hierarchical delta debugging

Outline

Introduction of the paper:

Hierarchical Delta Debugging

Implementation

Performance

Conclusion

Introduction

- Delta Debug by Zeller
  - Worse running time for large test case
  - Ignores input structure and may attempt many spurious input configurations

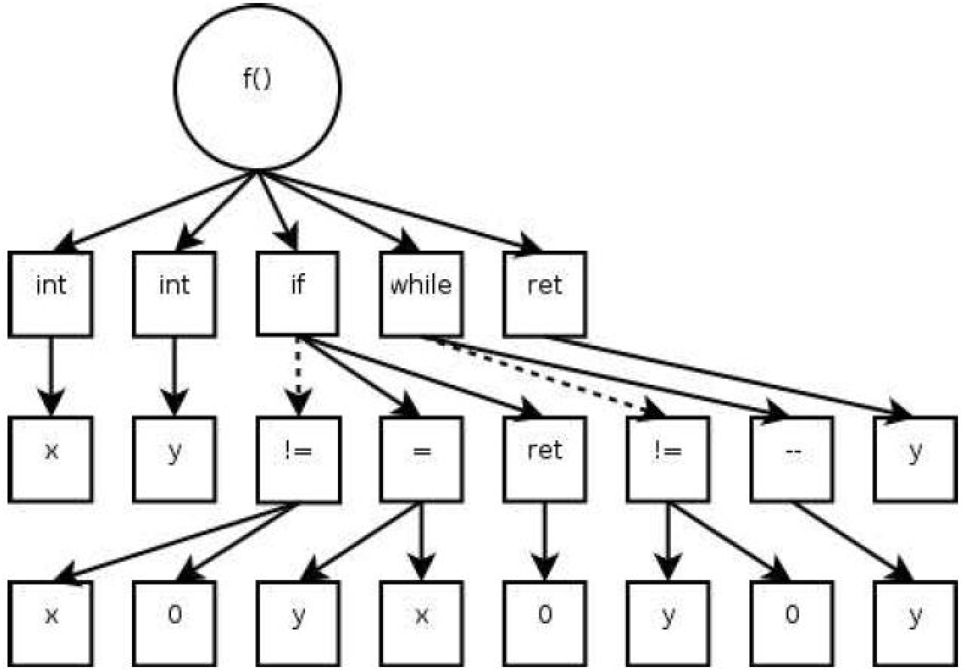
- Hierarchical Delta Debugging
  - Input data is often structured hierarchically
  - Input structure can be exploited to generate fewer input configurations and simpler test cases for more effective automated debugging
  - Good when input data is nested and at least partially balanced

Intro –Hierarchical Delta Debugging

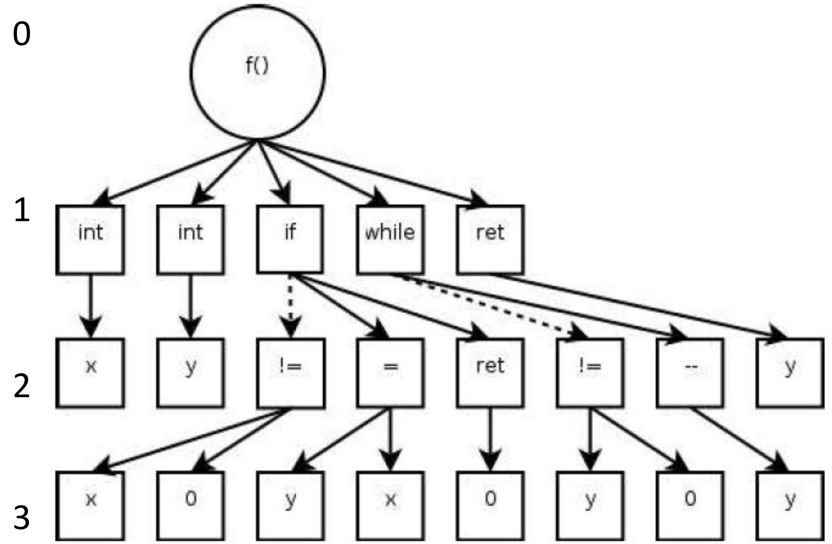
- Use:
  - Programming Languages
  - HTML/XML
  - Video Codes
  - UI Interactions
- Contribute(compare with original Delta Debugging):
  - Fewer test cases
  - Simpler outputs

HDD:Algorithm

- Input:



HDD:Algorithm(cont)



