F# and Go compared to Java

A BREADTH-FIRST SEARCH AND SURVEY COMPARISON OF F# AND GO VS JAVA

MARCUS ÅBERG AND JOHAN LINDEBERG
Abstract

This report follows the Bachelor exam specifications of the 6 credit version of the DD143X course at the Royal Institute of Technology from 2012 and was revised and restructured by Johan Lindeberg in April 2015. In this report we have explored the two up and coming languages Go and F# and made comparisons to the already established language Java. In the first part of the report we go through the syntax of the languages and present the results from a survey on code readability. In the second part of the report we present results from a benchmark of a breadth-first search algorithm. A conclusion is reached that Go has a higher chance of impacting and compete with well-established languages such as Java.
1 Table of Contents
2 Introduction................................................................................................................. 4
3 Purpose...................................................................................................................... 4
4 Part 1: Syntax and Survey.......................................................................................... 5
  4.1 Background........................................................................................................... 5
  4.1.1 The Java programming language...................................................................... 5
  4.1.2 The Go programming language....................................................................... 6
  4.1.3 The F# programming language........................................................................ 8
4.2 Code readability ...................................................................................................... 10
4.3 Method................................................................................................................... 10
4.4 Survey results......................................................................................................... 11
4.5 Discussion of the Survey results ........................................................................... 12
5 Part 2: Breadth-first search benchmark .................................................................. 13
  5.1 Purpose............................................................................................................... 13
  5.2 Method................................................................................................................. 13
  5.2.1 Code implementation..................................................................................... 13
  5.2.2 The benchmarking process............................................................................. 13
5.3 Results................................................................................................................... 14
  5.3.1 Implemented code ....................................................................................... 14
  5.3.2 Benchmark results....................................................................................... 14
5.4 Discussion of the benchmarking results................................................................. 14
  5.4.1 F# ................................................................................................................ 14
  5.4.2 Go ............................................................................................................... 15
6 Conclusion................................................................................................................. 16
7 Bibliography............................................................................................................... 17
  7.1 Source reference ............................................................................................... 17
8 Appendix................................................................................................................... 18
  8.1 Java in Master of Science in Engineering programs............................................ 18
  8.2 Survey............................................................................................................... 19
  8.3 Survey Results..................................................................................................... 30
  8.4 Implemented code in F#................................................................................... 37
  8.5 Implemented code in Go.................................................................................... 39
  8.6 Implemented code in Java................................................................................ 42
2 Introduction

This report will handle a study of the two programming languages F# and Go. The reason these languages were chosen is their potential to become strong challengers of today’s most popular way of programming. These two languages will be compared to the already well established language Java.

F# was created by Microsoft as an improvement of the old programming language ML. By implementing the language into the Visual Studio environment it was given resources such as a big library of predefined functions, a debugger and a code editor. Go is a very new language, version 1 was released March 28, 2012 (1) and has not yet had time to establish itself in the programming world.

The report is split in two separate parts. In Part 1 of the report relevant background information about Java, F# and Go will be given. This includes some general information and syntax explanation. After this results from a survey on code readability given to Computer Science in Engineering students across several different universities in Sweden will be presented and discussed. Part 2 is focused on benchmarking of a breadth-first search algorithm in the three different languages. First the implementation and benchmarking process is explained, after which results are presented. In the end our conclusions from both Part 1 and Part 2 are brought together.

3 Purpose

The main goals with this project are to find out about the differences and possible selling points of the F# and Go programming languages. This includes code readability, how hard it would be to learn these languages with previous knowledge of Java as well as performance. Java has been chosen as a comparison since most of the Computer Science and Engineering programs in Sweden include this language as a part of their course plan (2), and as it’s currently one of the most widely used programming languages.
4 Part 1: Syntax and Survey

4.1 Background
This chapter informs the reader about the three different languages. Only a brief description will be
given of Java, as the reader is presumed to have some previous knowledge of this language. In the Go
and F# topics important syntax differences will be explained, and examples provided to give a better
overview of these languages.

4.1.1 The Java programming language
Java is an object-oriented programming language which was released in 1995 by Sun Microsystems.
It promised a “Write once, run anywhere” (3) policy, with the intent to let the same code run over a
variety of different platforms. This is achieved through the Java Virtual Machine (JVM), a program
which runs between the Java program and the Operating System. JVM interprets compiled Java code
to call the appropriate commands of the current Operating System. Java quickly gained popularity,
especially because of its platform independence. Java is still one of the most popular programming
languages. For over 10 years Java been one of the most popular programming languages, as seen in
the below graph.

Figure 3.1.1: TIOBE Programming Community Index, an indicator of the popularity of programming
languages using popular search engines to calculate the number of skilled engineers, courses and
third party vendors to establish the ratings. (4) (5)
4.1.2 The Go programming language

4.1.2.1 General information
Go is an open source, general-purpose, compiled and concurrent programming language developed mainly by Google employees (6). The development of the language started in September 2007 by Rob Pike, Robert Griesemer and Ken Thompson. Since it is an open source project a lot of people have contributed over the years - with code, documentations and ideas. In November 2009 the language was officially announced, and implementations made available for Linux, Mac OS X and Windows (although the Windows implementation had not yet been optimized). As of March 28, 2012 Go version 1 (or Go 1 for short) was released, the first version of Go available in supported binary distributions across multiple platforms (7). With this release a set of core libraries were defined to provide a stable foundation for creating reliable programs.

Go is a statically-typed language that allows an object-oriented style of programming with types and methods, without actually fulfilling the requirements of an object-oriented language as there are no type hierarchies and no classes. Instead Go uses interfaces, which specifies the behavior of an object. It is possible for a type to implement several interfaces, which it does by just implementing the interfaces required methods. There is no explicit declaration “implements” as in Java.

Go is intended for program structure, not maximum performance. “And finally, the emphasis in Go is on concurrent programming rather than parallel programming” Russ Cox states (9). By this Cox means that Go is focused on structuring your program in a way so that it can cope with having to do many things at once, but still let you write a simple, well-structured program.

4.1.2.2 Types, variable declarations and interfaces
Types in Go are often inferred, not declared. Inside a Go function you can omit the type declaration when initializing a variable, and Go will automatically pull the type off of the initializer. Objects in Go function in a similar manner; they satisfy an interface just by implementing the methods that the interface defines. Explicitly declaring that the object is intended to satisfy an interface is not needed.

