XML and Databases
Tutorial session 3: SAX parsing
Binary tree encoding

Kim.Nguyen@nicta.com.au

Week 4

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute “save file” function, . . .)

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute "save file" function, ...)

- Hardware programming

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute “save file” function, . . .)

- Hardware programming

$\Rightarrow$ Interruption handling, hardware timers,. . .

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute "save file" function, ...)

- Hardware programming

$\Rightarrow$ Interruption handling, hardware timers,...

- XML Parsing!

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

⇒ user-based event (click on a widget ⇒ execute “save file” function, …)

- Hardware programming

⇒ Interruption handling, hardware timers,…

- XML Parsing!

⇒ a document can arbitrarily mix comments, text, elements, …

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute "save file" function, …)

- Hardware programming

$\Rightarrow$ Interruption handling, hardware timers,…

- XML Parsing!

$\Rightarrow$ a document can arbitrarily mix comments, text, elements, …

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute “save file” function, ...)

- Hardware programming

$\Rightarrow$ Interruption handling, hardware timers,...

- XML Parsing!

$\Rightarrow$ a document can arbitrarily mix comments, text, elements, ...

SAX: Simple API for XML

Event-based programming?

Want to execute a sequence of elementary actions, but the order is not known in advance:

- GUI programming

$\Rightarrow$ user-based event (click on a widget $\Rightarrow$ execute “save file” function, . . .)

- Hardware programming

$\Rightarrow$ Interruption handling, hardware timers,. . .

- XML Parsing!

$\Rightarrow$ a document can arbitrarily mix comments, text, elements, . . .

SAX: Simple API for XML

SAX Parser

Simple loop:

1. read some character from the input

SAX Parser

Simple loop:

1. read some character from the input
2. try to recognize an XML token

SAX Parser

Simple loop:

1. read some character from the input
2. try to recognize an XML token
3. call the corresponding *callback* (or *handler*)

SAX Parser

Simple loop:

1. read some character from the input
2. try to recognize an XML token
3. call the corresponding *callback* (or *handler*)
4. continue until the end of the input

SAX Parser

Simple loop:

1. read some character from the input
2. try to recognize an XML token
3. call the corresponding *callback* (or *handler*)
4. continue until the end of the input

SAX Parser

Simple loop:

1. read some character from the input
2. try to recognize an XML token
3. call the corresponding ***callback*** (or *handler*)
4. continue until the end of the input

- Only the callbacks need to be defined by the programmer

SAX Parser

Simple loop:

1. read some character from the input
2. try to recognize an XML token
3. call the corresponding *callback* (or *handler*)
4. continue until the end of the input

- Only the callbacks need to be defined by the programmer
- The programmer has to handle the storage/buildling

Sample SAX program in Java

Read an XML Document, count the number of elements.

```
import org.apache.xerces.parsers.SAXParser;
import org.w3c.dom.*;
import java.util.*;
import java.io.*;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;


 ...
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                       String raw, Attributes att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                     String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                      String raw, Attributes  att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                    String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                          String raw, Attributes  att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                        String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                          String raw, Attributes  att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                        String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                           String raw, Attributes att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                         String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                      String raw, Attributes att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                    String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
class SAXExample {

  class MyHandler extends DefaultHandler {
    int count;

    void startElement(String nsuri, String local,
                      String raw, Attributes att)
    {
      count++;
      System.out.println("<"+local+">");
    }
    void endElement(String nsuri, String local,
                    String raw)
    {
      System.out.println("</"+local+">");
    }
```

Sample SAX program in Java

```
  void characters(char[] buffer, int start,int len)
  {
    System.out.println(new String(buffer, start,len);
  }
  void startDocument()
  {
    count = 0;
  }
  void endDocument()
  {
   System.out.println(count + " elements in the document");
  }
}//MyHandler
```