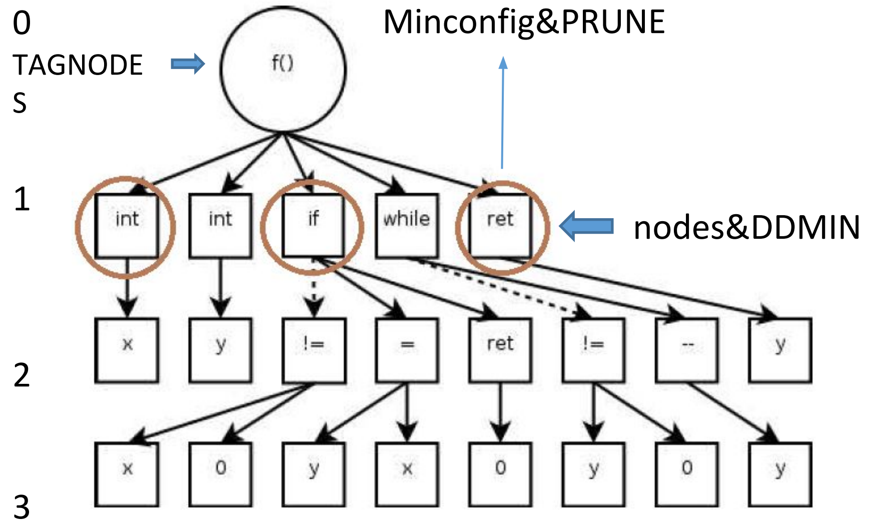
Algorithm 1 The Hierarchical Delta Debugging Algorithm

```
procedure HDD($input\_tree$)
    $level \leftarrow 0$
    $nodes \leftarrow$ TAGNODES($input\_tree, level$)
    while $nodes \neq \emptyset$ do
        $minconfig \leftarrow$ DDMIN($nodes$)
        PRUNE($input\_tree, level, minconfig$)
        $level \leftarrow level + 1$
        $nodes \leftarrow$ TAGNODES($input\_tree, level$)
    end while
end procedure
```

HDD:Algorithm(cont)

- TAGNODES: name each nodes in the level.
- minconfig: nodes that pass the test func.
- PRUNE: prune the nodes in minconfig