A regular variable declaration creates a variable and binds an identifier to the variable, as in Java. If no value is given to the variable during declaration it is initialized to its zero value. See the example below in Figure 4.1.2.2.1 and 4.1.2.2.2

```go
var i int
var i int = 0
```

*Figure 4.1.2.2.1: Both these lines of code initialize i as an int with the value zero.*
After declaring

```go
type T struct {
  i int;
  f float64;
  next *T
}
t := new(T)
```

The following holds:

```go
t.i == 0
t.f == 0.0
t.next == nil
```

*Figure 4.1.2.2.2: As you can see, all the variables in t are initialized to their respective zero-values.*

Go also introduces “Short variable declarations”, which may only appear inside functions. Unlike regular variable declarations, this type of declaration requires a value to be specified. The variable is initialized to the type of the assigned value.

```go
i := 0
f := func() int { return 7 }
aString := hello
var i int = 0, var f int = 7, var aString string = hello
```

*Figure 41.2.2.3: The := operator initializes a variable with the type of the assigned value.*

An interface type specifies a set of methods. Any type T which implements these methods are said to *implement the interface*. As stated earlier, no explicit declaration of this is needed.

```go
type Lock interface {
  Lock()
  Unlock()
}
func (p T) Lock() { }
func (p T) Unlock() { }
```

*Figure 4.1.2.2.4: With these few lines of code the type T will implement the interface Lock, since T have the methods Lock() and Unlock() defined.*

### 4.1.2.3 Functions and methods

A function declaration binds an identifier (the function name) to a function. First comes the `func` keyword, followed by the name of the function and input variables within brackets. After this the return type is stated, followed by the function body within curly brackets. A function may omit the body, this means that the function has been implemented externally, such as an assembly routine.
Figure 4.1.2.3.1: foo() takes two int variables, variable1 and variable2 and returns an int. bar() has no curly brackets, which is a sign that the function has been implemented externally.

A method in Go is a function with a receiver. The receiver must be of type T or *T, where T is a type name.

Figure 4.1.2.3.1: This shows how a method is created. x is the receiver, and has the type *T.

If we now create a struct of type T with a value variable the addOne() function declared above can easily be called using the selector operation, as seen below in figure 3.1.2.3.2.

Figure 4.1.2.3.2: Here we declare the variable z of type T. z.value is initialized to 0, as no value is given. z.addOne() calls the method addOne which increases z.value by one.

4.1.3 The F# programming language

4.1.3.1 General information
F# is a hybridized programming language crossbred from object oriented and functional programming and implemented into the .NET environment. The language originates from the functional programming language ML (10), which influenced several other programming languages, such as Standard ML, Haskell, C++, Caml and several others. As F# is an enhanced dialect of ML, it enables a user to copy ML written code straight into F#, and it will compile and run without any problems. Although that feature may only work one way, as many modern F# programmers uses functions from the .NET library that is not implemented in ML. The implementation of F# into Visual Studio’s .NET framework also allows F# to function freely with the other languages implemented into Visual Studio such as C/C++, C# and Visual Basic (11). The implementation also allows F# developers to access a very large library and set of tools provided when using Visual Studio.

4.1.3.2 White space based syntax
F# has a white space based syntax. It means that its syntax is based on new line, indentations and spaces. Compared to Java, where curly braces are required for the code within the body of each new
function, code within an F# function that is limited to one row is simply placed on the right side of the equal sign. If the code within the body of the function is longer than one row, an indented new line is used. Another case is when the function has several commands of code to execute. Each command in the body has to either be separated with new lines or separated with a comma. See the following examples in Figures 4.1.3.2.1, 4.1.3.2.2 and 4.1.3.2.3.

```fsharp
let executeThisAndThat () =  
  executeThis() 
  executeThat()

Figure 4.1.3.2.1: Allowed
```

```fsharp
let executeThisAndThat() =  
  executeThis() executeThat()

Figure 4.1.3.2.2: Not allowed as there is no comma or newline to separate the commands.
```

```fsharp
let checkIfTeenager x =  
  if x >= 13  
    then if x <= 19  
      then true

Figure 4.1.3.2.3 A function with multiple bodies expressed with if clauses, showing the white space syntax of F#.
```

4.1.3.3 Type inference
A few other things that is interesting to point out is the type inference shown in Figure 4.3.1.1 and how it affects the syntax. There is no need to declare what type a variable has, unless the compiler cannot decide of which type the variable is derived from. This also means that the programmer does not have to declare a return type as the compiler recognizes a return clause and its return type. In Figure 4.1.3.2.3 on the previous page, the type of input parameter “x” is not declared but the compiler assumes it is a variable of type integer as a boolean comparison is made between x and an integer.

4.1.3.4 Loops

```fsharp
for i = 1 to 10 
  do System.Console.Write(i)

for i = 10 downto 1 
  do System.Console.Write(i)

Figure 4.1.3.4.1 Loops in F#
```

The syntax of loops in F# is quite different from that of Java and many other languages. Except for the resemblance of “for each”-loops, F# requires the user to define an interval rather than a Boolean expression to calculate the amount of loops to execute (13). This is shown in the two loop examples in Code example 4.1.3.4.1 above.

4.1.3.5 Lambda expressions
Lambda expressions are powerful operations mainly used on lists and collections. They can be used as substitutes for functions, saving time since lambda expressions dissolves the need for declaring
functions as they can be used anywhere fit. In Figure 4.1.3.5.1 an example is shown of how the lambda expression can be used for a more efficient way of coding.

```plaintext
//Without lambda expression:
let square x = x*x
let squares = List.map square [1..10]

//Using lambda expression:
let squares = List.map (fun x -> x*x) [1..10]
```

*Figure 4.1.3.5.1: Shows a way to use lambda expressions.*

### 4.2 Code readability

The expression “code readability” will be used often when discussing the results of the survey. When referring to the expression in this report, it will simply be a question of “is one able to read and interpret the basics of the code?”. This includes understanding the intention of code segments such as function parameters and return values, variables and their types and classes with their properties. These qualities can be hard to comprehend for someone who has not studied or used the language previously. The reader would then have to guess the intuitive intention of the code based on his or hers former programming experiences in an attempt to establish code readability.

