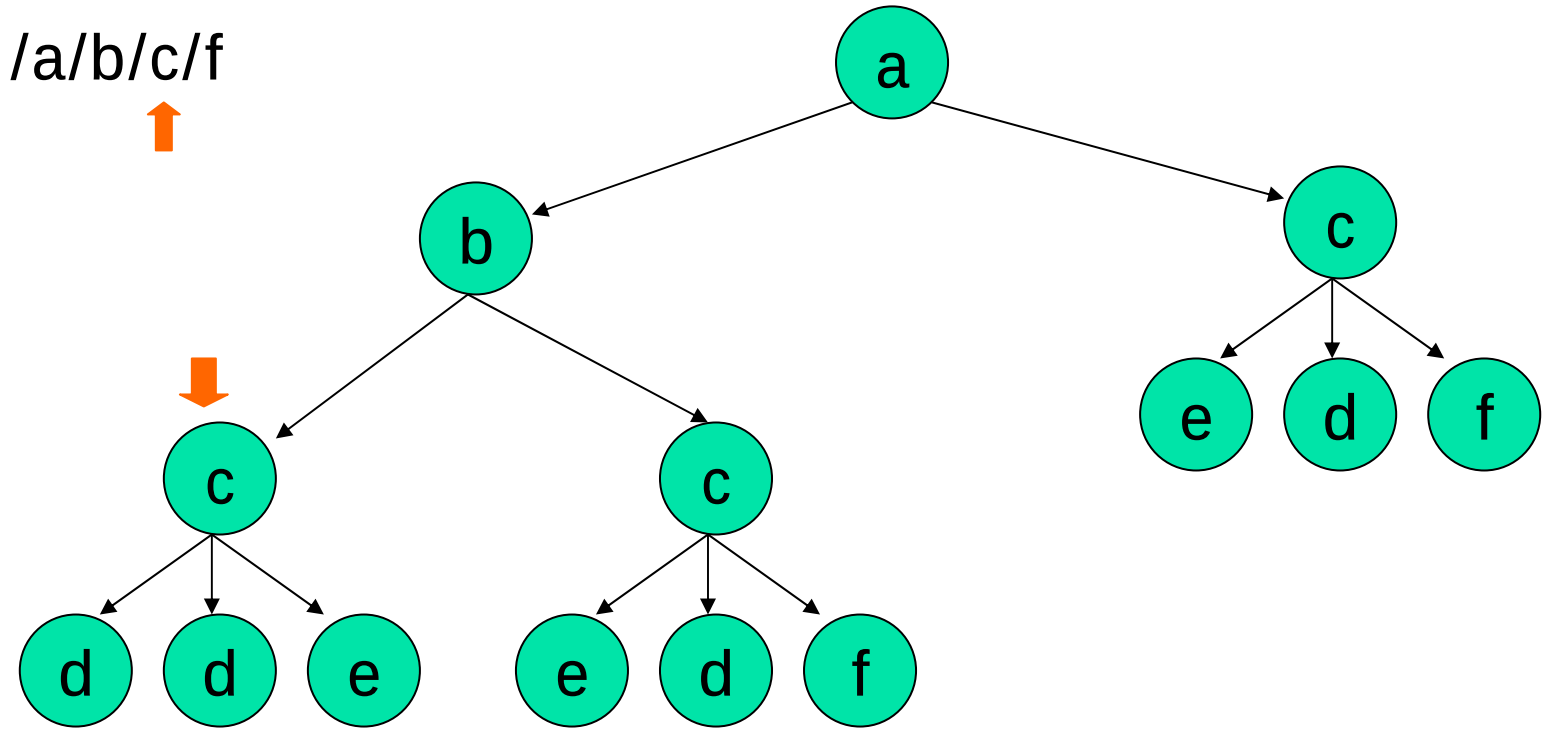
Simple exp on unranked Tree



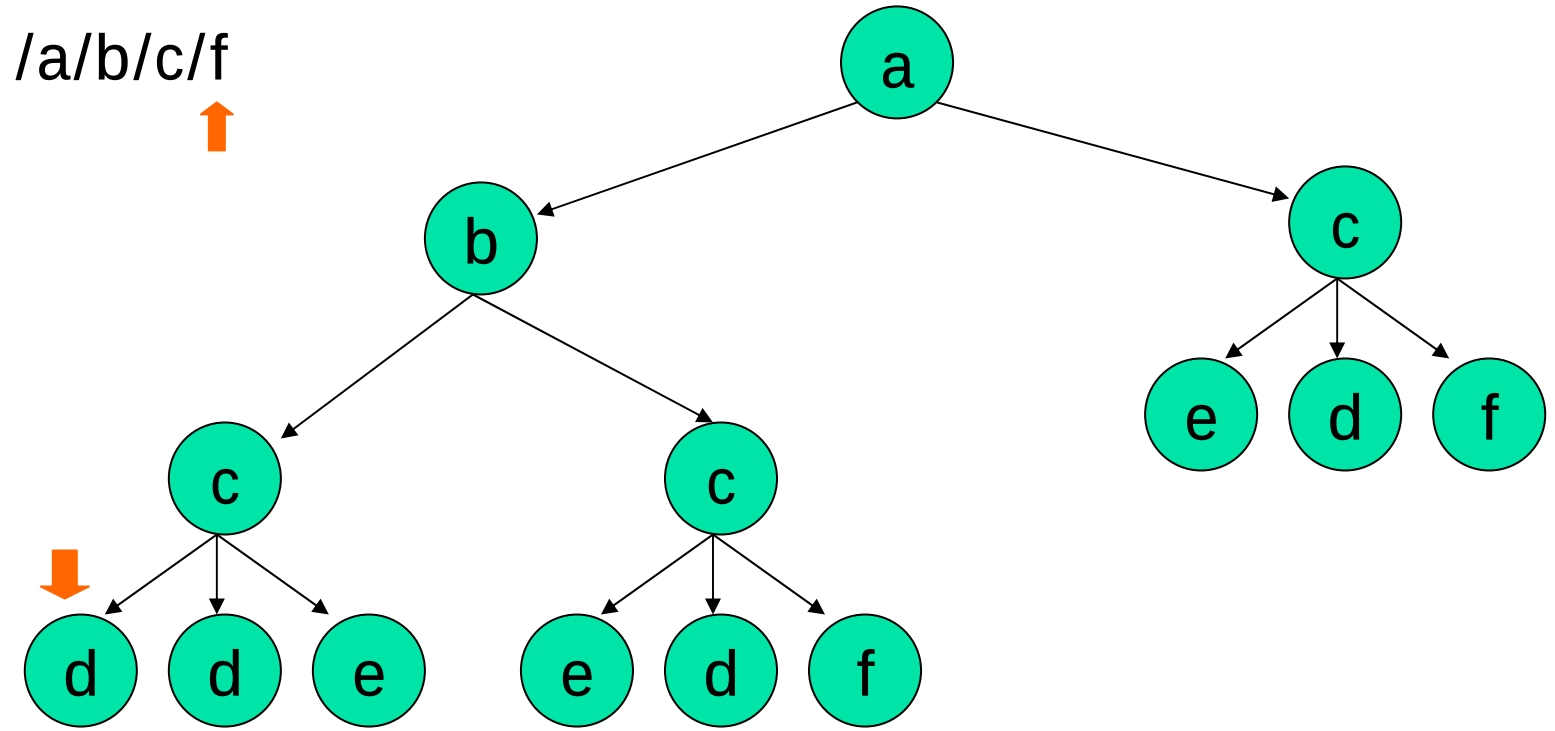
Simple exp on unranked Tree



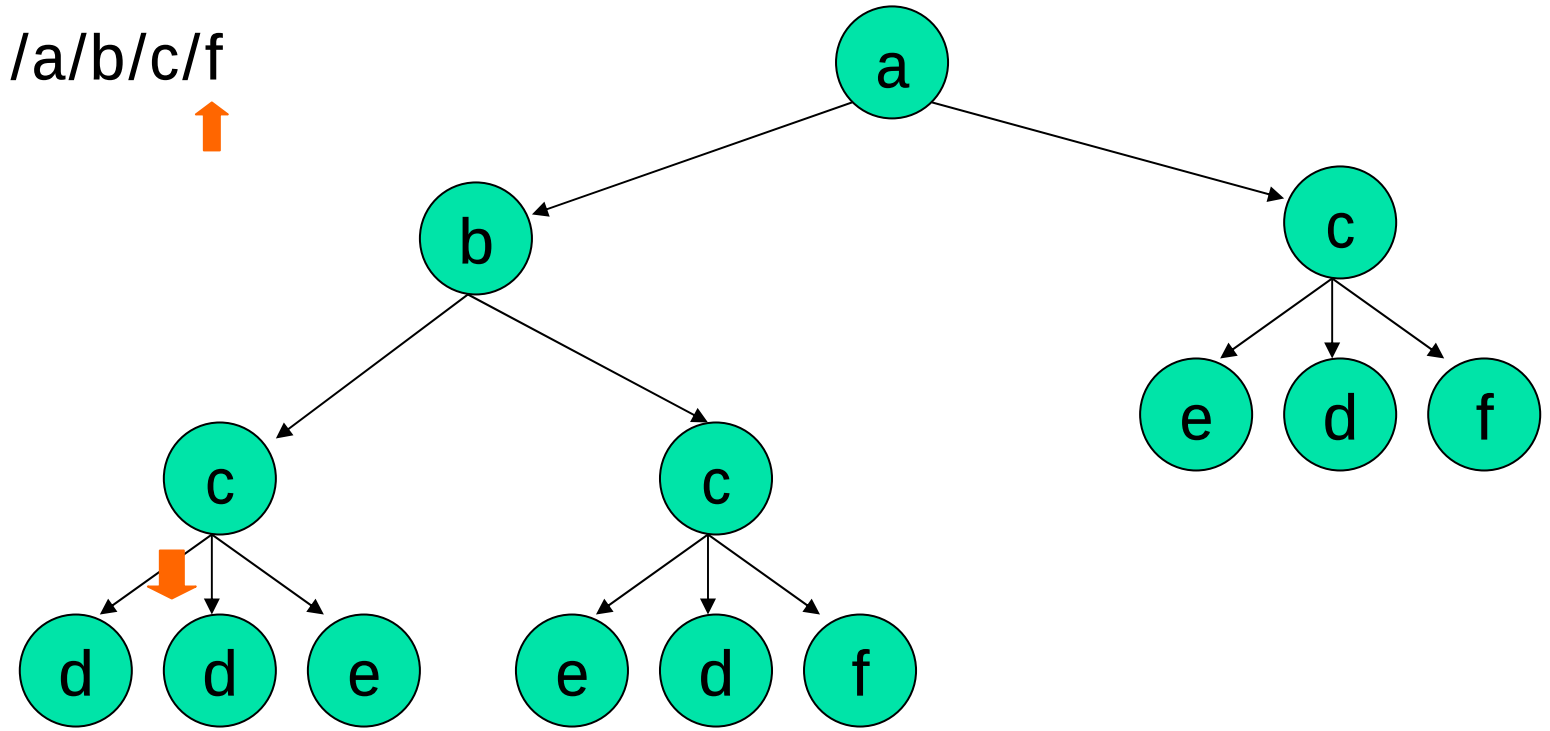
Simple exp on unranked Tree



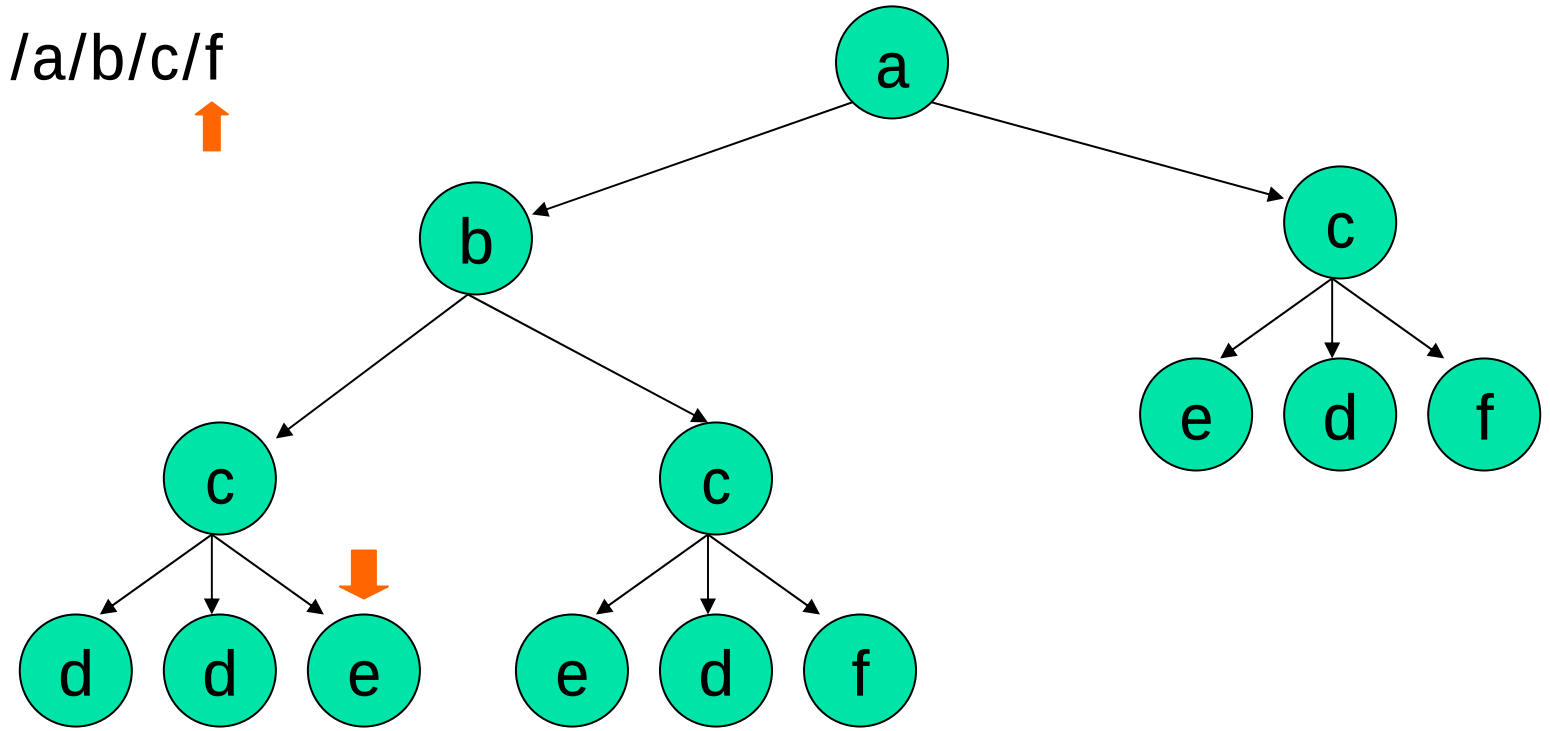
Simple exp on unranked Tree



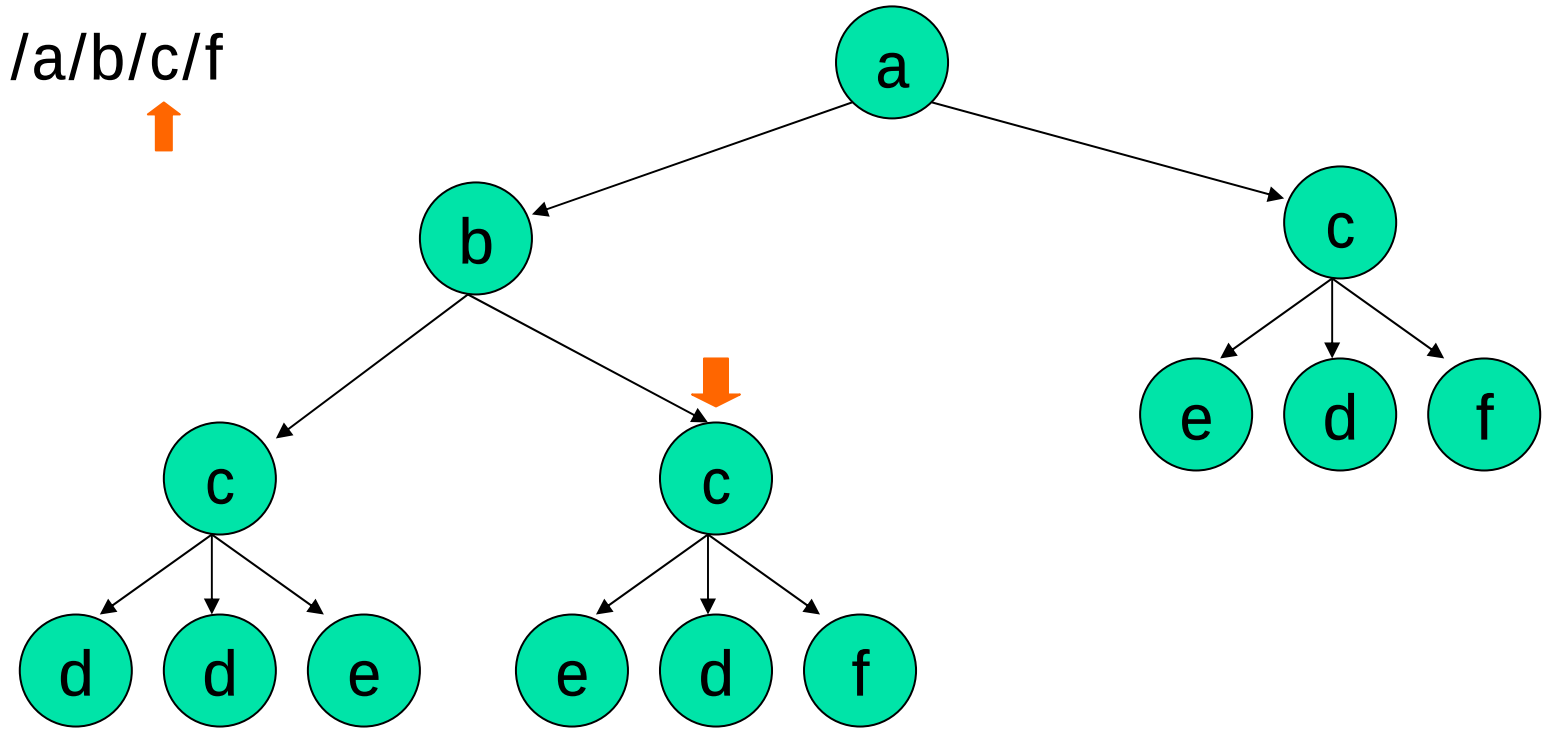
Simple exp on unranked Tree



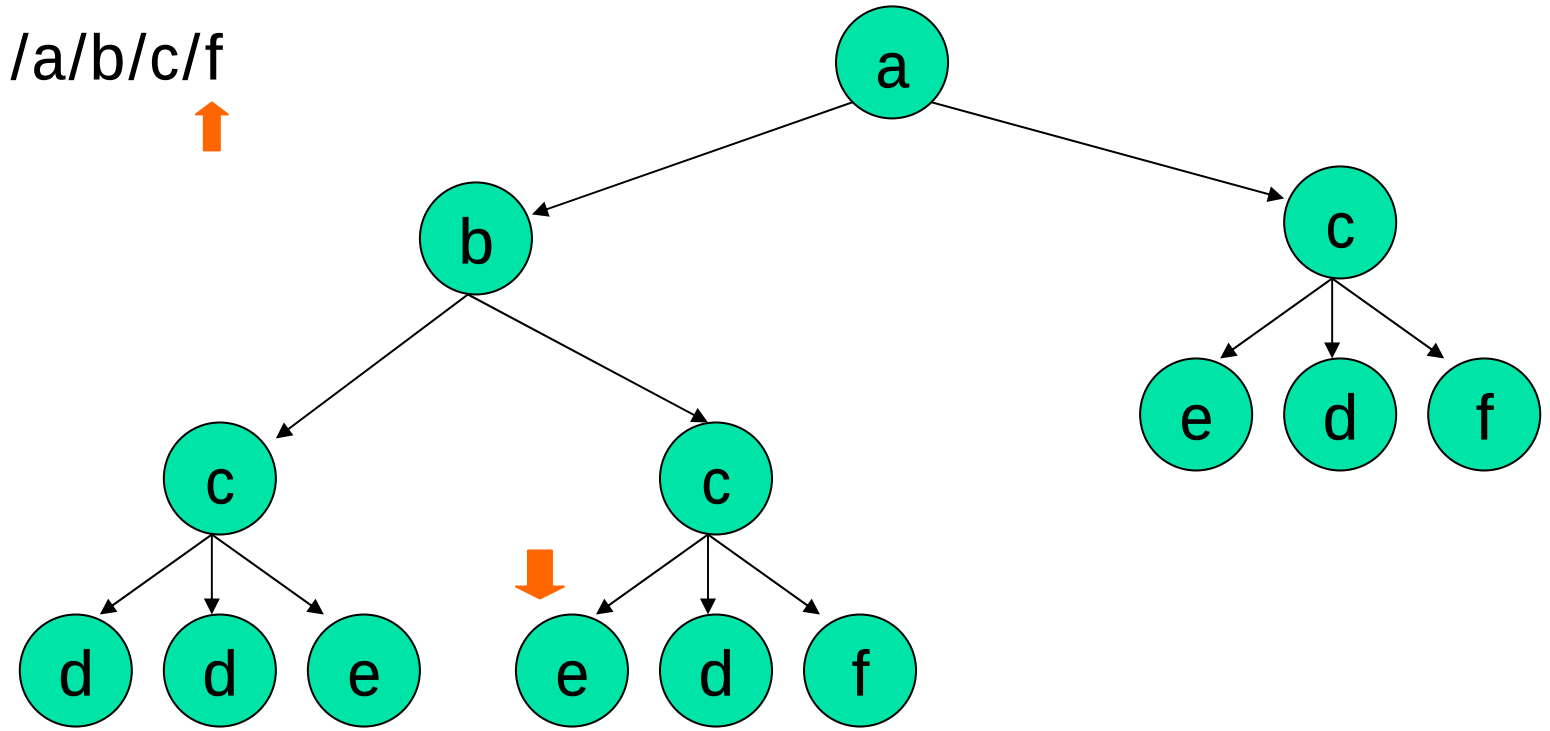
Simple exp on unranked Tree



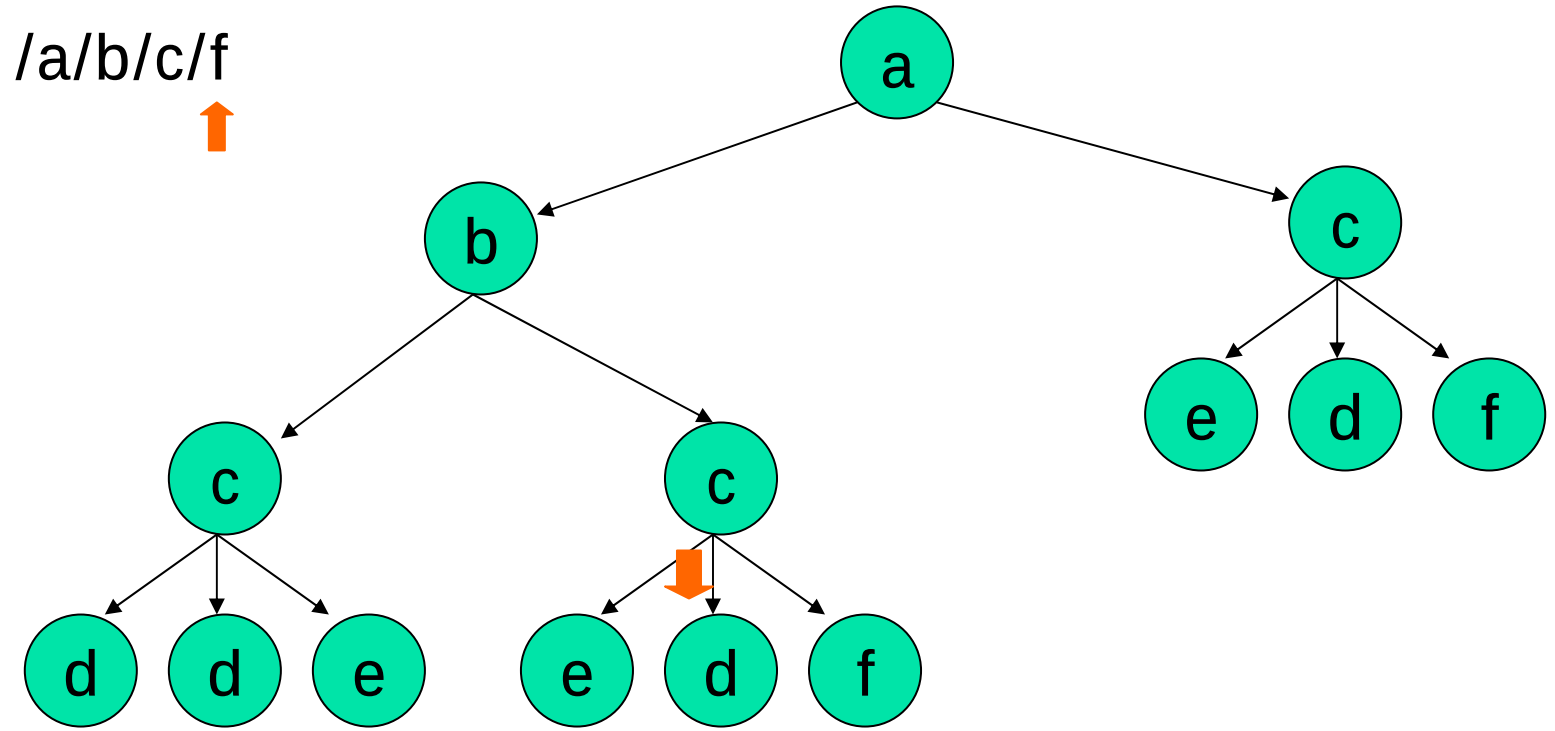
Simple exp on unranked Tree



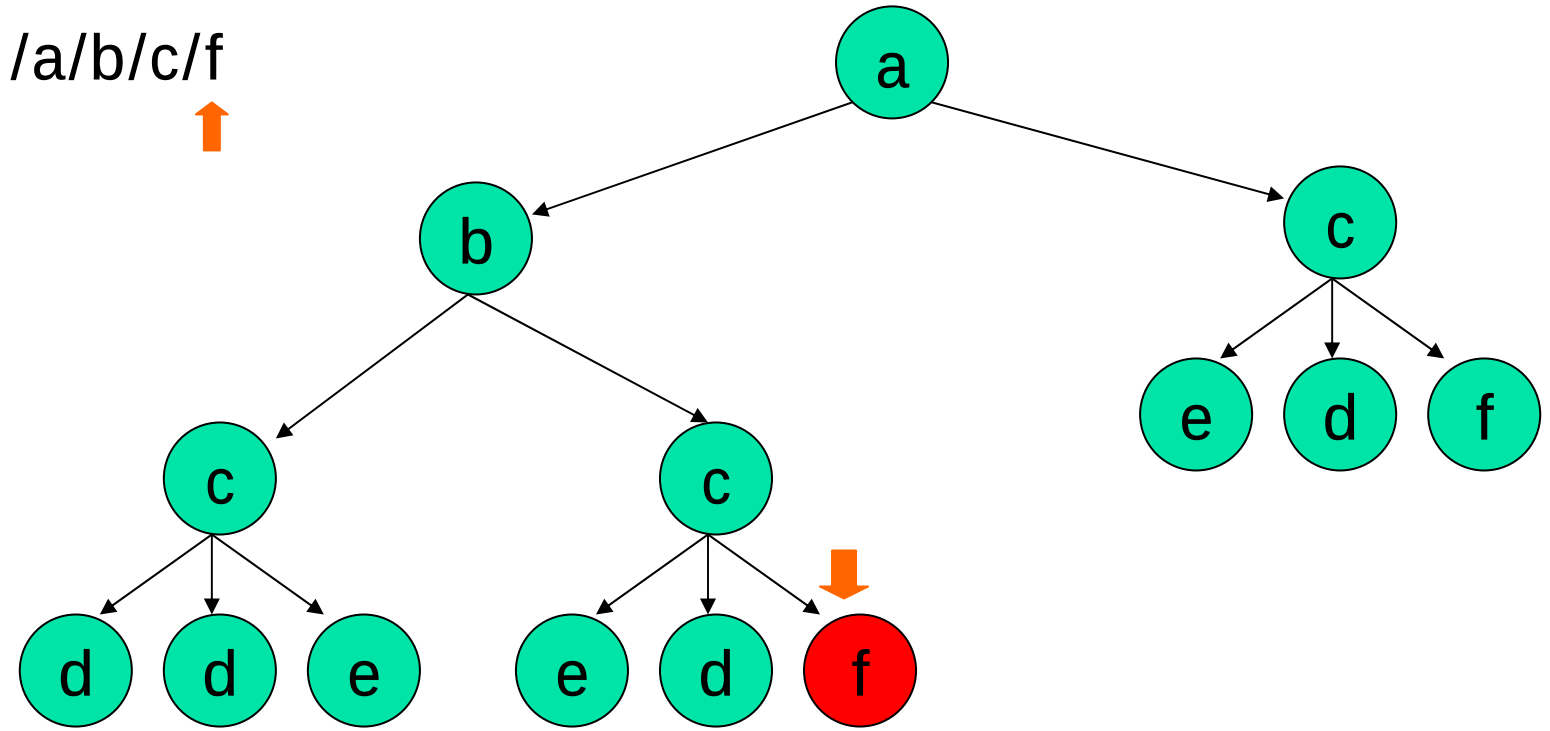
Simple exp on unranked Tree



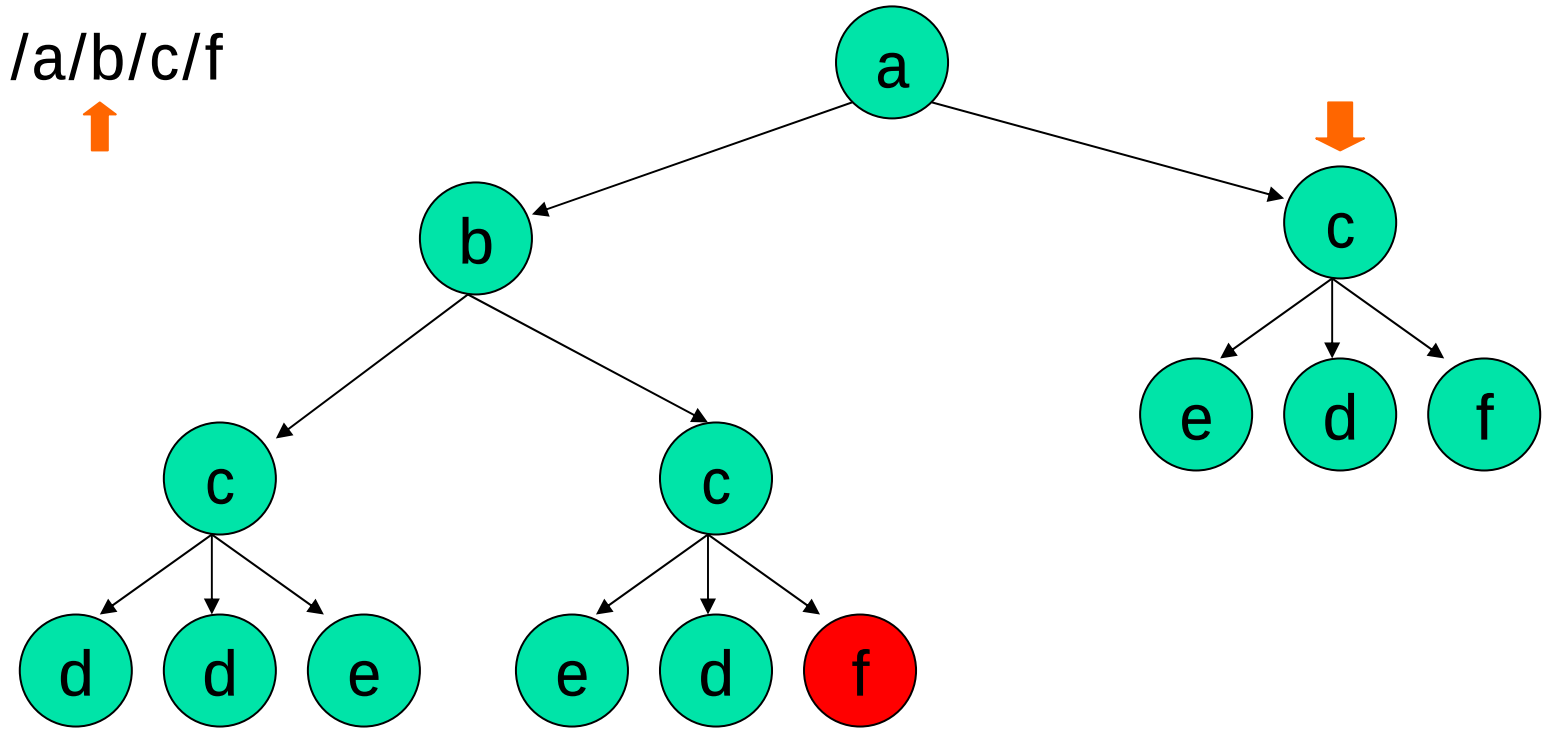
Simple exp on unranked Tree



Simple exp on unranked Tree

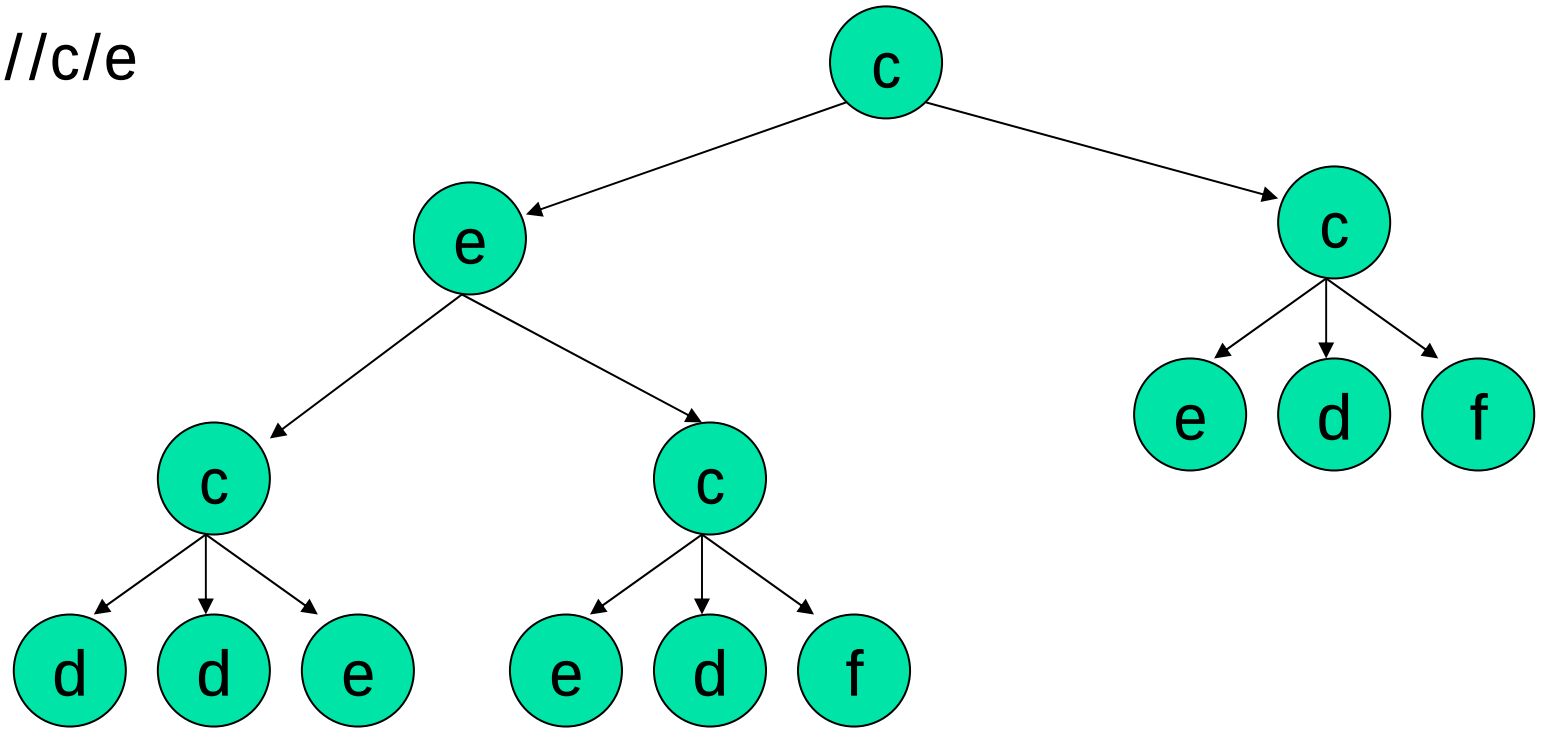


Simple exp on unranked Tree

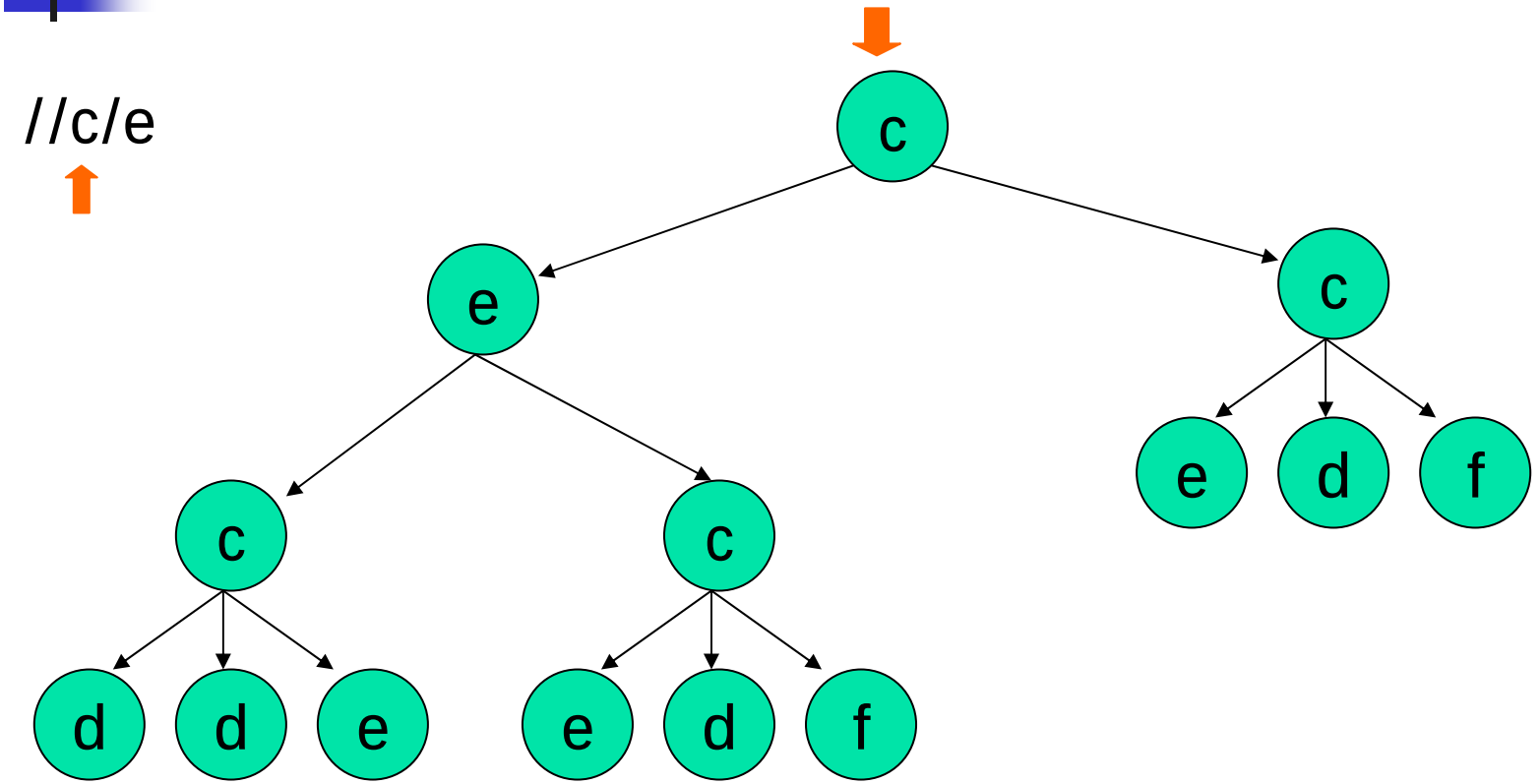


SS Exp on unranked Tree

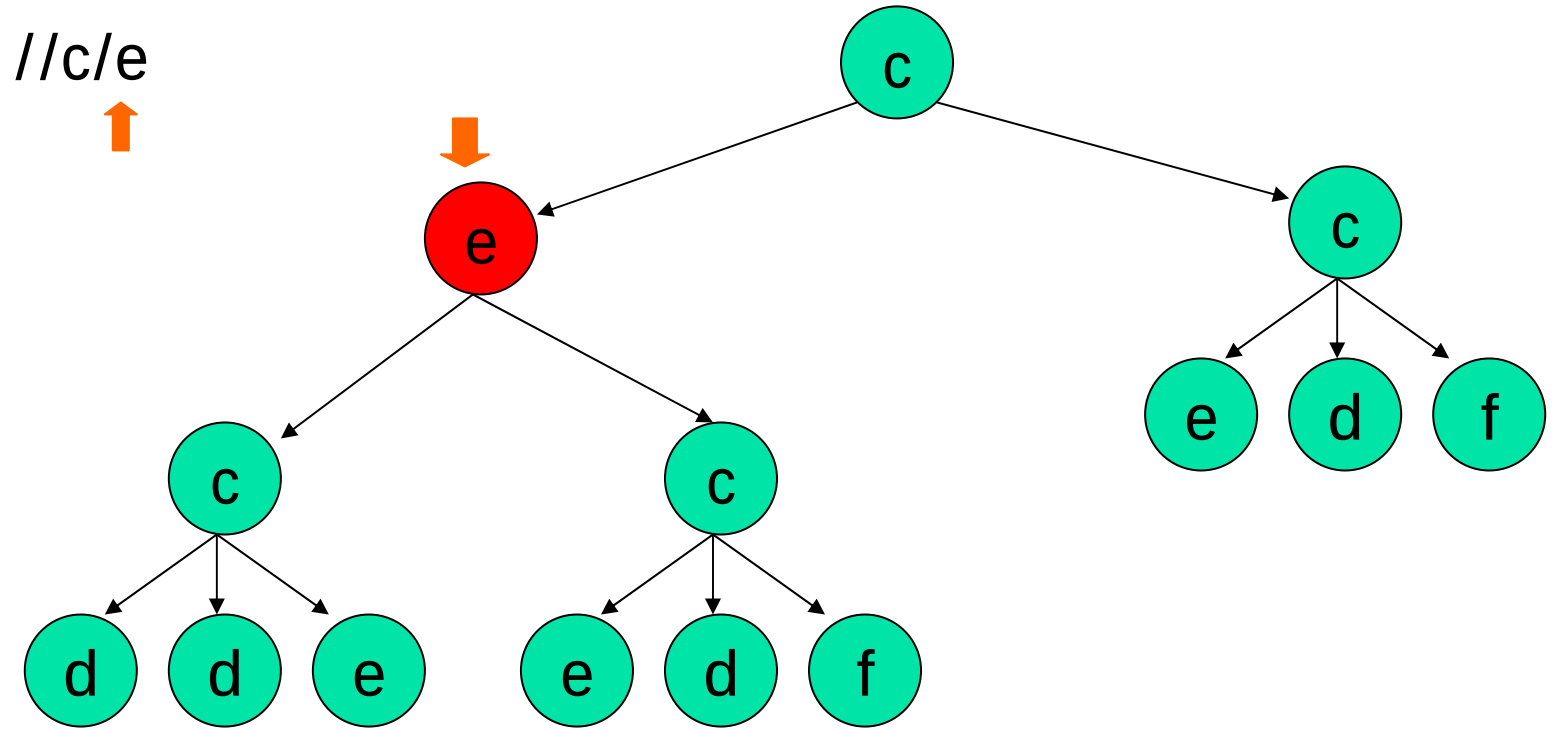
//c/e



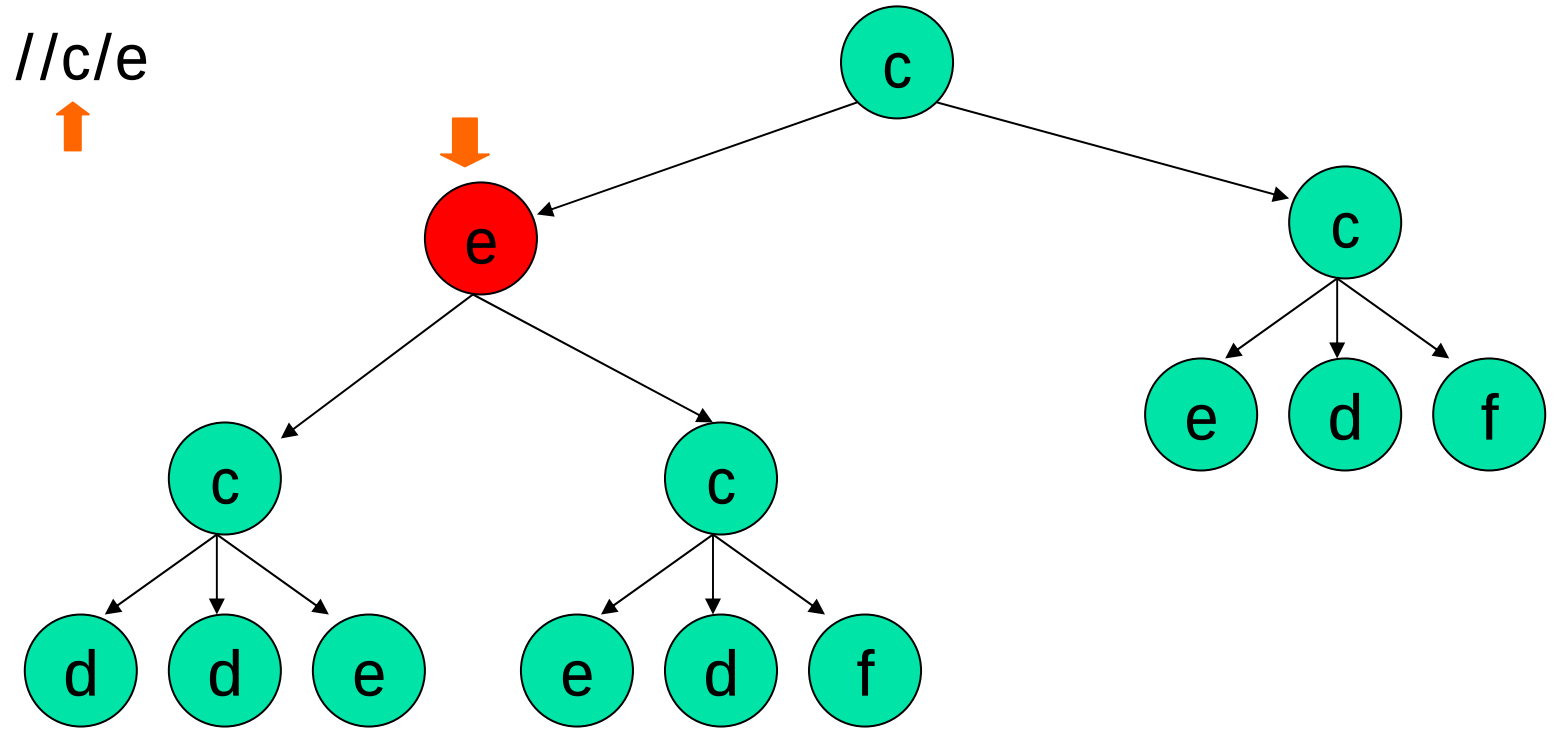
SS Exp on unranked Tree



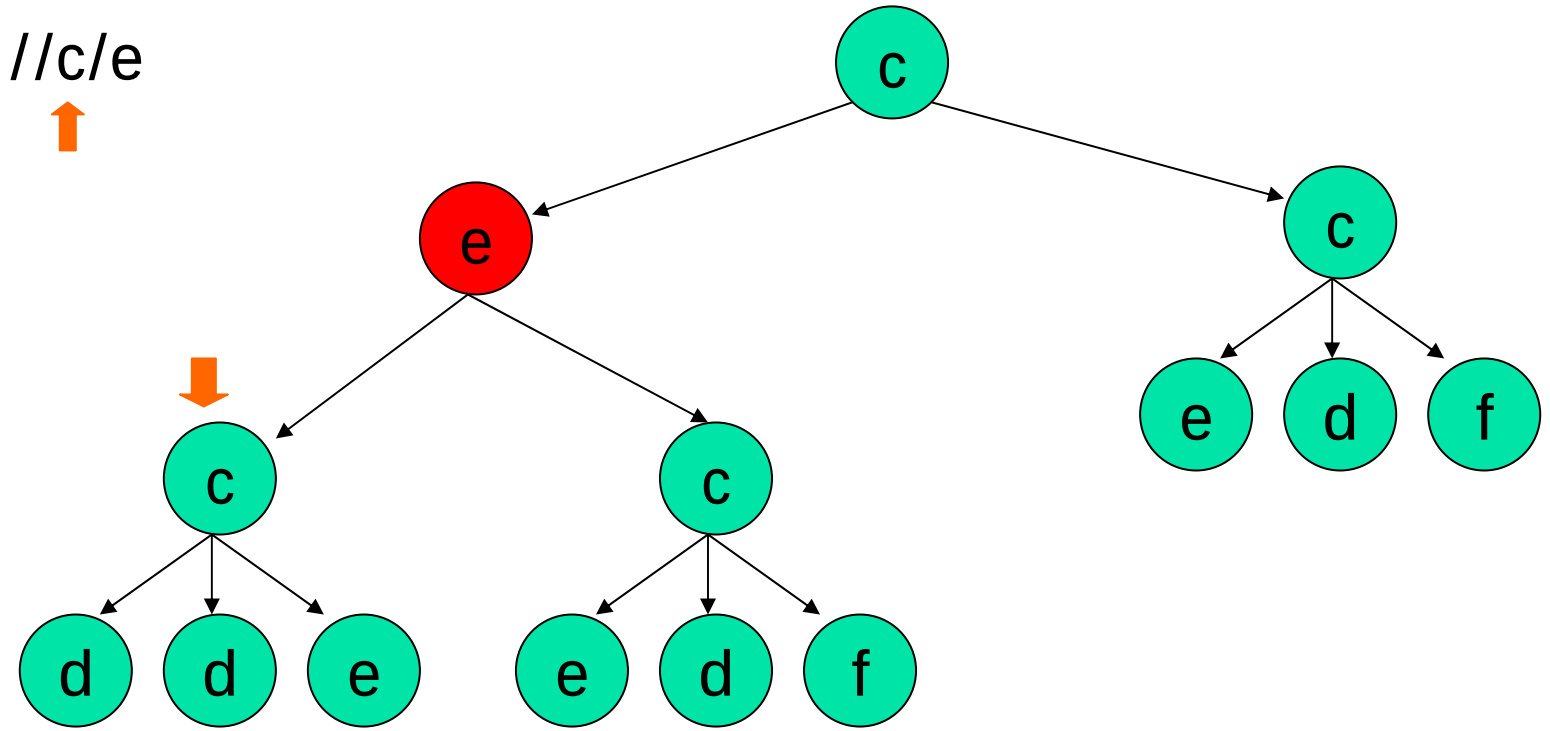
SS Exp on unranked Tree



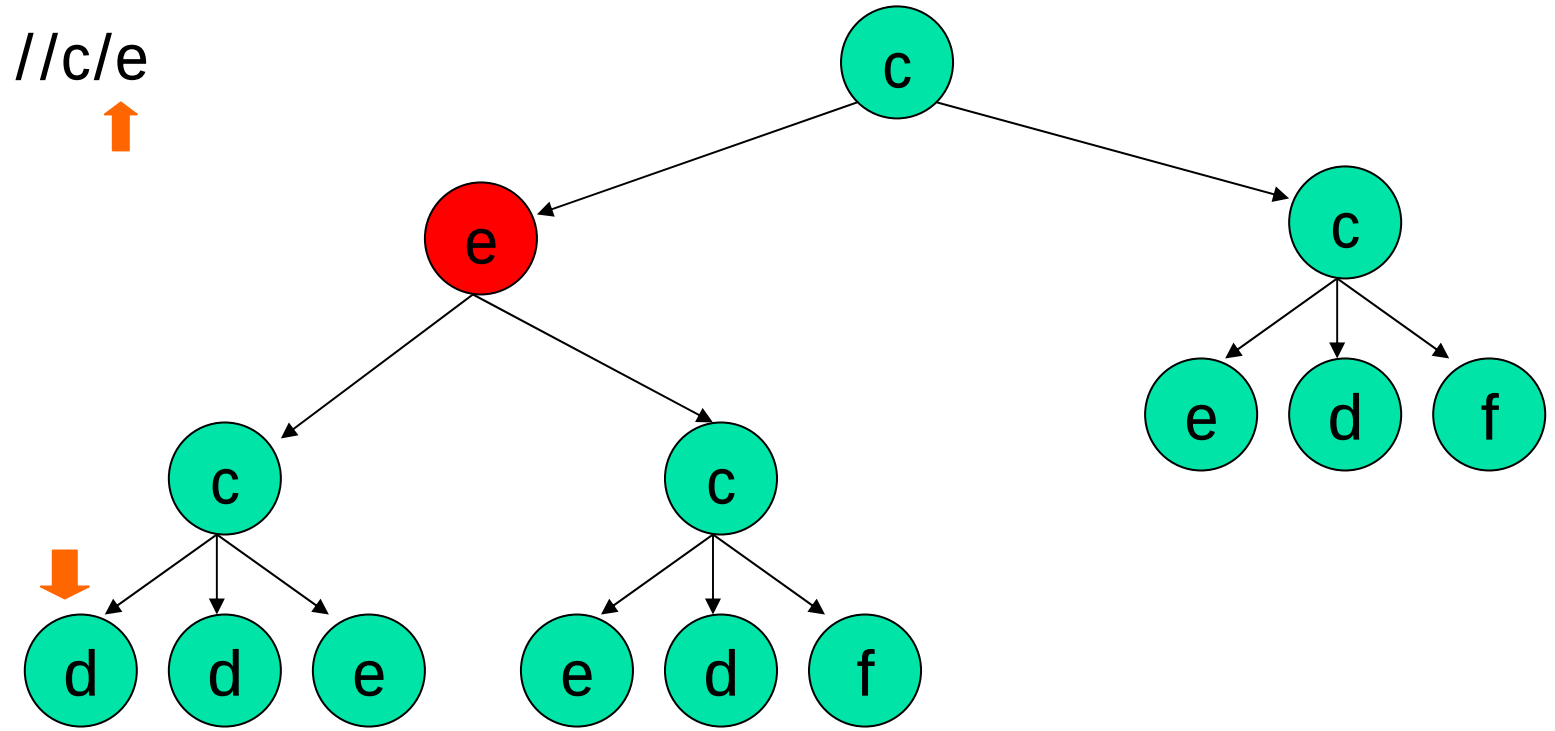
SS Exp on unranked Tree



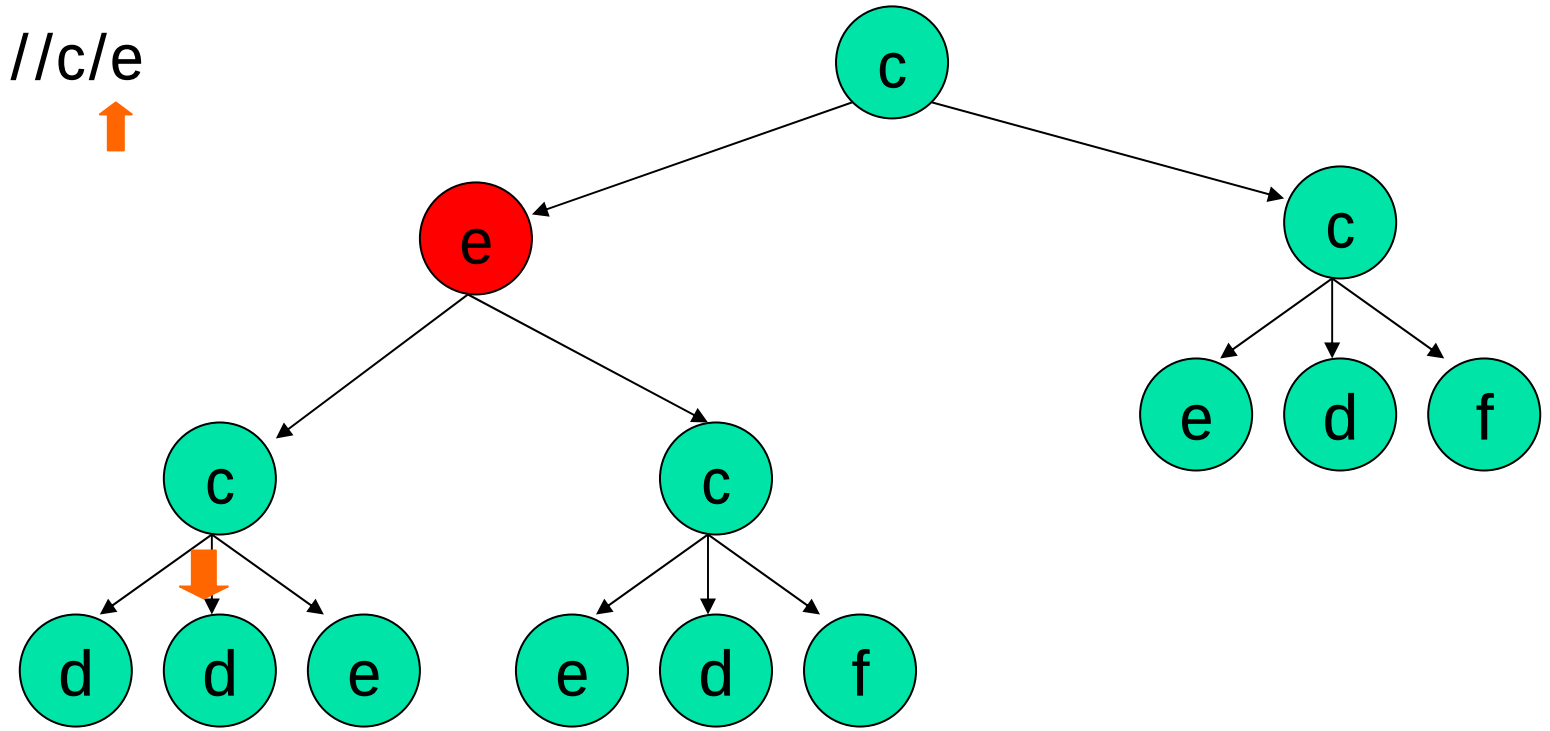