Sample SAX program in Java

```
  void characters(char[] buffer, int start,int len)
  {
    System.out.println(new String(buffer, start,len);
  }
  void startDocument()
  {
    count = 0;
  }
  void endDocument()
  {
   System.out.println(count + " elements in the document");
  }
}//MyHandler
```

Sample SAX program in Java

```
  void characters(char[] buffer, int start, int len)
  {
    System.out.println(new String(buffer, start, len));
  }
  void startDocument()
  {
    count = 0;
  }
  void endDocument()
  {
   System.out.println(count + " elements in the document");
  }
}//MyHandler
```

Sample SAX program in Java

```
public static void main(char [] args){
  SAXParser parser;
  try {
    parser = new SAXParser();
    MyHandler handle= new MyHandler();
    parser.setContentHandler(handle);
    parser.setErrorHandler(handle);
    parser.parse(args[0]);
  }
  catch (Exception e) {
    System.out.println ("Error during parsing"
                              + e.toString());
  };

} //SAXExample
```

Sample SAX program in Java

```
public static void main(char [] args){
  SAXParser parser;
  try {
    parser = new SAXParser();
    MyHandler handle= new MyHandler();
    parser.setContentHandler(handle);
    parser.setErrorHandler(handle);
    parser.parse(args[0]);
  }
  catch (Exception e) {
    System.out.println ("Error during parsing"
                        + e.toString());
  };

} //SAXExample
```

Sample SAX program in Java

```
public static void main(char [] args){
  SAXParser parser;
  try {
    parser = new SAXParser();
    MyHandler handle= new MyHandler();
    parser.setContentHandler(handle);
    parser.setErrorHandler(handle);
    parser.parse(args[0]);
  }
  catch (Exception e) {
    System.out.println ("Error during parsing"
                         + e.toString());
  };

} //SAXExample
```

Sample SAX program in Java

```
public static void main(char [] args){
  SAXParser parser;
  try {
    parser = new SAXParser();
    MyHandler handle= new MyHandler();
    parser.setContentHandler(handle);
    parser.setErrorHandler(handle);
    parser.parse(args[0]);
  }
  catch (Exception e) {
    System.out.println ("Error during parsing"
                        + e.toString());
  };

} //SAXExample
```

Sample SAX program in Java

```
public static void main(char [] args){
  SAXParser parser;
  try {
    parser = new SAXParser();
    MyHandler handle= new MyHandler();
    parser.setContentHandler(handle);
    parser.setErrorHandler(handle);
    parser.parse(args[0]);
  }
  catch (Exception e) {
    System.out.println ("Error during parsing"
                        + e.toString());
  };

} //SAXExample
```

SAX Summary

- extend the class `DefaultHandler`

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`
- bind the your handler to the parser (`parser.setContentHandler()`)

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`
- bind the your handler to the parser (`parser.setContentHandler()`)
- parse the file (`parser.parse()`)

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`
- bind the your handler to the parser (`parser.setContentHandler()`)
- parse the file (`parser.parse()`)

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`
- bind the your handler to the parser (`parser.setContentHandler()`)
- parse the file (`parser.parse()`)

Q: How much memory do you need to parse a file?(without validation)

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`
- bind the your handler to the parser (`parser.setContentHandler()`)
- parse the file (`parser.parse()`)

Q: How much memory do you need to parse a file?(without validation)
Q: How much memory do you need to parse a file?(with validation)

SAX Summary

- extend the class `DefaultHandler`
- create a `SAXParser`
- bind the your handler to the parser (`parser.setContentHandler()`)
- parse the file (`parser.parse()`)

Q: How much memory do you need to parse a file?(without validation)
Q: How much memory do you need to parse a file?(with validation)

Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

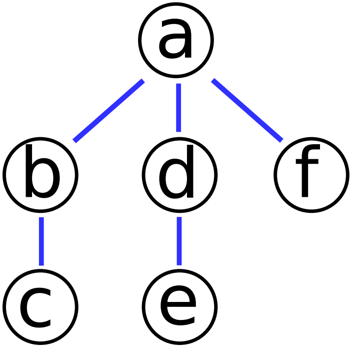
```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

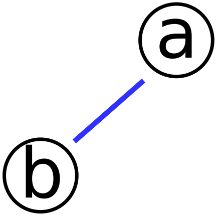


Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

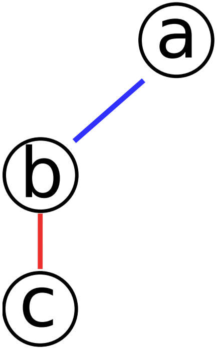


Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree. Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

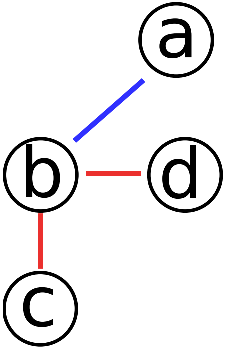


Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
   <b> <c/> </b>
   <d> <e/> </d>
   <f/>
</a>
```

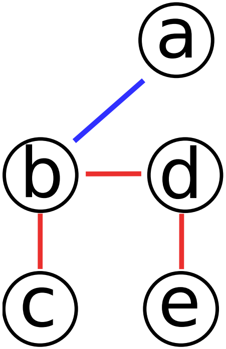


Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
   <b> <c/> </b>
   <d> <e/> </d>
   <f/>
</a>
```

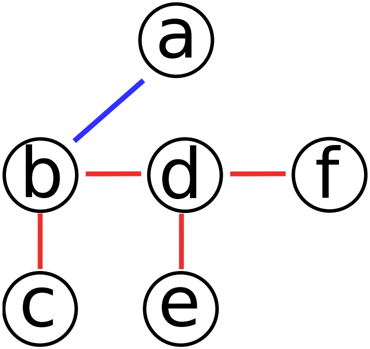


Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
   <b> <c/> </b>
   <d> <e/> </d>
   <f/>
</a>
```

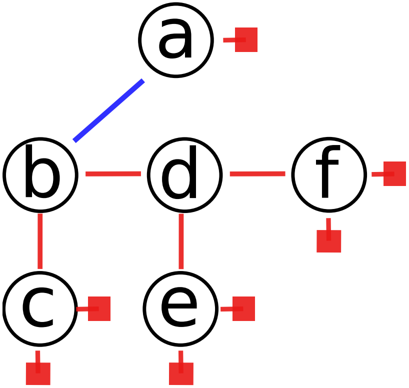


Binary trees: FirstChild/NextSibling encoding

Bijection between an unranked-tree (XML Document) and a binary tree.
Given a node:

- its first child points to its first child in the original document
- its second child points to its next sibling in the original document

```
<a>
   <b> <c/> </b>
   <d> <e/> </d>
   <f/>
</a>
```



Binary trees: FirstChild/NextSibling encoding

- Preserves the document order

```
<a>
   <b> <c/> </b>
   <d> <e/> </d>
   <f/>
</a>
```

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, ...)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, ...)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, ...)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Q: What is the sequence of SAX events for this document?

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, . . .)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Q: What is the sequence of SAX events for this document?
Q: Write an XML document which represents the binary tree (using <_/>) for the empty tree?

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, . . .)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Q: What is the sequence of SAX events for this document?
Q: Write an XML document which represents the binary tree (using <_/>) for the empty tree?
Q: What is the sequence of SAX events for the binary tree?

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, ...)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Q: What is the sequence of SAX events for this document?
Q: Write an XML document which represents the binary tree (using <_/>) for the empty tree?
Q: What is the sequence of SAX events for the binary tree?
Q: Find an algorithm to convert a document into a binary one during SAX parsing.