### 4.3 Method

For the second part of this essay we implemented a breadth-first search in Java, F# and Go. From the experience gained with this implementation, we decided upon a number of questions regarding code readability. We decided on the test group of students studying Computer Science, since the students of today are the future of tomorrow. The questions were put into a multiple choice survey and sent out to the Computer Science departments of all the major Universities in Sweden conducting Master of Science in Engineering programs (the schools can be found in Appendix 8.1). For each question a number of answers were given to choose from, including “not sure”, where only one answer was correct. A survey taker has to specify which University he or she is currently studying at, what year they are registered in, if they have had any previous experience with functional programming and if they have had any previous experience with Go or F#. Except from this information, the survey is totally anonymous. The survey was given online through the use of Google Forms, and it’s found in Appendix 8.2 Survey.
4.4 Survey results

Here the most interesting results from a Survey conducted for students studying Computer Science in Engineering across several different Universities in Sweden will be presented. The full list of results can be found in Appendix 8.3 Survey Results.

<table>
<thead>
<tr>
<th>F#</th>
<th>Correct answer</th>
<th>Correct answer %</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1..10]</td>
<td>66</td>
<td>93%</td>
</tr>
<tr>
<td>[x*x]-lista</td>
<td>60</td>
<td>84.5%</td>
</tr>
<tr>
<td>F# map</td>
<td>38</td>
<td>53.5%</td>
</tr>
<tr>
<td>Node, mutable</td>
<td>44</td>
<td>62%</td>
</tr>
<tr>
<td>Change Nodes value to 1</td>
<td>9</td>
<td>12.7%</td>
</tr>
<tr>
<td>Array.map(x, y)</td>
<td>51</td>
<td>71.8%</td>
</tr>
<tr>
<td>Match, head-tail</td>
<td>32</td>
<td>45.1%</td>
</tr>
<tr>
<td>List.map List.sum</td>
<td>39</td>
<td>54.9%</td>
</tr>
</tbody>
</table>

*Figure 4.4.1: results from F# questions found in Appendix 8.2 Survey results*

<table>
<thead>
<tr>
<th>Go</th>
<th>Correct answer</th>
<th>Correct answer %</th>
</tr>
</thead>
<tbody>
<tr>
<td>:= operator</td>
<td>45</td>
<td>63.4%</td>
</tr>
<tr>
<td>Functions, parameters</td>
<td>49</td>
<td>69%</td>
</tr>
<tr>
<td>Method</td>
<td>39</td>
<td>54.9%</td>
</tr>
<tr>
<td>Structs</td>
<td>28</td>
<td>39.4%</td>
</tr>
<tr>
<td>Array, functions</td>
<td>45</td>
<td>63.4%</td>
</tr>
<tr>
<td>Go map function</td>
<td>53</td>
<td>74.6%</td>
</tr>
<tr>
<td>:=, pointer, value</td>
<td>37</td>
<td>52.1%</td>
</tr>
</tbody>
</table>

*Figure 4.4.2: results from Go questions found in Appendix 8.2 Survey results*

Do you have any previous experience with functional programming (like Haskell for example)?

- Yes: 23 (32.4%)
- No: 16 (22.5%)
- Some: 12 (16.9%)
- Very little: 17 (23.9%)

Do you have any previous experience with Go or F#?

- Yes, Go: 2 (2.8%)
- Yes, F#: 0 (0%)
- Yes, both: 0 (0%)
- No: 66 (93%)

*Picture 4.4.3: Results from the question about previous experiences.*
4.4.4: able to read F# and Go code in the future?

Now that a few examples has been presented for F#, do you think that you would be able to read more code written in F#?

<table>
<thead>
<tr>
<th>Option</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>17</td>
<td>23.9%</td>
</tr>
<tr>
<td>Most likely</td>
<td>11</td>
<td>15.5%</td>
</tr>
<tr>
<td>Maybe</td>
<td>25</td>
<td>35.2%</td>
</tr>
<tr>
<td>Probably not</td>
<td>13</td>
<td>18.3%</td>
</tr>
<tr>
<td>No</td>
<td>4</td>
<td>5.6%</td>
</tr>
</tbody>
</table>

Now that a few examples has been presented for Go, do you think that you would be able to read more code written in Go?

<table>
<thead>
<tr>
<th>Option</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>10</td>
<td>14.1%</td>
</tr>
<tr>
<td>Most likely</td>
<td>23</td>
<td>32.4%</td>
</tr>
<tr>
<td>Maybe</td>
<td>21</td>
<td>29.6%</td>
</tr>
<tr>
<td>Probably not</td>
<td>10</td>
<td>14.1%</td>
</tr>
<tr>
<td>No</td>
<td>3</td>
<td>4.2%</td>
</tr>
</tbody>
</table>

4.5 Discussion of the Survey results

The code examples for the questions in the survey were taken directly from our own breadth-first algorithm implementations as well as tutorials provided by Microsoft and Google. The questions may be of varying complexity, but we put effort into keeping them at an equal difficulty, while still incorporating some of the syntactic sugar and subtleties the languages offer. The results should therefore give a general overview of how difficult it is for Computer Science students to read the code. In the end of the survey students were also asked how difficult they think it would be to learn F# and Go.

Figure 4.4.5: how hard would it be to learn F# and Go?
the language as well as if they think they would be able to read more code. We use this as a measurement to establish how complex they deem the languages to be.

As we can see in Figure 4.4.5, almost 46.5% of the participants thinks that learning Go would be Easy or Probably Easy, and only 9.8% thought it would be Hard or Probably Hard. This is better than F# where only 33.8% thought it would be easy or probably easy to learn the language, while 18.3% thought it would be hard or probably hard.

The only part in the Go-section of the survey where more participants stated the wrong answer rather than the right was regarding structs and inferred type declarations. This is a very distinct part of the Go language, and was an easy task for us to learn.

The question referred to as “Change Nodes value to 1” in Figure 4.4.1 only had 12.7% correct answers. However if you look at Appendix 8.3 Survey Results you see that 35.2% thought that rootNode.value = 1, which is a very small mistake from the correct answer rootNode.internalValue = 1. If we allow both answers to be regarded as correct, this adds up to 47.9% correct answers.

5 Part 2: Breadth-first search benchmark

5.1 Purpose
The main purpose of the second part of this report is to give an indication on how the languages perform for a common programming task; the breadth-first search (further referenced as BFS).