SS Exp on unranked Tree



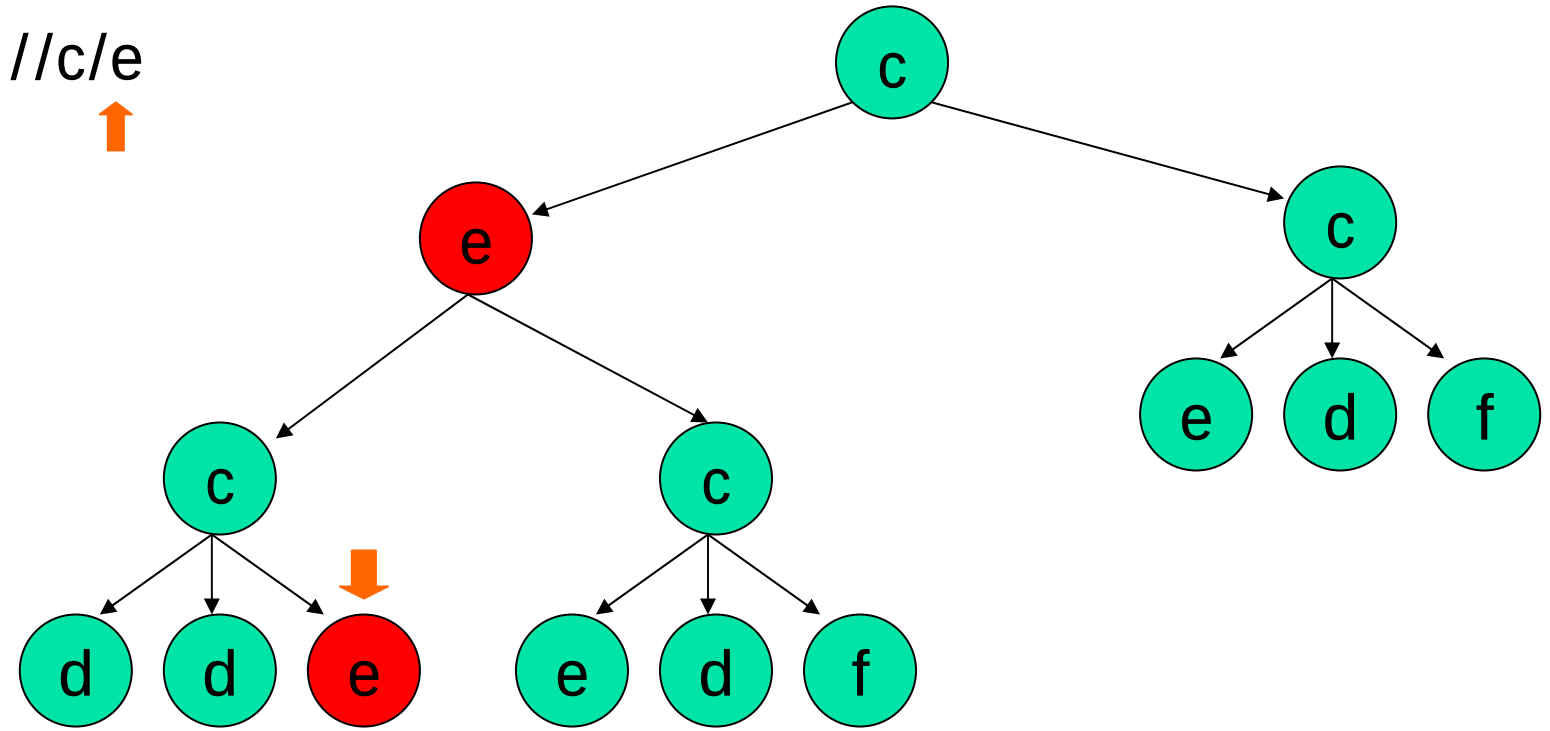
SS Exp on unranked Tree



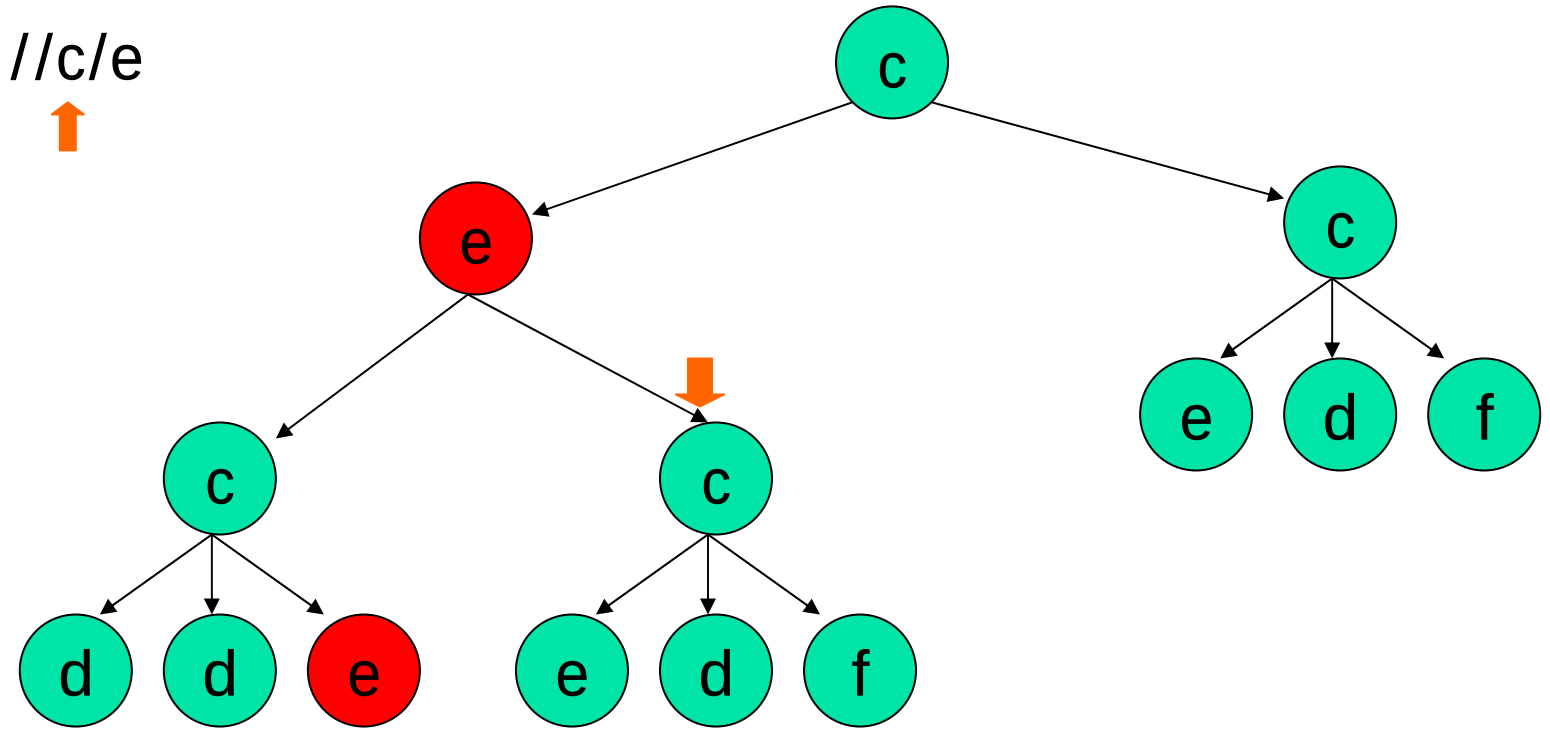
SS Exp on unranked Tree



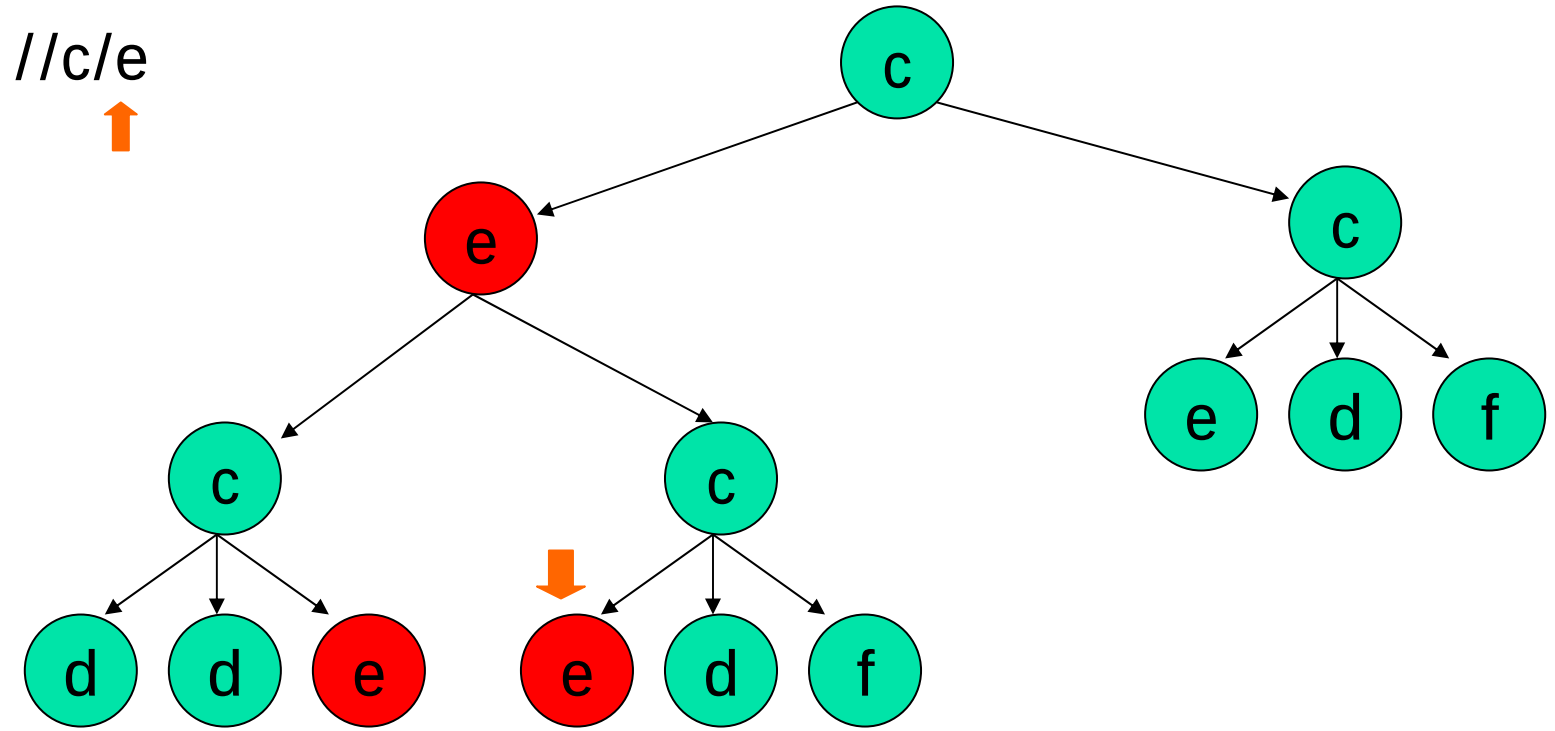
SS Exp on unranked Tree



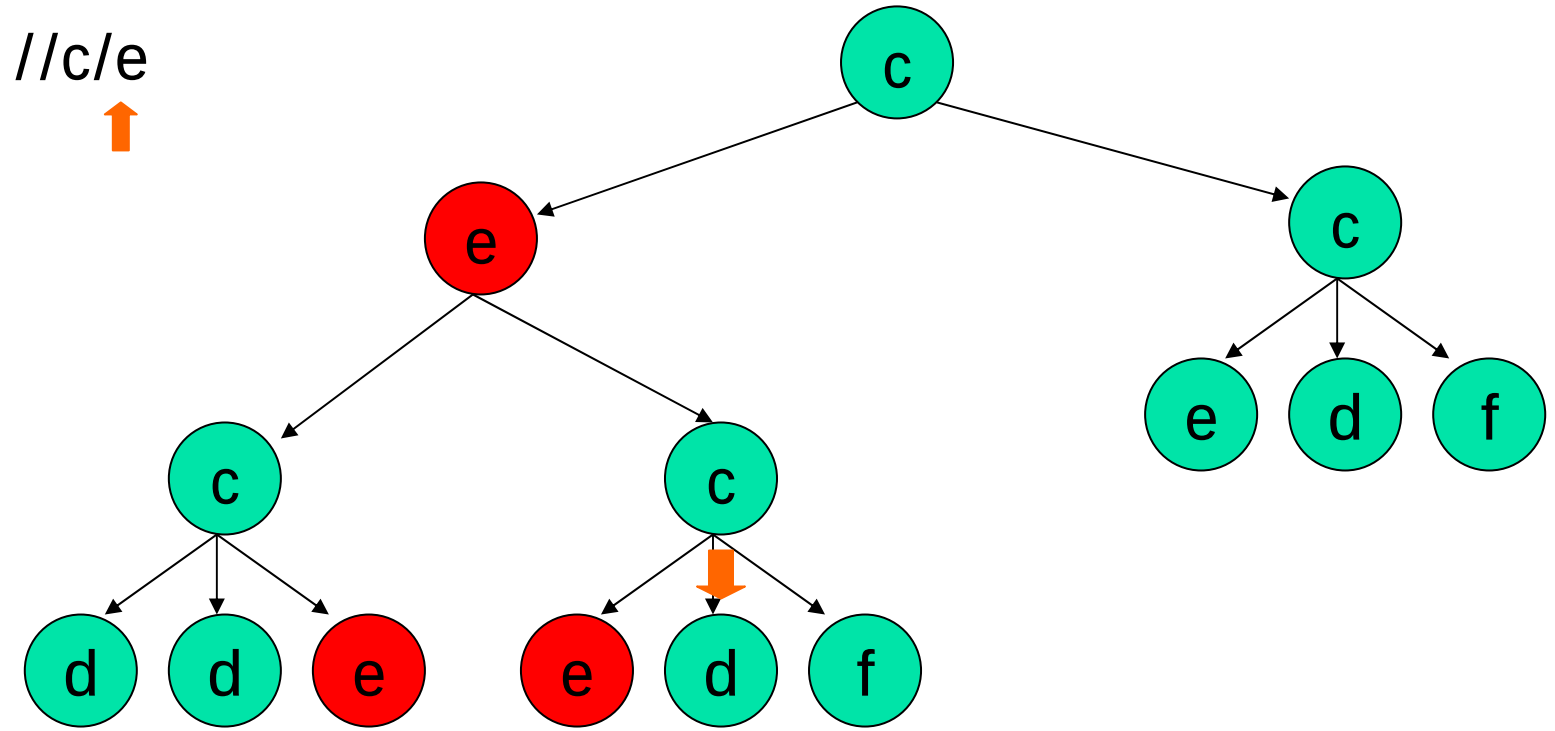
SS Exp on unranked Tree



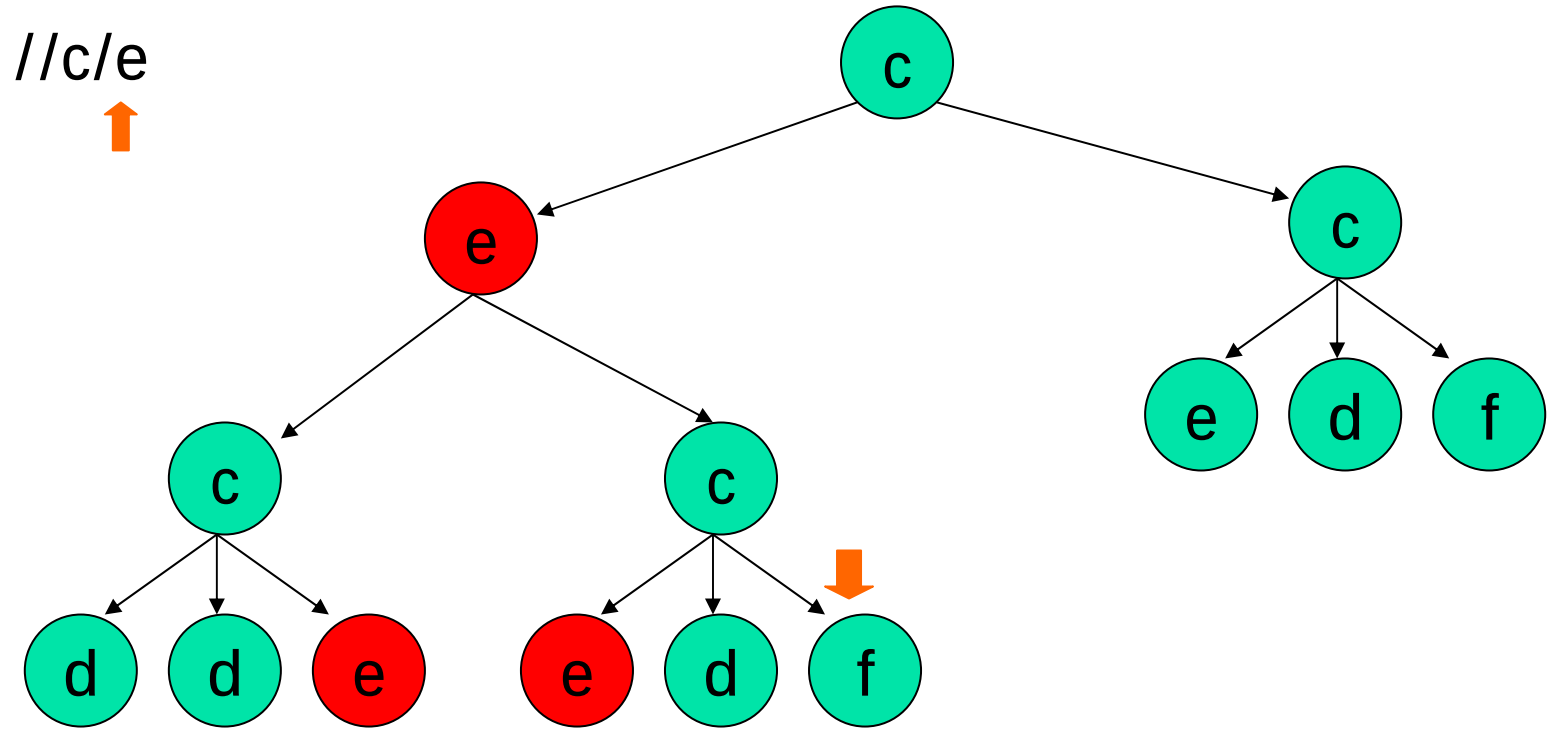
SS Exp on unranked Tree



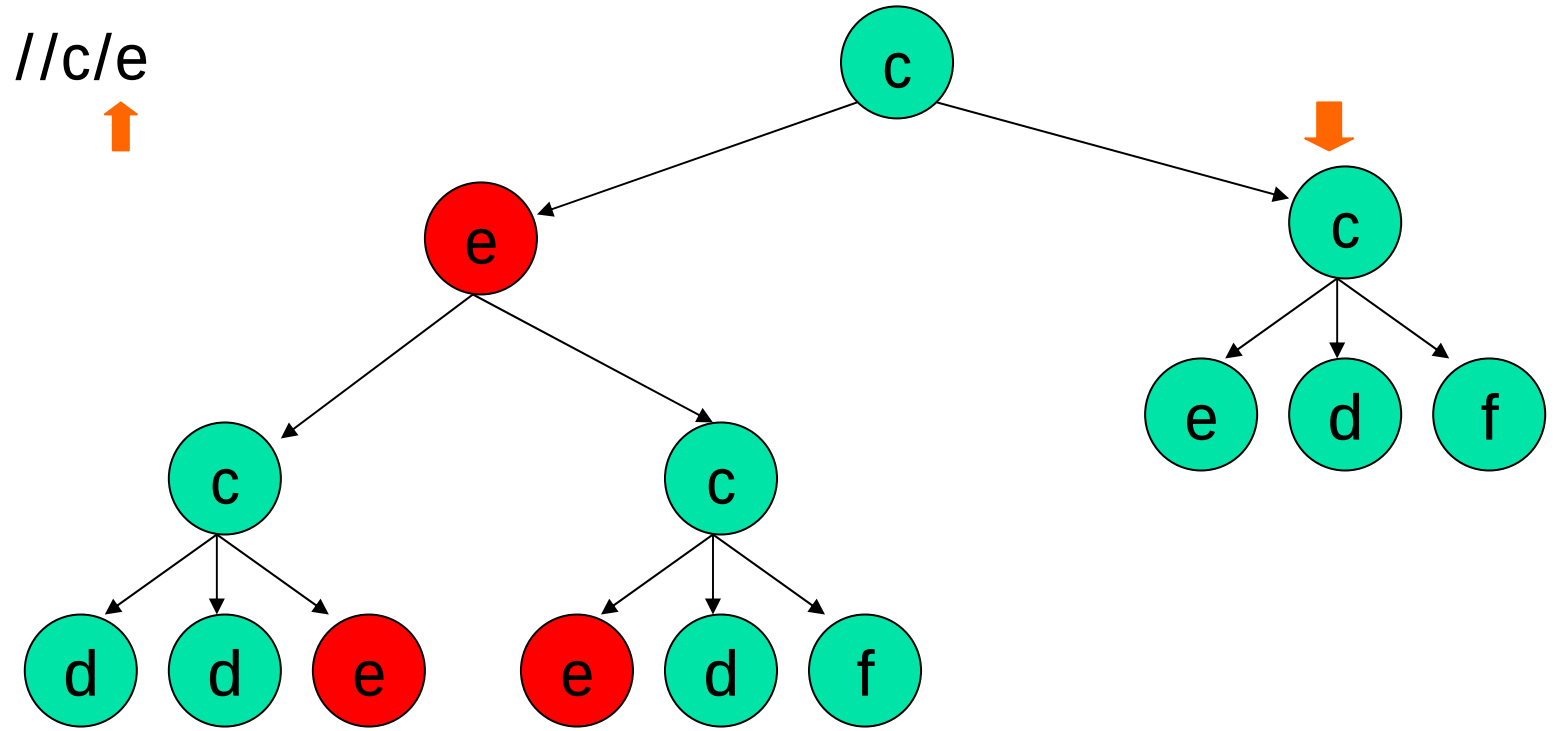
SS Exp on unranked Tree



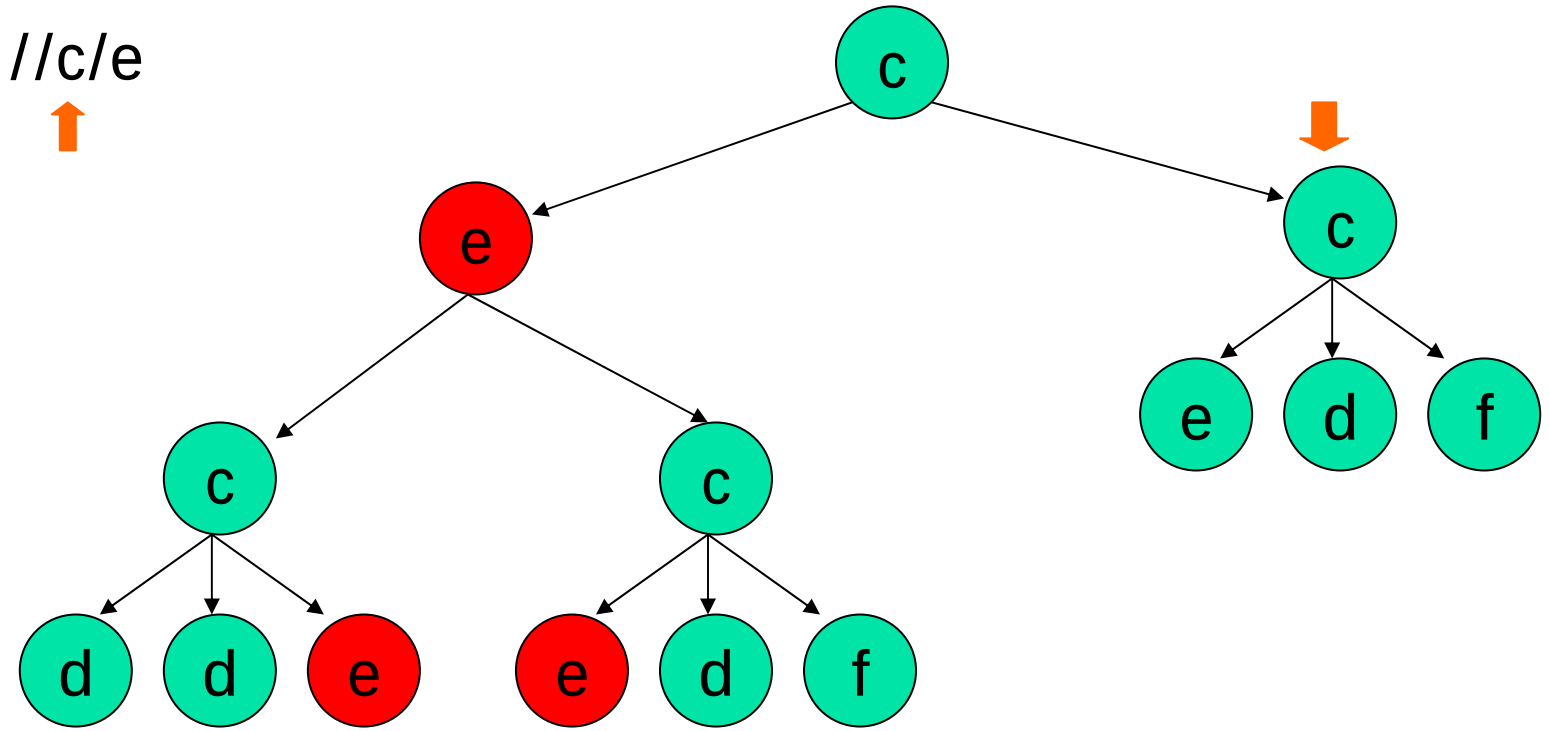
SS Exp on unranked Tree



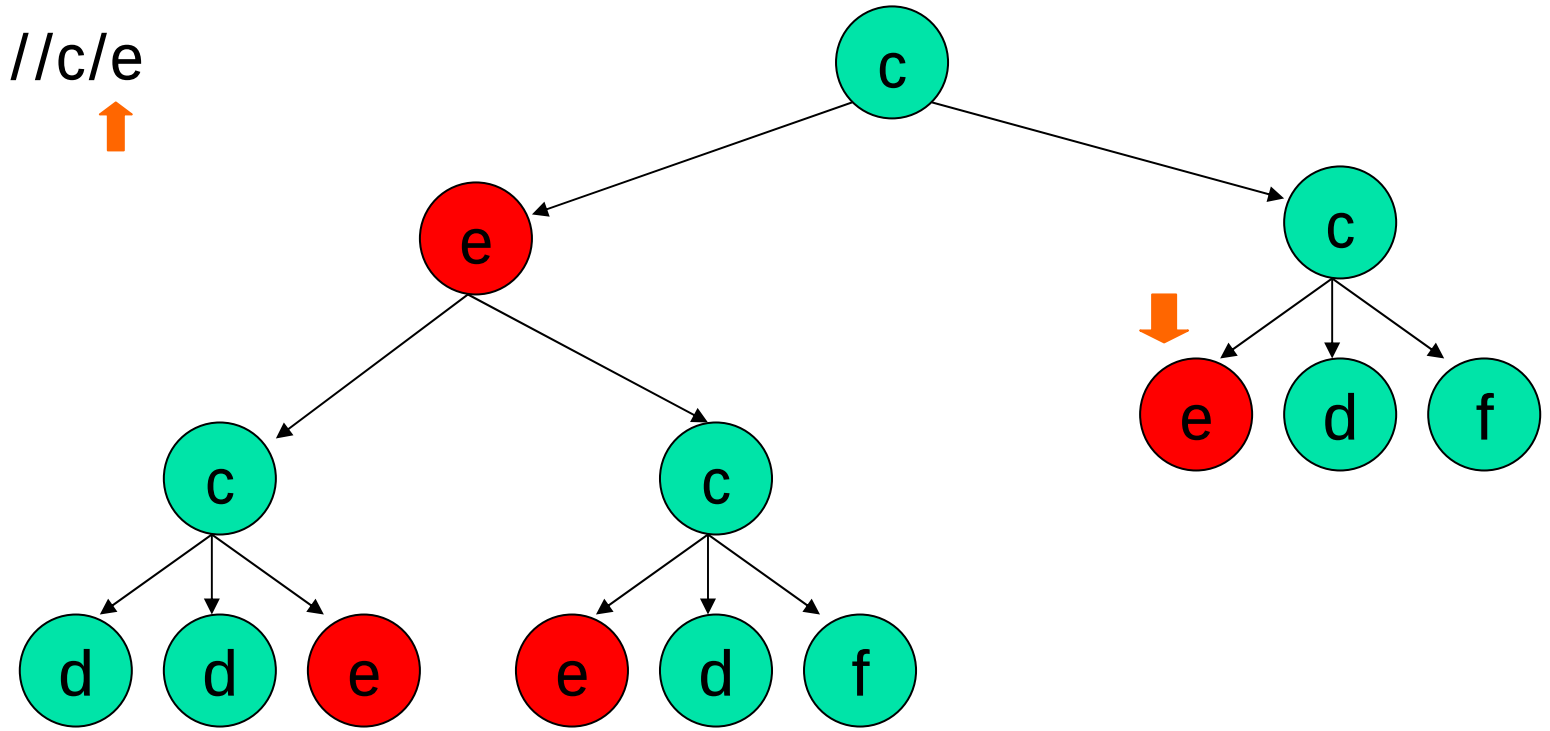
SS Exp on unranked Tree



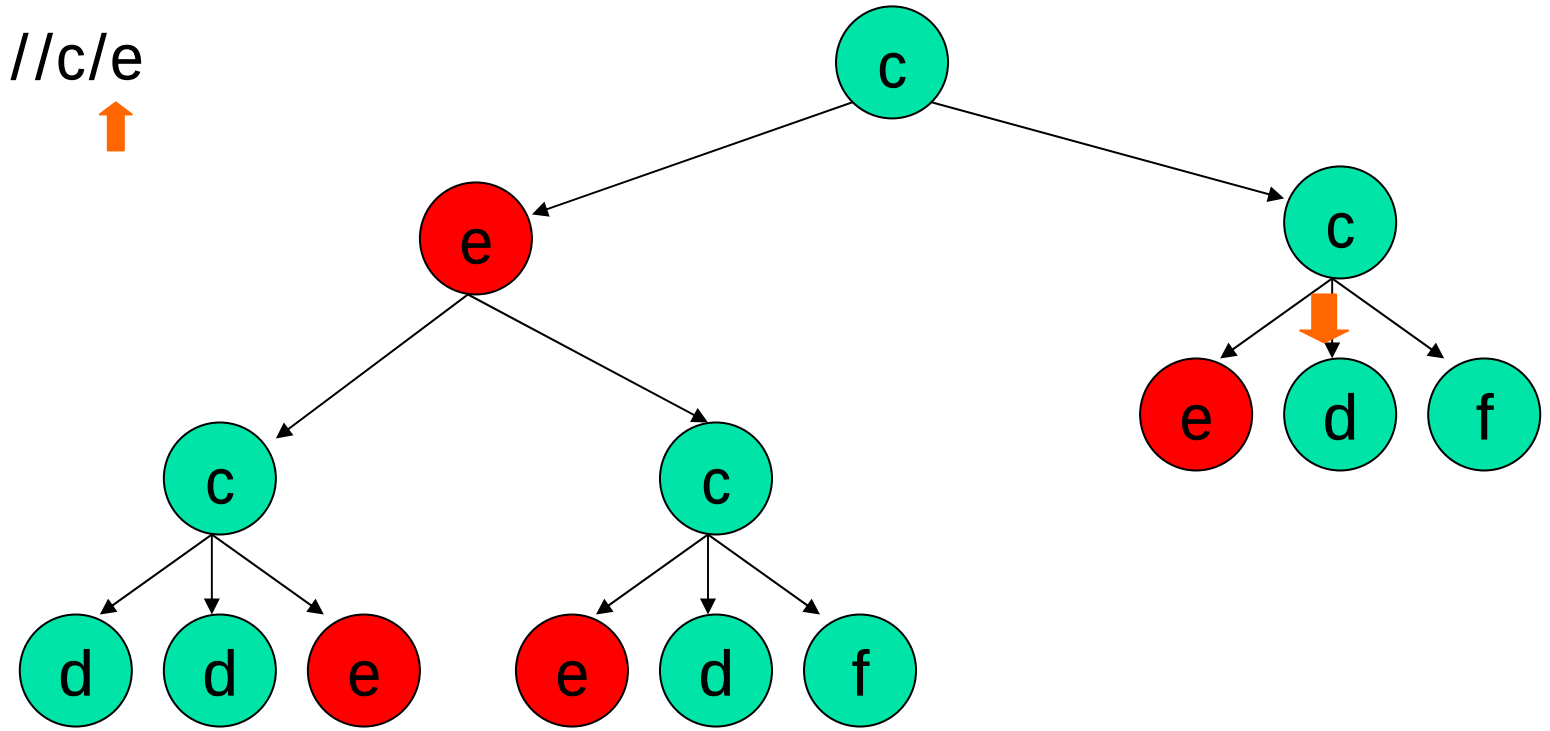
SS Exp on unranked Tree



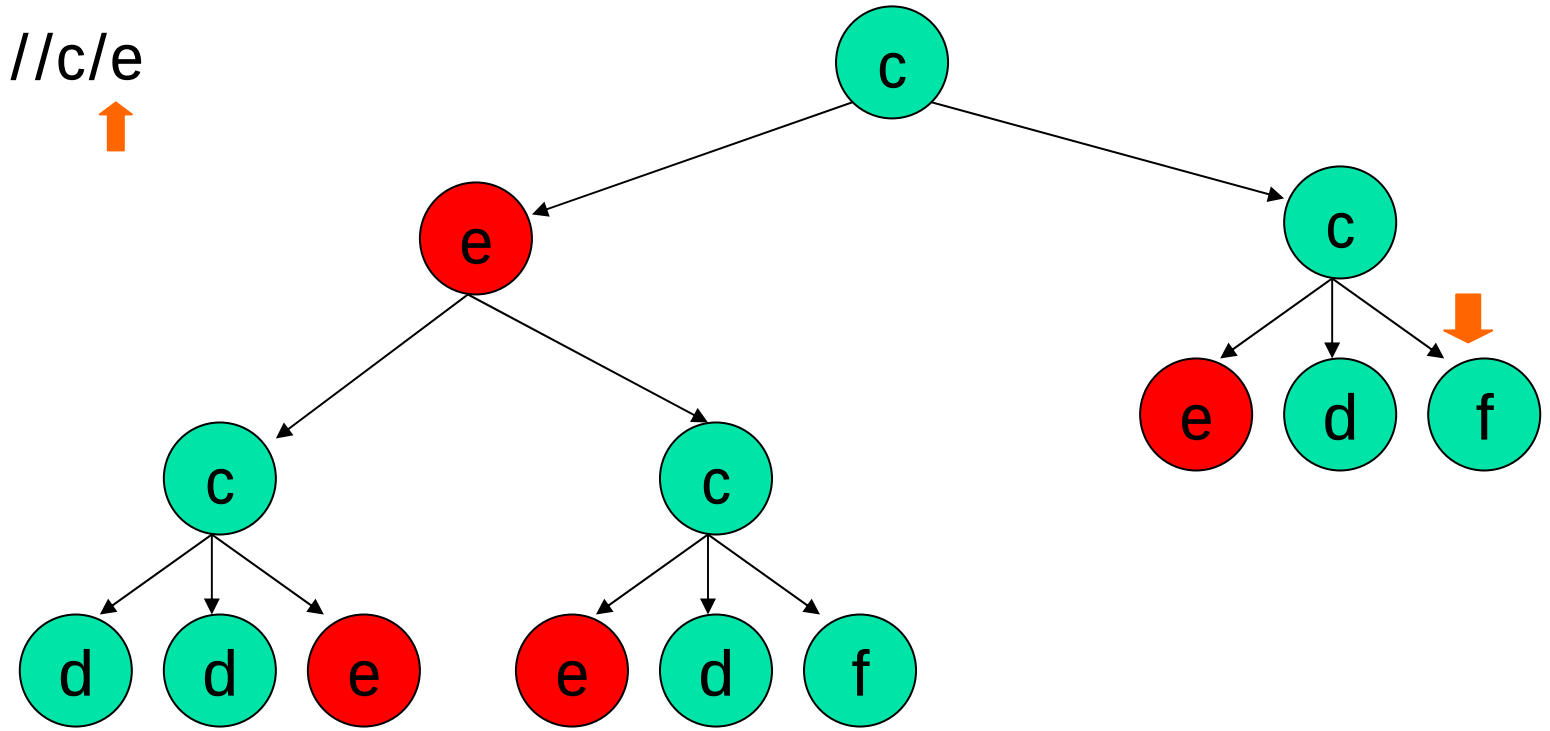
SS Exp on unranked Tree

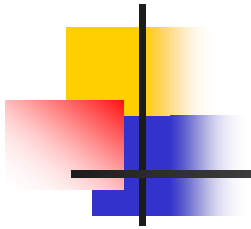


SS Exp on unranked Tree



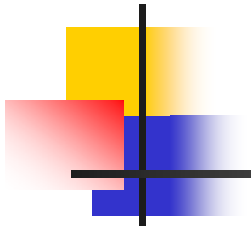
SS Exp on unranked Tree





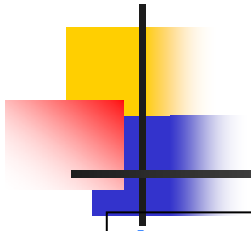
Input stream vs. file stream

- SAX Parser
  - File stream: XMLReaderFactory.createXMLReader(...).parse(filename)
  - Input stream: XMLReaderFactory.createXMLReader(...).parse(new InputSource(System.in))