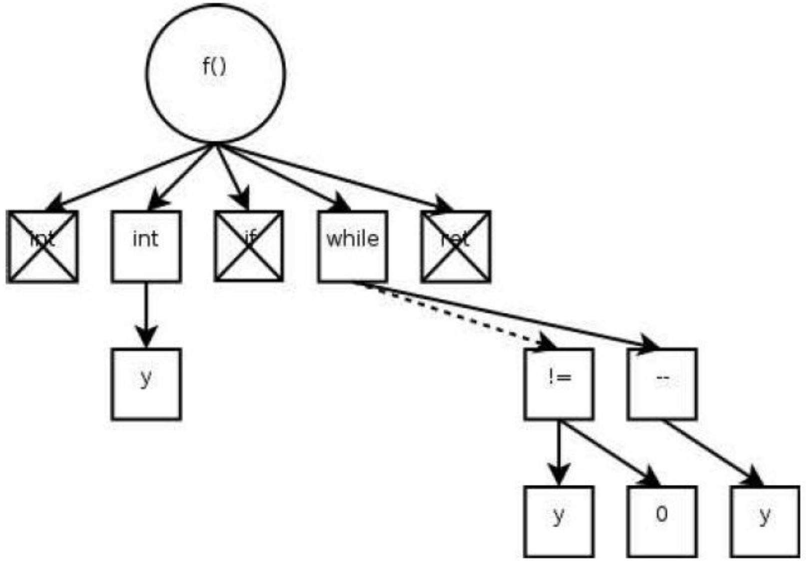
HDD:Algorithm(cont)-level1 for example



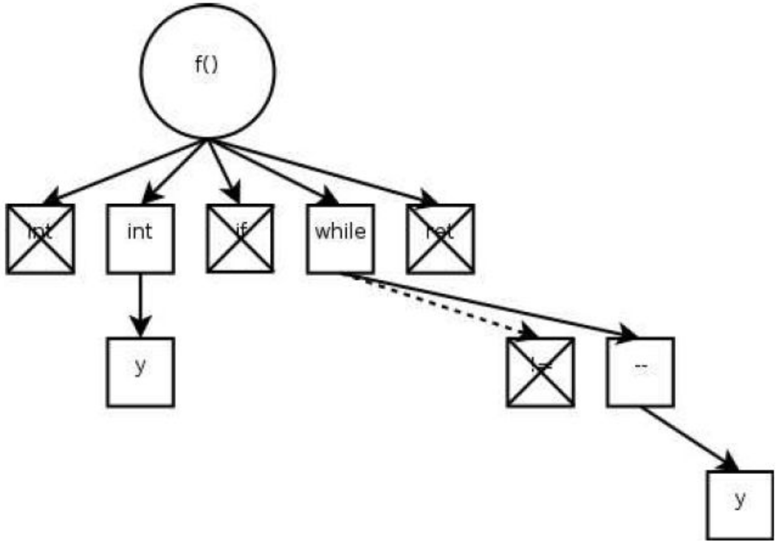
Algorithm 1 The Hierarchical Delta Debugging Algorithm

```
1: procedure HDD($input\_tree$)
2:     $level \leftarrow 0$
3:     $nodes \leftarrow$ TAGNODES($input\_tree, level$)
4:     while $nodes \neq \emptyset$ do
5:         $minconfig \leftarrow$ DDMIN($nodes$)
6:         PRUNE($input\_tree, level, minconfig$)
7:         $level \leftarrow level + 1$
8:         $nodes \leftarrow$ TAGNODES($input\_tree, level$)
9:     end while
10: end procedure
```

HDD:Algorithm(cont)-level1 for example



After the first iteration of HDD

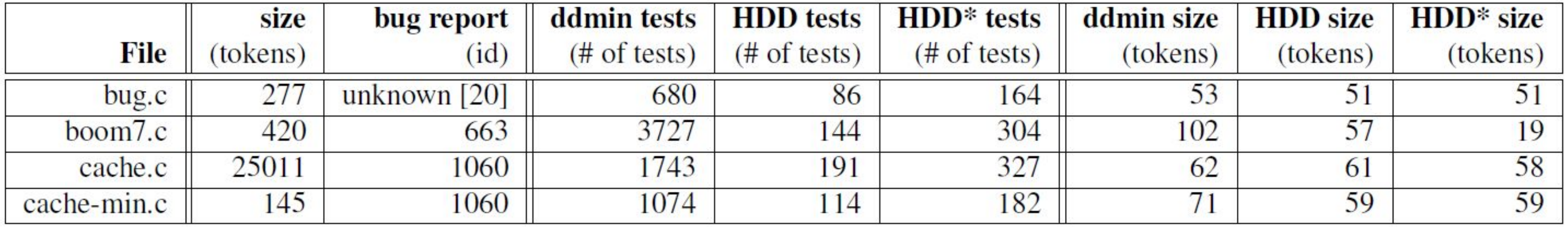


The result of HDD algo

HDD:Algorithm(cont)-Complexity

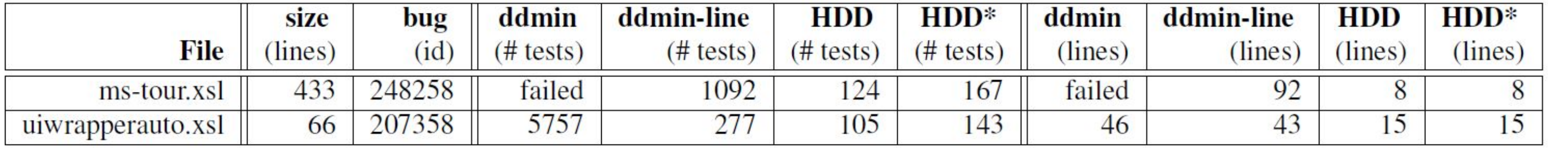
- Balanced Tree:O(logn), same as oringinal DDmin
- Best Case: unbalanced tree.
- Worst case: flat tree(too many nodes at one level)

Evaluation result of this paper



File	size (tokens)	bug report (id)	ddmin tests (# of tests)	HDD tests (# of tests)	HDD* tests (# of tests)	ddmin size (tokens)	HDD size (tokens)	HDD* size (tokens)
bug.c	277	unknown [20]	680	86	164	53	51	51
boom7.c	420	663	3727	144	304	102	57	19
cache.c	25011	1060	1743	191	327	62	61	58
cache-min.c	145	1060	1074	114	182	71	59	59

Table 1: Experimental results. All tests were performed against GCC version 2.9.5.2.



File	size (lines)	bug (id)	ddmin (# tests)	ddmin-line (# tests)	HDD (# tests)	HDD* (# tests)	ddmin (lines)	ddmin-line (lines)	HDD (lines)	HDD* (lines)
ms-tour.xsl	433	248258	failed	1092	124	167	failed	92	8	8
uiwrapperauto.xsl	66	207358	5757	277	105	143	46	43	15	15

Table 2: Experimental results for the XML study.