5.2 Method

5.2.1 Code implementation
The code writing process followed the same pattern in all of the languages. First off, a small algorithm was implemented to make some values for the nodes in our graph. This algorithm will be referred to as “Make list” in the future. After this a way to represent the nodes was required. In Java and F# this took the form of a class, while in Go, which does not support classes, the nodes were represented as structs. To create the graph on which the BFS later was performed on, a “Make graph” function was implemented. “Make graph” takes values computed in the “Make list” function to create a graph (a list of node elements) for the BFS to run on. The BFS itself was implemented according to a common BFS pseudo code (14). This implementation of BFS uses a queue structure for unvisited nodes, and as Go had no package routine for a Queue structure, an open source Go implementation of Queue was used (15).

5.2.2 The benchmarking process
When the group had implemented the algorithms separately into the different languages a stopwatch benchmarking was executed. The stopwatch benchmarking process was conducted by using a number of implementations of the stopwatch class found in the languages libraries. Each stopwatch was programmed to time a specified algorithm in a series of executions. This process was performed by creating a loop for each algorithm, calling it a specified number of times. The result was then divided by the value used in the loops in order to establish an average execution time for
each call to the algorithm. In this study the value 10,000 were used to define the number of times to loop.

The study was conducted on two separate computers whose technical details are found in Table 5.2.2.1 below, whereas the results of the benchmarking process are found in 5.3.2 Benchmark results.

<table>
<thead>
<tr>
<th>Computer</th>
<th>CPU</th>
<th>CPU Bus speed</th>
<th>DRAM Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computer 1</strong></td>
<td>Intel i7 950 @ 3,8GHz</td>
<td>200 MHz</td>
<td>6Gb DDR3 RAM @ 800Mhz</td>
</tr>
<tr>
<td><strong>Computer 2</strong></td>
<td>AMD Phenom II Quad core 940 @ 3,0GHz</td>
<td>200 Mhz</td>
<td>6Gb DDR2 RAM @ 333Mhz</td>
</tr>
</tbody>
</table>

*Table 5.2.2.1: Technical details of the computers the benchmark was performed on.*

### 5.3 Results

#### 5.3.1 Implemented code

All code implemented in F# is found in the Appendix at section 8.4 Implemented code in F#, while all code implemented in Go is found in the Appendix 2 at section 8.5 Implemented code in Go and all code implemented in Java is found in the Appendix 2 at section 8.6 Implemented code in Java.

#### 5.3.2 Benchmark results

<table>
<thead>
<tr>
<th>Computer 1</th>
<th>Make list</th>
<th>Create graph</th>
<th>Run BFS</th>
<th>Everything</th>
</tr>
</thead>
<tbody>
<tr>
<td>F#</td>
<td>28,87</td>
<td>41,86</td>
<td>1,50</td>
<td>72,23</td>
</tr>
<tr>
<td>Go</td>
<td>0,05</td>
<td>0,179</td>
<td>0,753</td>
<td>0,982</td>
</tr>
<tr>
<td>Java</td>
<td>0,006</td>
<td>0,144</td>
<td>0,175</td>
<td>0,325</td>
</tr>
</tbody>
</table>

*Table 5.3.2.1: Computer 1 test results, all times in milliseconds*

<table>
<thead>
<tr>
<th>Computer 2</th>
<th>Make list</th>
<th>Create graph</th>
<th>Run BFS</th>
<th>Everything</th>
</tr>
</thead>
<tbody>
<tr>
<td>F#</td>
<td>54,57918572</td>
<td>76,56771481</td>
<td>4,17422756</td>
<td>135,32212809</td>
</tr>
<tr>
<td>Go</td>
<td>0,038</td>
<td>0,266</td>
<td>0,852</td>
<td>1,156</td>
</tr>
<tr>
<td>Java</td>
<td>0,021</td>
<td>0,551</td>
<td>0,806</td>
<td>1,378</td>
</tr>
</tbody>
</table>

*Table 5.3.2.2: Computer 2 test results, all times in milliseconds*

### 5.4 Discussion of the benchmarking results

#### 5.4.1 F#

Table 5.3.2.1 and 5.3.2.2 shows a clear time discrepancy between F# and the two other languages, especially for the “Make list” and “Create graph” results. There is a high chance this has to do with our inexperience in programming in the language, as the language is written in a very different way from Go and Java which in turn are somewhat more similar to each other. What we regard as easy tasks such as creating a list of values and a graph, was quite tricky for us to implement in F#. With this in mind, the benchmark results still clearly shows that Go and Java perform a lot better. Our main criteria here is the time it takes to run the actual breadth-first search. Omitting the times it takes for “Make list” and “Create graph” F# performs about half as quick as Go, and almost at 1/10th the speed of Java on Computer 1, and around 1/5th the speed of both Go and Java on Computer 2.

14
5.4.2 Go
The breadth-first search executed in Go runs at about 1/4th the speed compared to Java on Computer 1, and about the same speed on Computer 2. This is probably due to the fact that Go is a quite new language that has not yet been optimized completely for the different architectures.
6 Conclusion
As both F# and Go are backed up by two of the leading software companies in today’s industry, they certainly have a possibility to become challengers of today’s big programming languages. Most participants gave the right answer for the code examples given in the survey seen in chapter 3.4. Both languages seem quite understandable for students currently studying Computer Science and Engineering. However, for a common task such as running a breadth-first search our results show that F# is not performing as well as Go and Java. As discussed in section 4.4.1, the benchmarks of “Make list” and “Create graph” is probably due to our inexperience using the language. However, this also shows that F# was quite hard for us to use with mostly a background of Java programming. As discussed in section 3.5, a similar result can be seen from the survey where only 33.8% thought it would be easy to learn the language, compared to 46.5% in the case of Go. The benchmarking clearly showed that Go was performing better than F# when running the breadth-first search. This leads us to the conclusion that Go is both out-performing F# as well as being more easily read and understood by Computer Science students with a background of writing code in Java, and therefore shows a stronger possibility of becoming a real challenger to today’s more common programming languages.
7 Bibliography

7.1 Source reference

   http://blog.golang.org/2012/03/go-version-1-is-released.html (Accessed 2012-04-05)

2. Appendix 8.1 Java in Master of Science and Engineering Programs


5. Tiobe Software. TIOBE Programming Community Index Definition. Date not available.

   http://golang.org/ref/spec (Accessed 2012-04-01)

   http://blog.golang.org/2012/03/go-version-1-is-released.html (Accessed 2012-04-05)


    http://smlnj.sourceforge.net/ (Accessed 2012-03-16)