Input stream vs. file stream

- SAX Parser
  - File stream:
    - Java: java YourClass filename ...
    - C++: ./YourProgram filename ...
  - Input stream:
    - Java: cat filename | java YourClass ...
    - C++: cat filename | ./YourPorgram ...



Input stream vs. file stream

```
import javax.xml.xpath.*;
import org.xml.sax.*;
import org.w3c.dom.*;

public class tutor6_new {
  public static void main(String[] args)
  {
    InputSource xml = new InputSource(System.in);
    String expr = args[0];

    // Create a new XPath
    XPathFactory factory = XPathFactory.newInstance();
    XPath xpath = factory.newXPath();
```

Input stream vs. file stream

```
$ cat book.xml | java tutor6_new "/book//title[..//price]"
1:
<title>
TCP/IP Illustrated
</title>

$ cat h.xml | java tutor6_new "/a/b"
1:
<b>
Hello
</b>
2:
<b>
World
</b>
```

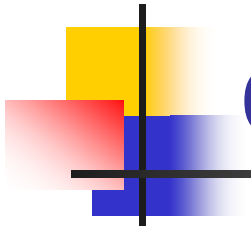
character event in SAX Parser

- Thanks to Zoran Petkovic <zpet731@cse.unsw.edu.au>
- http://xerces.apache.org/xerces2-j/faq-sax.html

Why does the SAX parser lose some character data or why is the data split into several chunks?