Limitation&Future work of this paper

- This approach works best if few dependencies between data at different level.
- Tree processing(C/C++) is hard to implement (parsing ,unparsing , pruning nodes). XML is easier.

Conclusion of this paper

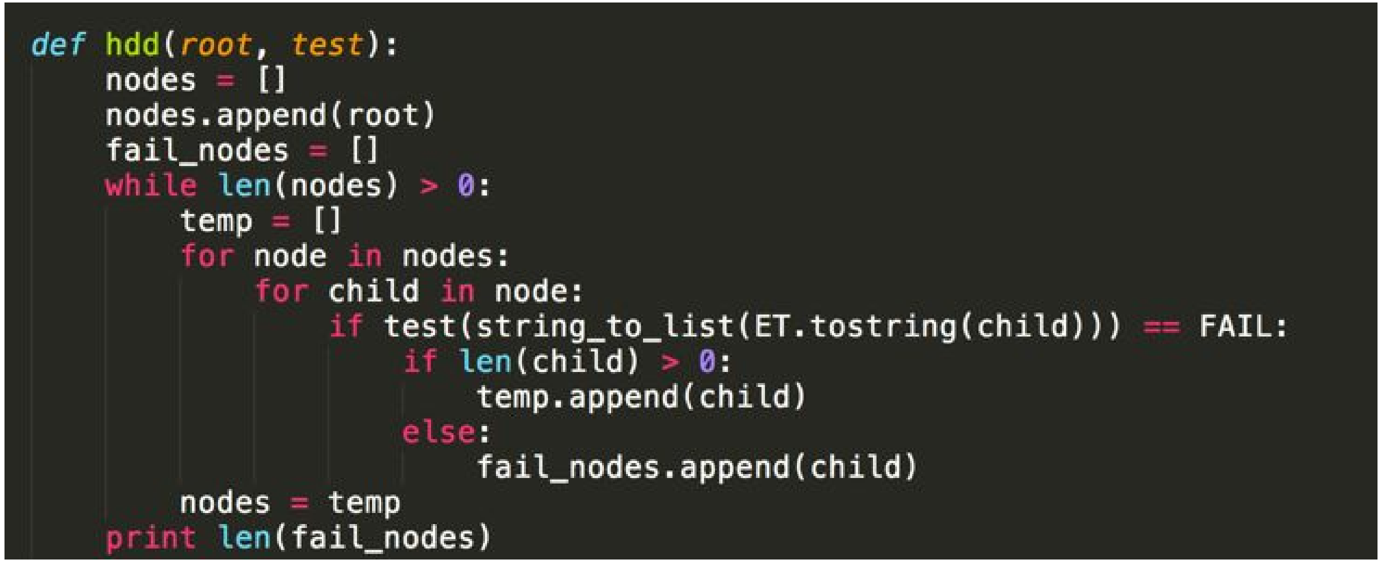
- exploits input structure to minimize failure-inducing inputs.

- Hierarchical Delta Debugging can reduce the number of generated tests and at the same time produce smaller output than the original Delta debugging algorithm

Implementation (1/2)

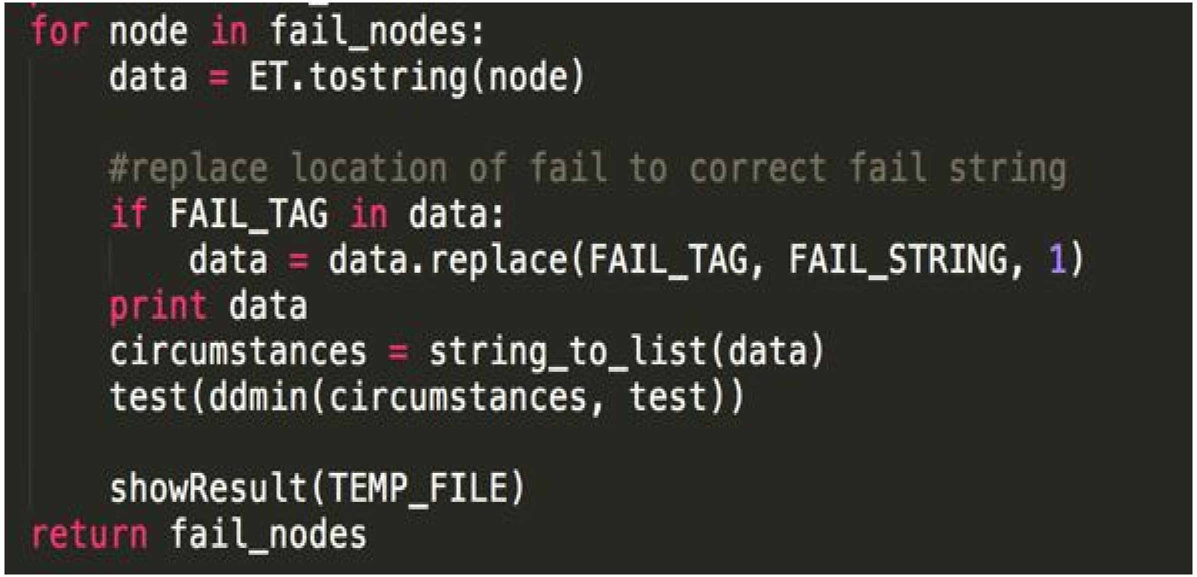
First,we parse XML file to AST by function ET.parse(XML file).