11. Microsoft Research. F# at Microsoft Research. Date not available.

12. Microsoft Developer Network. Type Inference (F#). Date not available.


    http://rosettacode.org/wiki/Queue/Definition#Go (Accessed 2012-04-12)
## Appendix

### 8.1 Java in Master of Science in Engineering programs

<table>
<thead>
<tr>
<th>University</th>
<th>Studies Java</th>
<th>Program if other than Master of Science in Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blekinge Tekniska Högskola</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Chalmers Tekniska Högskola</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Karlstad Universitet</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Kungliga Tekniska Högskolan</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Linköpings Tekniska Universitet</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Luleå Tekniska Universitetet</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Lunds Tekniska Högskola</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Mittuniversitetet</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Umeå Högskola</td>
<td>Yes</td>
<td>Civilingenjörprogrammet I Tekniska Datavetenskap</td>
</tr>
<tr>
<td>Uppsala Universitet</td>
<td>Yes</td>
<td>Civilingenjörprogrammet I Informationsteknologi</td>
</tr>
</tbody>
</table>
8.2 Survey

Survey F# and Go

At some point you will be given the answer to a previous question, please do not go back in the form to change an answer.

At which university/college are you studying at?
- Blekinge Tekniska Högskola
- Chalmers Tekniska Högskola
- Karlstad Universitet
- Kungliga Tekniska Högskolan
- Linköpings Tekniska Universitet
- Luleå Tekniska Universitet
- Lunds Tekniska Universitet
- Mittuniversitetet
- Örebro Högskola
- Uppsala Universitet

In which year are you currently registered in?
- Year 1
- Year 2
- Year 3
- Year 4
- Year 5

Do you have any previous experience with functional programming (like Haskell for example)?
- Yes
- No
- Some
- Very little

Do you have any previous experience with Go or F#?
- Yes, Go
- Yes, F#
- Yes, both
- No
Survey F# and Go

F# code readability
Here follows a couple of F# example code. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

Consider the code below. What do you think it does?
[1..10]
- Creates a list of some sort containing the elements [1, empty, empty,..., 10]
- Creates a list of some sort containing the elements [1, 2, 3,..., 10]
- Creates a list of some sort containing the elements [1,10]
- Not sure

Consider the code below. What do you think it does?
[for x in 0..10 -> x*x]
- Creates a list in which each index contains a function
- Creates a list in which each index contains the return value of the function x*x
- Creates a list that contains the values [0*0, empty, empty,..., 10*10]
- Creates a list that contains the values [0*0, 10*10]
- Not sure

Consider the code below. What do you think it does?
Array.map(x y)
- Creates a map, with x and y coordinates, of the size given by x and y
- Creates an array containing the elements of x plus the elements of y
- Creates an array containing the elements according to the pattern [x[0], y[0], x[1], y[1],...x[n], y[n]]
- Takes a function x and applies it on every element in y
- Not sure
Survey F# and Go

**F# code readability**

Here follows an example of an F# class. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

```fsharp
type Node(value, visited) =
    let mutable internalValue = value
    new (value)
    member this.value =
        with get() = internalValue
        and set(value) = internalValue <- value
```

Consider the code in the header of this page. Which row is the constructor of the class?

- [ ] `type Node(value, visited) =`  
- [ ] `let mutable internalValue = value`  
- [ ] `new (value)`  
- [ ] `with get() = internalValue`  
- [ ] `and set(value) = internalValue <- value`  
- [ ] Not sure

Consider the code in the header of this page. One can create a new object of the Node class by writing for example "let rootNode = new Node(0, false)", but what would someone have to write in order to change the Nodes value to 1?

- [ ] `let rootNode = Node(1, false)`  
- [ ] `rootNode.value = 1`  
- [ ] `rootNode.intValue = 1`  
- [ ] `rootNode.value <- 1`  
- [ ] `rootNode.intValue <- 1`  
- [ ] Not sure
F# code readability
Here follows an example of F# code. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

```fsharp
let testArray = [[1;2;3;4;5]]
let square x = x*x
Array.map(square testArray)
```

Consider the code in the header of the page. What do you think the code "Array.map(x y)" does?
- Creates a map, with x and y coordinates, of the size given by x and y
- Creates an array containing the elements of x plus the elements of y
- Creates an array containing the elements according to the pattern [x[0],y[0],x[1],y[1],...x[n],y[n]]
- Takes a function x and applies it on every element in y
- Not sure
F# code readability
Here follows an example of F# code. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

```fsharp
let rec SomeFunction x =
    match x with
    | [] -> 0
    | y::ys -> y + SomeFunction ys
```

Consider the code in the header of the page. What do you think the function does if \( x \) contains a value?

- It takes a word and returns the first letter
- It takes an integer and creates a list of size \( x \)
- It removes all zeroes from the list
- Takes a list \( x \) and returns the sum of the elements in \( x \)
- Not sure
F# code readability
Here follows an example of F# code. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

```fsharp
let test n =
    [1..n]
    |> List.map Square
    |> List.sum
```

Consider the code in the header of the page. What do you think the function does?

- It sums the result and then squares the sum
- It squares each element and then sums the squares
- It returns an integer containing the value of the sum function and a list containing the squares of each element
- It says that only the functions “square” and “sum” are applicable on the list
- Not sure
Go code readability
Here follows a couple of examples of Go code. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

Consider the code below. Which answer is correct?
```go
def makeList(n):
    list = []
    for i in range(n):
        list.append(i)
    return list
```
- list is initialized to the same type as the returned value from makeList(100), and holds the returned value.
- list has been initialized earlier, and holds the returned value of makeList(100)
- Not sure

Consider the code below. Which answer is correct?
```go
def someFunction(graph []Node) bool
    return true
```
- someFunction takes some variable of type []Node and returns nothing
- someFunction takes a specific variable named graph and returns a bool
- someFunction takes some variable of type []Node and returns a bool
- someFunction takes a variable of type bool and returns a variable graph of type []Node
- Not sure

Consider the code below. On what struct / type is the method performed, what variables does it take and what is the return type?
```go
def Push(node, queue):
    queue.append(node)
```
- Performed on struct / type: bool, Input variable: x, Return type: Queue
- Performed on struct / type: *Queue, Input variable: bool, Return type: Node
- Performed on struct / type: Node, Input variable: x, Return type: bool
- Performed on struct / type: Node, Input variable: q, Return type: bool
- Performed on struct / type: bool, Input variable: q, Return type: Node

Consider the code below. What values does r hold after execution?
```go
type Vertex struct {X, Y int}
r := Vertex{X: 1}
```
- (1,0)
- [0,1]
- Uncertain
Go code readability
Here follows an example of Go code. Just check the option that you think is the right one, and if you do not know just check the bottom option of the question.

fmt.Println() is a standard package function to print text.

```go
x := []int{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
for i := 0; i < 10; i++ {
    x[i] = x[i]*x[i]
}
fmt.Println(x[0:5])
```

Consider the code in the header of the page. What do you think is displayed after execution?