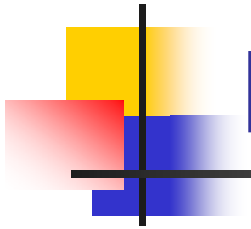
If you read the SAX documentation, you will find that SAX may deliver contiguous text as multiple calls to `characters`, for reasons having to do with parser efficiency and input buffering. It is the programmer's responsibility to deal with that appropriately, e.g. by accumulating text until the next non-characters event.

Xerces will split calls to `characters` at the end of an internal buffer, at a new line and also at a few other boundaries. You can never rely on contiguous text to be passed in a single characters callback.



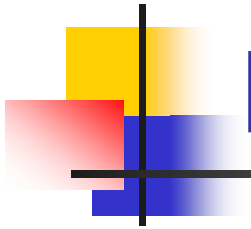
character event in SAX Parser

- Need a string buffer
- character event
  - First time => empty the buffer
  - Append the current text to buffer
- non-character events (element, comment, ...)
  - Serve the buffer as a single text



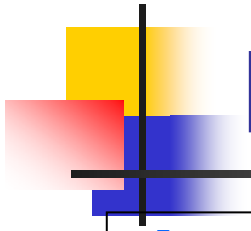
Build a parser

- Build a parser yourself?
  - While (ch)
    - While (isSpace(ch)) ch=charFromStream()
    - Switch (ch)
      - case '<':
        - <?xml ...?>
        - <!DOCTYPE ...>