Then push AST into hdd function.



```
def hdd(root, test):
    nodes = []
    nodes.append(root)
    fail_nodes = []
    while len(nodes) > 0:
        temp = []
        for node in nodes:
            for child in node:
                if test(string_to_list(ET.tostring(child))) == FAIL:
                    if len(child) > 0:
                        temp.append(child)
                    else:
                        fail_nodes.append(child)
        nodes = temp
    print len(fail_nodes)
```

Implementation(2/2)



```
for node in fail_nodes:
    data = ET.tostring(node)

    #replace location of fail to correct fail string
    if FAIL_TAG in data:
        data = data.replace(FAIL_TAG, FAIL_STRING, 1)
    print data
    circumstances = string_to_list(data)
    test(ddmin(circumstances, test))

    showResult(TEMP_FILE)
return fail_nodes
```

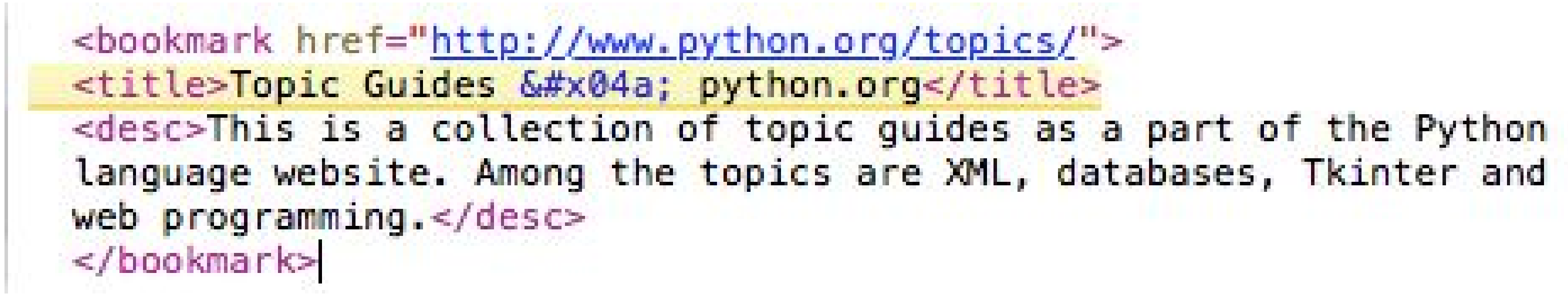
Performance

Hddmin v.s ddmin

- Where fails
- Result

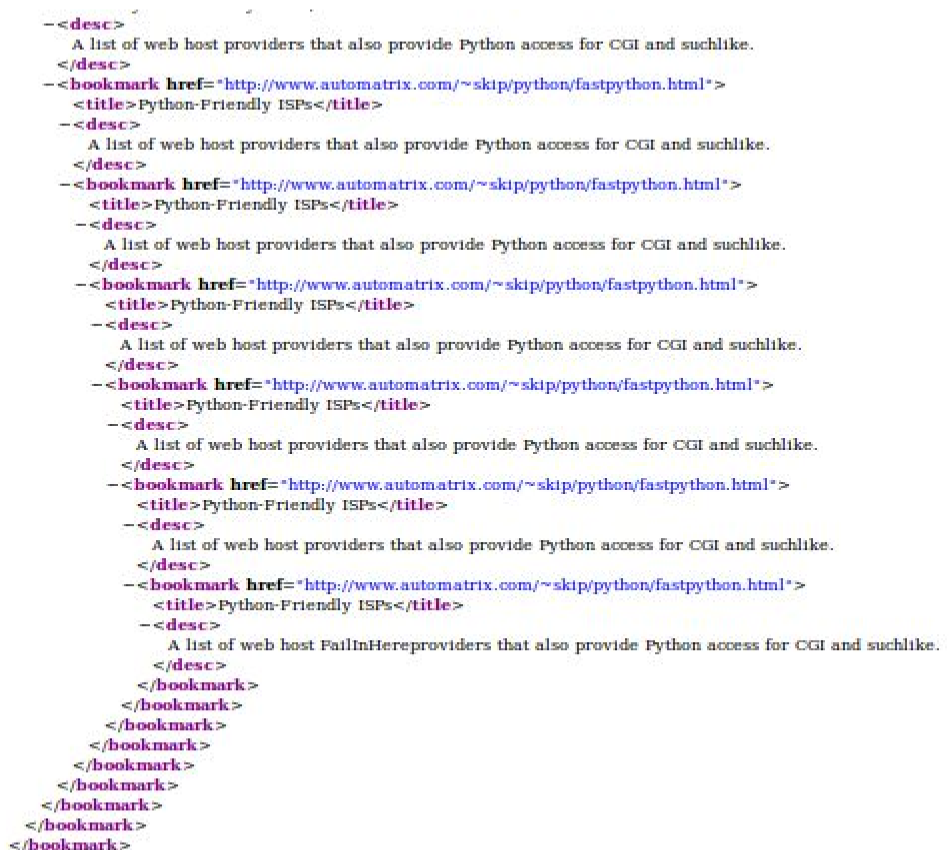
urls.xml

The input file in HW1.3, which fails on



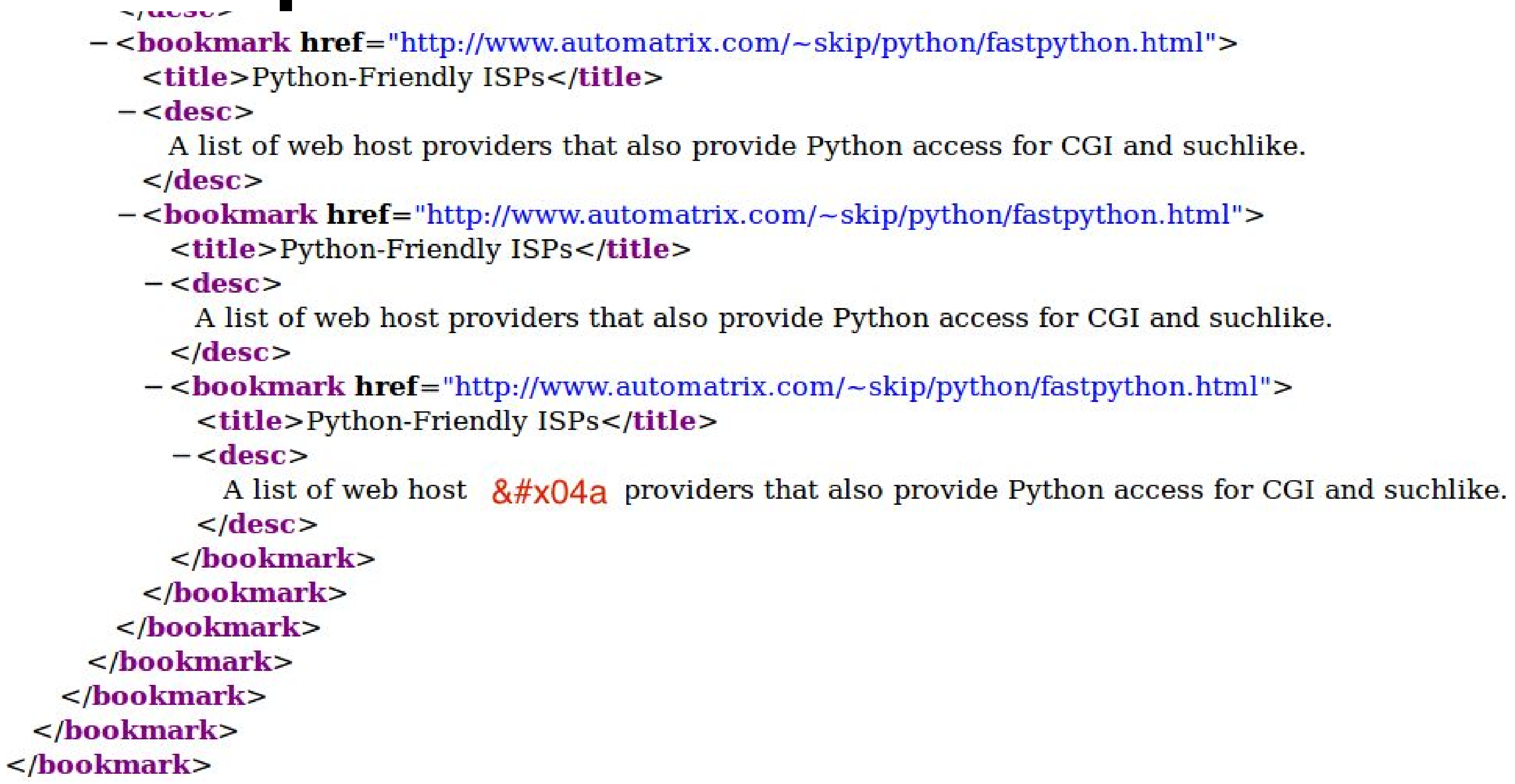
```
<bookmark href="http://www.python.org/topics/">
<title>Topic Guides &#x04a; python.org</title>
<desc>This is a collection of topic guides as a part of the Python
language website. Among the topics are XML, databases, Tkinter and
web programming.</desc>
</bookmark>
```