- {0,1,4,9,16,25,36,49,64,81}
- {0,1,4,9,16,25}
- {5,6,7,8,9}
- {0,5}
- Not sure
Go code readability
Here follows an example of Go code. What do you think is printed out on screen after execution?

```go
m := map[string]Vertex{
    "Bell Labs": Vertex{
        40.68433, 74.39967,
    }
} fmt.Println(m["Bell Labs"])```

Go map. What is printed out after execution?
- {40.68433 74.39967}
- Bell labs {40.68433 74.39967}
- Bell labs
- {40.60433 74.39967} Bell labs
- Not sure
Survey F# and Go

Go code readability
Here follows an example of Go code. What do you think is printed out on screen after execution?

```go
p := Vertex{1, 2}
q := &p
q.X = 1e9
fmt.Println(p)
```

What is printed out after execution?

- [ ] {0,2}
- [ ] {1000000000 0}
- [ ] {1000000000 2}
- [ ] {1,2}
- [ ] Not sure

« Back  Continue »
Survey F# and Go

What did you think about F# and Go?
This is the last page and we would very much like to know what you thought about the two
programming languages based on the example code that you have seen

Now that a few code examples has presented for F#, do you think that you would be able to
read more code written in F#?
- Yes
- Most likely
- Maybe
- Probably not
- No

Now that a few code examples has presented for Go, do you think that you would be able to
read more code written in Go?
- Yes
- Most likely
- Maybe
- Probably not
- No

Now that a few code examples has presented for F#, how hard do you think it would be for
you to learn F#?
- Hard
- Probably hard
- Not hard but not easy either
- Probably easy
- Easy

Now that a few code examples has presented for Go, how hard do you think it would be for
you to learn Go?
- Hard
- Probably hard
- Not hard but not easy either
- Probably easy
- Easy

[Back] [Submit]
8.3 Survey Results

At which university/college are you studying at?

- Ericsson 7%
- KTH 21%
- Linköpings Tekniska Universitet 14%
- Lunds Tekniska Universitet 14%
- Mittuniversitetet 4%
- Umeå högskola 35%
- Uppsala Universitet 0%

In which year are you currently registered in?

- Year 1 9%
- Year 2 8%
- Year 3 14%
- Year 4 14%
- Year 5 20%

Do you have any previous experience with functional programming (like Haskell for example)?

- Yes 23%
- No 16%
- Some 12%
- Very little 17%

Do you have any previous experience with Go or F#?

- Yes, Go 2%
- Yes, F# 0%
- Yes, both 0%
- No 68%

F# code readability

Consider the code below. What do you think it does?
Consider the code below. What do you think it does?

- Creates a list in which each index contains a function
  - 2 votes (2.6%)
- Creates a list in which each index contains the return value of the function x*x
  - 60 votes (84.5%)
- Creates a list that contains the values [0, empty, empty, ..., 10*10]
  - 1 vote (1.4%)
- Creates a list that contains the values [0,0, 10*10]
  - 3 votes (4.2%)
- Not sure
  - 2 votes (2.8%)

Consider the code below. What do you think it does?

- Takes a list [30]
  - 1 vote (1.4%)
- Creates a list [30]
  - 1 vote (1.4%)
- Creates a list [10]
  - 9 votes (12.7%)
- Not sure
  - 13 votes (18.3%)

F# code readability

Consider the code in the header of this page. Which row is the constructor of the class?
Consider the code in the header of this page. One can create a new object of the Node class by writing for example "let rootNode = new Node(0, false)", but what would someone have to write in order to change the Nodes value to 1?

<table>
<thead>
<tr>
<th>Code</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>let rootNode = Node(1, false)</td>
<td>8</td>
<td>11.3%</td>
</tr>
<tr>
<td>rootNode.value = 1</td>
<td>25</td>
<td>35.2%</td>
</tr>
<tr>
<td>rootNode.internalValue = 1</td>
<td>9</td>
<td>12.7%</td>
</tr>
<tr>
<td>rootNode.value = 1 &lt; 1</td>
<td>9</td>
<td>12.7%</td>
</tr>
<tr>
<td>rootNode.internalValue = 1 &lt; 1</td>
<td>7</td>
<td>9.9%</td>
</tr>
<tr>
<td>Not sure</td>
<td>10</td>
<td>14.1%</td>
</tr>
</tbody>
</table>

**F# code readability**

Consider the code in the header of the page. What do you think the code "Array.map(x y)" does?

<table>
<thead>
<tr>
<th>Code</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates a map, with x and y coordinates , of the size given by x and y</td>
<td>2</td>
<td>2.8%</td>
</tr>
<tr>
<td>Creates an array containing the elements of x plus the elements of y</td>
<td>1</td>
<td>1.4%</td>
</tr>
<tr>
<td>Creates an array containing the elements according to the pattern [[x(1),y(1)],x(1)...[x(n),y(n]</td>
<td>5</td>
<td>7%</td>
</tr>
<tr>
<td>Takes a function containing the elements and applies it on every element in y</td>
<td>51</td>
<td>71.0%</td>
</tr>
<tr>
<td>Not sure</td>
<td>8</td>
<td>11.3%</td>
</tr>
</tbody>
</table>

**F# code readability**

Consider the code in the header of the page. What do you think the function does if x contains a value?

<table>
<thead>
<tr>
<th>Code</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not sure</td>
<td>2</td>
<td>2.8%</td>
</tr>
<tr>
<td>If contains x then adds x to the set otherwise adds ()</td>
<td>10</td>
<td>14.1%</td>
</tr>
<tr>
<td>If contains x then adds x to the set</td>
<td>2</td>
<td>2.8%</td>
</tr>
<tr>
<td>If contains x then adds x to the set and adds ()</td>
<td>2</td>
<td>2.8%</td>
</tr>
<tr>
<td>If contains x then adds x to the set and does nothing</td>
<td>5</td>
<td>7%</td>
</tr>
</tbody>
</table>
It takes a word and returns the first letter 2 2.8%
It takes an integer and creates a list of size x 2 2.8%
It removes all zeroes from the list 1 1.4%
It takes a list x and returns the sum of the elements in x 32 45.1%
Not sure 21 28.8%

F# code readability

Consider the code in the header of the page. What do you think the function does?

- It sums the result and then squares the sum 0 0%
- It squares each element and then sums the squares 39 54.9%
- It returns an integer containing the value of the sum function and a list containing the squares of each element 1 1.4%
- It says that only the functions “square” and “sum” are applicable on the list 9 12.7%
- Not sure 18 25.4%

Go code readability

Consider the code below. Which answer is correct?

- list has been [0]
- list has been [1]
- list has been [2]

list is initialized to the same type as the returned value from makeList(100), and holds the returned value. 45 63.4%
list has been initialized earlier, and holds the returned value of makeList(100) 9 12.7%
Not sure 12 16.9%

Consider the code below. Which answer is correct?

- someFunction(42)
- someFunction(52)
- someFunction(62)
- someFunction(72)
- someFunction(82)
- someFunction(92)

- No one [0]
- No one [1]
- No one [2]
- No one [3]
- No one [4]
- No one [5]
- No one [6]
- No one [7]
- No one [8]
- No one [9]
- No one [10]
- No one [11]
- No one [12]
- No one [13]
- No one [14]
- No one [15]
- No one [16]
- No one [17]
- No one [18]
- No one [19]
- No one [20]
- No one [21]
- No one [22]
- No one [23]
- No one [24]
- No one [25]
- No one [26]
- No one [27]
- No one [28]
- No one [29]
- No one [30]
- No one [31]
- No one [32]
- No one [33]
- No one [34]
- No one [35]
- No one [36]
- No one [37]
- No one [38]
- No one [39]
- No one [40]
- No one [41]
- No one [42]
- No one [43]
- No one [44]
- No one [45]
- No one [46]
- No one [47]
- No one [48]
- No one [49]
- No one [50]
- No one [51]
- No one [52]
- No one [53]
- No one [54]
- No one [55]
- No one [56]
- No one [57]
- No one [58]
- No one [59]
- No one [60]
- No one [61]
- No one [62]
- No one [63]
- No one [64]
- No one [65]
- No one [66]
- No one [67]
- No one [68]
- No one [69]
- No one [70]
- No one [71]
- No one [72]
- No one [73]
- No one [74]
- No one [75]
- No one [76]
- No one [77]
- No one [78]
- No one [79]
- No one [80]
- No one [81]
- No one [82]
- No one [83]
- No one [84]
- No one [85]
- No one [86]
- No one [87]
- No one [88]
- No one [89]
