Foundations of computing: Algorithms and Data Structures

Rasmus Pagh

September 1, 2011

1 Programming exercise

The aim of this exercise is to make you comfortable with the basics of reading a file that represents the input to an algorithm. You will need to do this several times later in connection with mandatory hand-ins, so it's better to get started! We recommend doing this in Java. You are welcome to use the `stdlib.jar` library, but this is not required.

Simple XML validation Your program should be able to read a file whose content is a simple XML document. The following is an example input:

```
<xml>
<tag> </tag>
<gart> <nested> </nested> </gart>
</xml>
```

The document consists of opening tags (without /) and closing tags (with /). Each line may have any number of tags, separated by space characters. The task is to figure out if all the tags are matched to a tag with the same name. This can be done by putting all opening tags on a stack. Then, for each closing tag the latest item on the stack is popped, and the name of the tag is checked against that of the closing tag. If there is a mismatch, this should be reported. For example, on this input:

```
<xml>
<mid> </dag>
</xml>
```

your program should output: `</dag> on line 2 does not match opening tag <mid>`. Other sources of error is that there are too many opening tags, or that some closing tag has no matching opening tag at all. You may use the data files provided on the course schedule for testing. Start out with small instances!

1.1 Java reference

What is the output of the following Java program? Compile and run it to check your answer.

```
public class Reference
{
    public static void main (String[] args) {
    C x = new C(42);
    C y = x;
    y.a++;
    System.out.println("x.a="+x.a);
    int z = 42;
    int w = z;
    w++;
    System.out.println("z="+z);
    }
    public static class C {
    public int a;
    C (int x) { a = x; }
    }
}
```

2 Oyster of the week

The "oyster of the week" is a special problems that we pose some weeks that requires special creativity. We expect few of you to be able to solve this, but try it out for fun!

Constant Time Arrays. As we have mentioned, array creation in Java takes linear time: `int [] A= new int[n]` gives you a fresh array of size n with the guarantee that `A[i]==0` for all i.

Assume now that Java also allows you to create uninitialized ("dirty") arrays: `int [] A= dirty int[n]` creates an array of size n, but now you have no guarantee of what the value at `A[i]` is. On the other hand it takes constant time.

Given a programming language with such a "dirty" array, show how to make it behave like a standard array: implement `setValue(i)` and `getValue(i)` in constant time, and ensure that `getValue(i)==0` for all i (until the value changes). It's the best of both worlds!