urls_more_depth.xml



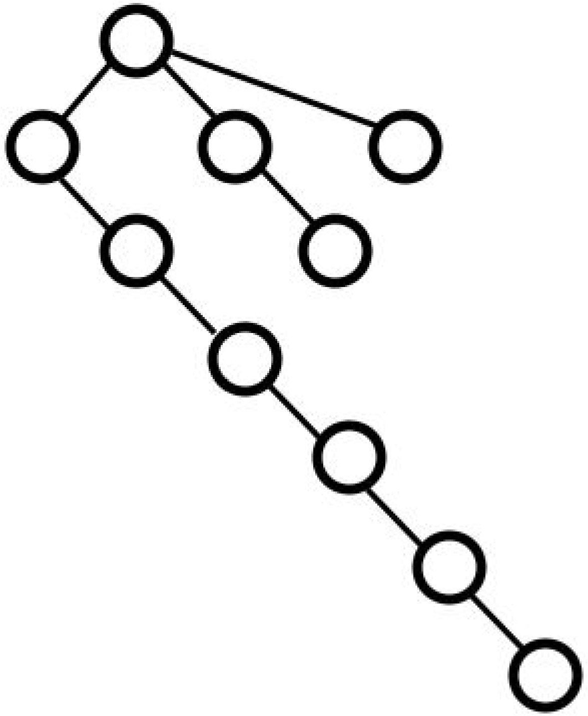
```
     -<desc>
        A list of web host providers that also provide Python access for CGI and suchlike.
      </desc>
     -<bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
        <title>Python-Friendly ISPs</title>
       -<desc>
          A list of web host providers that also provide Python access for CGI and suchlike.
        </desc>
       -<bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
          <title>Python-Friendly ISPs</title>
         -<desc>
            A list of web host providers that also provide Python access for CGI and suchlike.
          </desc>
         -<bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
            <title>Python-Friendly ISPs</title>
           -<desc>
              A list of web host providers that also provide Python access for CGI and suchlike.
            </desc>
           -<bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
              <title>Python-Friendly ISPs</title>
             -<desc>
                A list of web host providers that also provide Python access for CGI and suchlike.
              </desc>
             -<bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
                <title>Python-Friendly ISPs</title>
               -<desc>
                  A list of web host providers that also provide Python access for CGI and suchlike.
                </desc>
               -<bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
                  <title>Python-Friendly ISPs</title>
                 -<desc>
                    A list of web host FailInHereproviders that also provide Python access for CGI and suchlike.
                  </desc>
                </bookmark>
              </bookmark>
            </bookmark>
          </bookmark>
        </bookmark>
      </bookmark>
    </bookmark>
  </bookmark>
</bookmark>
```