        - <element>
        - </element>



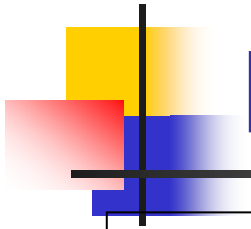
Build a parser

- case '<':
  - ...
  - <?process instruction?>
  - <!--comment-->
  - <![CDATA [...]]>
  - Parse all until '>'
  - Text element (if possible)



Dump the input stream

```
class tutor6_1 {
  static public void main(String args[]) {
    int ch;
    try {
      while ((ch = System.in.read()) != -1) {
        System.out.print((char)ch);
      }
    } catch (Exception e) { e.printStackTrace(); }
    System.out.println();
  }
}
```



Dump the input stream

```
$ javac tutor6_1.java
$ cat h.xml | java tutor6_1
<a>
  <b>Hello</b>
  <c></c>
  <b>World</b>
</a>
$
```

Simple parser: elements only

```
class tutor6_2 {
   static int readOne(InputStream in, int ch) throws Exception {
    try {
      boolean endEl = false;
      while ((ch != -1) && (ch != '<')) ch = in.read();
      ch = in.read(); //the first character after '<'
      if (ch == '/') {endEl = true; ch = in.read(); }
      String name = "";
      while ((ch != -1) && (ch != '>') ) {
        name += (char)ch; ch = in.read(); }