- No one [90]
- No one [91]
- No one [92]
- No one [93]
- No one [94]
- No one [95]
- No one [96]
- No one [97]
- No one [98]
- No one [99]
- No one [100]
Consider the code below. On what struct / type is the method performed, what variables does it take and what is the return type?

- Performed on struct / type: bool, Input variable: x, Return type: Queue
- Performed on struct / type: *Queue, Input variable: bool, Return type: Node
- Performed on struct / type: Node, Input variable: x, Return type: bool
- Performed on struct / type: Node, Input variable: q, Return type: bool
- Performed on struct / type: *Queue, Input variable: x, Return type: bool
- Performed on struct / type: bool, Input variable: q, Return type: Node

Consider the code below. What values does r hold after execution?

- Unseen: 31
- (0, 1): 7
- (1, 0): 7
- (0, 1, 0): 28
- (1, 0, 1): 2

Go code readability

Consider the code in the header of the page. What do you think is displayed after execution?

- (0, 1, 4, 9, 10, 25, 36, 40, 64, 81): 2
- (0, 1, 10, 25): 40
- (0, 1, 9, 10, 25): 2
- (0, 5): 1
- Not sure: 13

Go code readability

Go map. What is printed out after execution?
Go code readability

What is printed out after execution?

What did you think about F# and Go?

Now that a few code examples has presented for F#, do you think that you would be able to read more code written in F#?

Now that a few code examples has presented for Go, do you think that you would be able to read more code written in Go?

Now that a few code examples has presented for F#, how hard do you think it would be for you to learn F#?
Now that a few code examples has presented for Go, how hard do you think it would be for you to learn Go?

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard</td>
<td>2</td>
<td>2.8%</td>
</tr>
<tr>
<td>Probably hard</td>
<td>5</td>
<td>7%</td>
</tr>
<tr>
<td>Not hard but not easy either</td>
<td>27</td>
<td>36%</td>
</tr>
<tr>
<td>Probably easy</td>
<td>25</td>
<td>36.2%</td>
</tr>
<tr>
<td>Easy</td>
<td>8</td>
<td>11.3%</td>
</tr>
</tbody>
</table>
8.4 Implemented code in F#

module VertexClass

open System;

type Vertex (value: int, adjVertexPointers:int array) =
    //Where "adjVertexPointers" has the index value of the number "value"
    //was made from.
    //In other words, "value", has the value of the number located at index
    //i when creating the Vertex objects from list "List"
    new (value) = Vertex(value)
    new (value, adjVertexPointers) = Vertex(value, adjVertexPointers)

    member this.value = value
    member this.adjVertexPointers:int array = adjVertexPointers

    override this.ToString() =
        if adjVertexPointers.Length = 0 then (string)value
        else (string)value + " " + (string)adjVertexPointers.[0] + " " +
             (string)adjVertexPointers.[1]

module NumberGenerator

open System;

//Creates numbers 1,2,4,7,11,16... n times
let rec helpToPopulateArray n count number (myArray:int array) =
    if count < n then helpToPopulateArray n (count + 1) (number + count)
       (Array.append myArray [|number + count|])
    else myArray

let populateArray n = helpToPopulateArray n 1 1 [|1|]

module Main

open System;
open System.Diagnostics;
open System.Collections.Generic;
open VertexClass;
open NumberGenerator;

let number = 10000
let listOfNumbers:int array = populateArray number
let endVertex = 48595012 //Vertex nr 9858
let VertexQueue = new Queue<(Vertice)>()

//Help method for MakeVerticeList
let rec ConnectEndVertices (listOfNumbers:int array) indexCount
    (verticeList: Vertice array)
    =
        if indexCount < listOfNumbers.Length
           then ConnectEndVertices (listOfNumbers:int array) (indexCount + 1)
              (Array.append verticeList [|Vertice(listOfNumbers.[indexCount], [|[]|])|])
        else verticeList

    //indexCount remembers on which index in listOfNumbers to get the value
//treeLevel tells us at which level we are in the tree, verticeList is the result of the function
//Call MakeVerticeList with 0, 1, 1, []

let rec MakeVerticeList indexCount currentTreeLevel
numbersLeftBeforeLevelChange(verticeList:Vertice array) =
    if indexCount < (listOfNumbers.Length - currentTreeLevel - 1)
        then if numbersLeftBeforeLevelChange = 0
        then MakeVerticeList indexCount (currentTreeLevel + 1)
            (currentTreeLevel + 1) (verticeList:Vertice array)

        //Still have numbers to cover on this level
        else MakeVerticeList (indexCount + 1) currentTreeLevel
            (numbersLeftBeforeLevelChange - 1)(Array.append verticeList
                [Vertice(listOfNumbers.(indexCount), [Vertice(listOfNumbers.(indexCount +
                    currentTreeLevel)); listOfNumbers.(indexCount +
                    currentTreeLevel + 1)])])
    else ConnectEndVertices (listOfNumbers:int array) indexCount
        (verticeList: Vertice array)

let verticeList = MakeVerticeList 0 1 1 []

//Queues all vertices in the interval startVertice to startVertice+treeLevel
let helpToQueueVertices indexFrom treeLevel =
    for i = indexFrom to treeLevel do
        VertexQueue.Enqueue(Array.get verticeList i)

let helpToQueueVertices2 indexFrom indexTo =
    for i = indexFrom to indexTo do
        VertexQueue.Enqueue(Array.get verticeList i)

helpToQueueVertices 0 0 //Enqueue the first vertice before starting the BFS
let rec doTheBFS indexFrom indexTo treeLevel verticesLeftInGraph =
    if verticeList.Length > indexTo
    then while VertexQueue.Count > 0 do
        if VertexQueue.Dequeue().value = endVertice then
            System.Console.WriteLine("I found your vertice")
        if (Array.IndexOf(listOfNumbers, endVertice)) <=
            indexTo
            then doTheBFS indexFrom verticeList.Length treeLevel
                ((indexFrom +
                treeLevel + treeLevel))
                doTheBFS (indexFrom + treeLevel) (indexFrom +
                treeLevel +
                treeLevel) (treeLevel + 1) (number - treeLevel)

let MakeListStopwatch = new Stopwatch()
let MakeGraphStopwatch = new Stopwatch()
let DoBFSSStopwatch = new Stopwatch()

MakeListStopwatch.Start()
let doListOfNumbersAThousandTimes =
    for i = 1 to 10000 do
        populateArray number
    MakeListStopwatch.Stop()

MakeGraphStopwatch.Start()
let doVerticeListAThousandTimes =  
    for i = 1 to 10000 do  
        MakeVerticeList 0 1 1 [[]]  
MakeGraphStopwatch.Stop()  
DoBFSStopwatch.Start()  
    let doBFSAThousandTimes =  
        for i = 1 to 10000 do  
            doTheBFS 0 0 1 number  
DoBFSStopwatch.Stop()  
System.Console.Write("The time of doListOfNumbersAThousandTimes took ")  
System.Console.WriteLine((string)(MakeListStopwatch.Elapsed) + " ")  
System.Console.Write("The time of doVerticeListAThousandTimes took: ")  
System.Console.WriteLine((string)MakeGraphStopwatch.Elapsed + " ")  
System.Console.Write("The time of doBFSHunderedAThousandTimes took: ")  
System.Console.WriteLine((string)DoBFSStopwatch.Elapsed + " ")

8.5 Impleneted code in Go

package main
import (  "fmt"  "time")
type Node struct {  value int64  adjNodes [2]  int visited bool}
type Queue struct {  b []Node  head, tail int}
//how many times to run the test
const AmountOfRuns = 100
func main() {  //How many times the test is run  //time variables  var everythingTime time.Duration  var listTime time.Duration  var graphTime time.Duration  var bfsTime time.Duration  //test-loop  for i:=0; i<AmountOfRuns; i++ {  
    e1 := time.Now()  //create some values for the graph  
    l1 := time.Now()  
    list := makeList(10000)  
    l2 := time.Now()  //build the graph with the values from list  
    g1 := time.Now()  
}
```go
g := makeGraph(list)
g2 := time.Now()
b1 := time.Now() q := new(Queue)
bfs(graph,q,49995001)
b2 := time.Now()
e2 := time.Now()