Binary trees: FirstChild/NextSibling encoding

- Preserves the document order
- Everything else may be different (height, width, . . .)

```
<a>
  <b> <c/> </b>
  <d> <e/> </d>
  <f/>
</a>
```

Q: What is the sequence of SAX events for this document?
Q: Write an XML document which represents the binary tree (using <_/>) for the empty tree?
Q: What is the sequence of SAX events for the binary tree?
Q: Find an algorithm to convert a document into a binary one during SAX parsing.

Binary Tree Wrapper around an Unranked Event Handler

We want to use the previouly defined `MyHandler` to print the binary tree

```
class MyBinaryHandler extends Default Handler{
  MyHandler handler;
```

Binary Tree Wrapper around an Unranked Event Handler

We want to use the previouly defined `MyHandler` to print the binary tree

```
class MyBinaryHandler extends Default Handler{
  MyHandler handler;
  Stack<Pair<String,Integer>> stack;
```

Binary Tree Wrapper around an Unranked Event Handler

We want to use the previouly defined `MyHandler` to print the binary tree

```
class MyBinaryHandler extends Default Handler{
  MyHandler handler;
  Stack<Pair<String,Integer>> stack;
  Integer LEFT = new Integer(0);
  Integer RIGHT = new Integer(1);
```

Binary Tree Wrapper around an Unranked Event Handler

We want to use the previouly defined `MyHandler` to print the binary tree

```
class MyBinaryHandler extends Default Handler{
  MyHandler handler;
  Stack<Pair<String,Integer>> stack;
  Integer LEFT = new Integer(0);
  Integer RIGHT = new Integer(1);

  void startDocument(){
    handler = new MyHandler();
    stack = new Stack<Pair<String,Integer>>();
    stack.push(new Pair<String,Integer>("",LEFT));
  }
```

Binary Tree Wrapper around an Unranked Event Handler

```
  void startElement(String nsuri, String label,
                    String raw,Attributes atts)
{
   stack.push(new Pair<String,Integer>(label,LEFT));
   handler.startElement(nsuri,label,raw,atts);
  }

  void endElement(String nsuri, String label,
                  String raw)
{
   Pair<String,Integer> top = stack.peek();
   if (top.getSecond().equals(LEFT)){
     top.setSecond(RIGHT);
     handler.startElement(null,"_","_",null);
     handler.endElement(null,"_","_",null);
}
```

Binary Tree Wrapper around an Unranked Event Handler

```
else { // Direction is RIGHT
    handler.startElement(null,"_","_",null);
    handler.endElement(null,"_","_",null);
```

Binary Tree Wrapper around an Unranked Event Handler

```
else { // Direction is RIGHT
     handler.startElement(null,"_","_",null);
     handler.endElement(null,"_","_",null);

     while(top.getSecond().equals(RIGHT)){
       handler.endElement(null,top.getFirst(),top.getFi
       stack.pop();
       top = stack.peek();
     };
```

Binary Tree Wrapper around an Unranked Event Handler

```
else { // Direction is RIGHT
     handler.startElement(null,"_","_",null);
     handler.endElement(null,"_","_",null);

     while(top.getSecond().equals(RIGHT)){
       handler.endElement(null,top.getFirst(),top.getFi
       stack.pop();
       top = stack.peek();
     };

     top.setSecond(RIGHT);
```

Binary Tree Wrapper around an Unranked Event Handler

```
else { // Direction is RIGHT
     handler.startElement(null,"_","_",null);
     handler.endElement(null,"_","_",null);

     while(top.getSecond().equals(RIGHT)){
       handler.endElement(null,top.getFirst(),top.getFi
       stack.pop();
       top = stack.peek();
     };

     top.setSecond(RIGHT);
   }//else
 }// endElement()
```

Binary Tree Wrapper around an Unranked Event Handler

```
void endDocument(){
   Pair<String,Integer> top = stack.peek();
   if (top.getSecond().equals(RIGHT)){
     handler.startElement(null,"_","_",null);
     handler.endElement(null,"_","_",null);
     handler.endElement(null,top.getFirst(),top.getFirs
   };
 }

} //end class MyBinHandler
```