      System.out.println((endEl? "end:":"start:") + name);
    } catch (Exception e) { throw e; }
    return ch;
  }
```

Simple parser: elements only

```
 static public void main(String args[]) {
    int ch = 0;
    try {
      while (ch != -1) {
        ch = readOne(System.in, ch);
      }
    } catch (Exception e) { System.out.println("Error!"); }
  }
}
```

Simple parser: elements only

```
$ javac tutor6_2.java
$ cat h.xml | java tutor6_2
start:a
start:b
end:b
start:c
end:c
start:b
end:b
end:a
$
```

Simple parser: elements & text

```
class tutor6_3 {
  static public void main(String args[]) {
    int ch = 0;
    try {
      while (ch != -1) {
        ch = readOne(System.in, ch);
        if (ch != -1)
          ch = readText(System.in, ch);
      }
    } catch (Exception e) { System.out.println("Error!"); }
  }
```

Simple parser: elements & text

```
static int readText(InputStream in, int ch) throws Exception {
   try {
     if ((ch = in.read()) == -1)
       return ch; //first char after '>'
     String str = "";
     while ((ch != -1) && (ch != '<')) { //text: read until '<'
       str += (char)ch; ch = in.read();
     }
     str = str.trim();
     if (!str.equals(""))
       System.out.println("text:" + str);
   } catch (Exception e) { throw e; }
   return ch;
 }
```

Simple parser: elements & text

```
$ javac tutor6_3.java
$ cat h.xml | java tutor6_3
start:a
start:b
text:Hello
end:b
start:c
end:c
start:b
text:World
end:b
end:a
$
```

Simple parser: elements & text & attributes

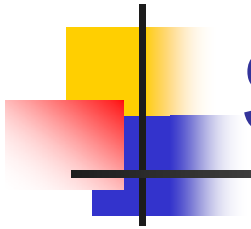
```
class tutor6_4{
   static int readOne(InputStream in, int ch) throws Exception {
    try {
      boolean endEl = false;
      while ((ch != -1) && (ch != '<')) ch = in.read();
      ch = in.read(); //the first character after '<'
      if (ch == '/') {endEl = true; ch = in.read(); }
      String name = "";
      while ((ch != -1) && (ch != '>') && (!isSpace(ch))) {
        name += (char)ch; ch = in.read(); }
      System.out.println((endEl? "end:":"start:") + name);
      if (ch != '>') ch = readAttrs(in, ch);
    } catch (Exception e) { throw e; }
    return ch;  }
```

Simple parser: elements & text & attributes

```
static int readAttrs(InputStream in, int ch) throws Exception {
   try {
     while ((ch != -1) && (ch != '>')) {
       while ((ch != -1) && (isSpace(ch))) ch = in.read();
       String attr = "", value = "";
       while ((ch != -1) && (ch != '=')) {
         attr += (char)ch; ch = in.read(); }
       ch = in.read(); //for '='
       while ((ch != -1) && (!isSpace(ch)) && (ch != '>')) {
         value += (char)ch; ch = in.read(); }
       value = value.substring(1, value.length()-1);
       System.out.println("attr:" + attr + ", value:" + value);
     } } catch (Exception e) { throw e; }
   return ch; }
```

Simple parser: elements & text & attributes

```
$ javac tutor6_4.java
$ cat h1.xml | java tutor6_4
start:a
attr:a1, value:1
start:b
text:Hello
end:b
start:c
end:c
start:b
text:World
end:b
end:a
$
```



Streaming XPath

- Using SAXParser/YourParser with the input stream
- Store necessary subtrees only
- Convert all backward axes into forward axes (if possible)
- Remember which expression is on the rail:
  - //a[./b/c/d]/c
    - May see the child <c> before <b>
- Using KMP search/automaton