fail in depth

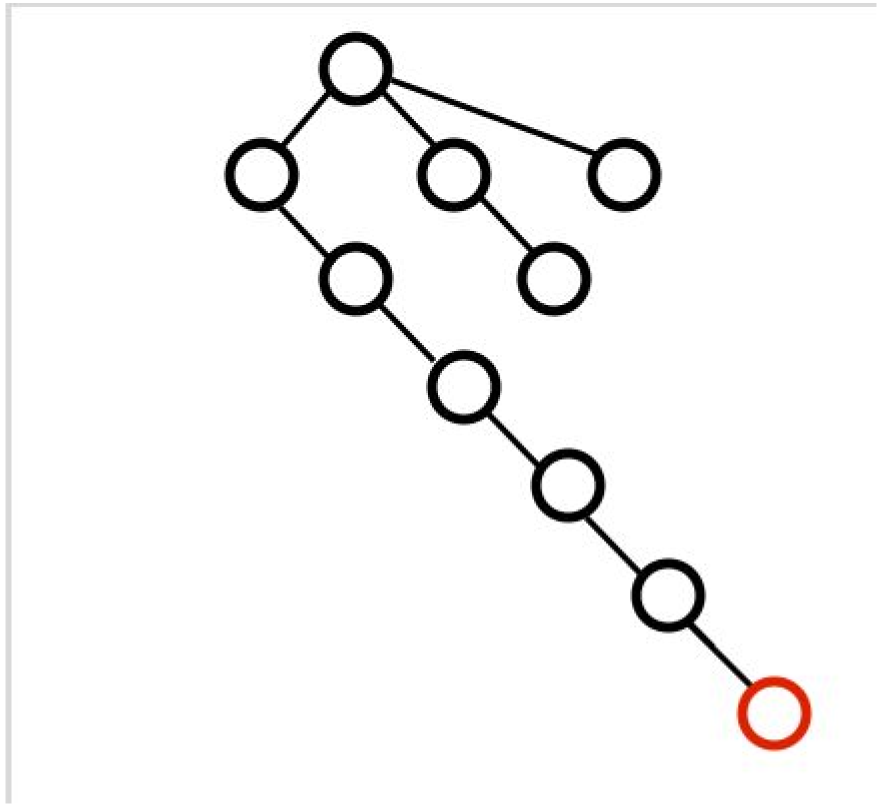


```
– <bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
    <title>Python-Friendly ISPs</title>
  – <desc>
      A list of web host providers that also provide Python access for CGI and suchlike.
    </desc>
  – <bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
      <title>Python-Friendly ISPs</title>
    – <desc>
        A list of web host providers that also provide Python access for CGI and suchlike.
      </desc>
    – <bookmark href="http://www.automatrix.com/~skip/python/fastpython.html">
        <title>Python-Friendly ISPs</title>
      – <desc>
          A list of web host  &#x04a  providers that also provide Python access for CGI and suchlike.
        </desc>
      </bookmark>
    </bookmark>
  </bookmark>
</bookmark>
</bookmark>
</bookmark>
</bookmark>
```

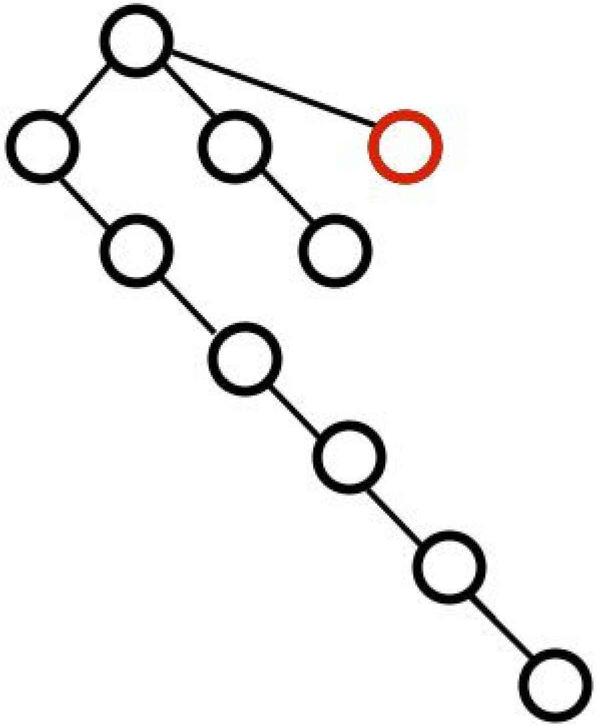
fails



fails in depth



fails not in depth



Running time(ms)

after 1000 tests

	ddmin	HDD
Balanced(url.xml)	20.6	21.5
Unbalanced(fail in depth)	43.4	47.8
Unbalanced1(fail not in depth)	46.8	**23.4**
Unbalanced2(fail not in depth)	43.5	**23.0**

Iterations

	ddmin	HDD
Balanced(url.xml)	22	50
Unbalanced(fail in depth)	79	52
Unbalanced1(fail not in depth)	50	42
Unbalanced2(fail not in depth)	44	48

Conclusion

- We introduced the HDD method
- We implement the real HDD method based on our HW1.3
- In the worst case, HDD has the same performancr as ddmin
- In general case, HDD has great improvement of efficiency