everythingTime += e2.Sub(e1)
listTime += l2.Sub(l1)
graphTime += g2.Sub(g1)
bfsTime += b2.Sub(b1)

fmt.Println("Everything took: ", everythingTime/AmountOfRuns)
fmt.Println("Making list took: ", listTime/AmountOfRuns)
fmt.Println("Making graph took: ", graphTime/AmountOfRuns)
fmt.Println("Doing bfs took: ", bfsTime/AmountOfRuns)

func bfs (graph []Node, q *Queue, searchValue int64) bool {
currentNode := graph[0]
var notEmpty bool = true
q.Push(currentNode)
for ;notEmpty == true; {
currentNode, notEmpty = q.Pop()
if searchValue == currentNode.value {
    return true
}
for j:=0;j<len(currentNode.adjNodes);j++ {
    if !graph[currentNode.adjNodes[j]].visited {
        graph[currentNode.adjNodes[j]].visited = true
        q.Push(graph[currentNode.adjNodes[j]-1])
    }
}
return false
}

//Returns an array of Nodes with the values from list
func makeGraph (list []int64) []Node {
g := make([]Node, len(list)*2)
g[0].value = 1
g[0].adjNodes[0] = 2
g[0].adjNodes[1] = 3
j := 3
for i:=1;i<len(list);i++ {
g[i].value = list[i]
j++
g[i].adjNodes[0] = j
j++
g[i].adjNodes[1] = j;
}
return g
}

//Returns a list of values of size x
func makeList(x int64) []int64 {
var number int64 = 1
return []int64{
...}
```
var count int64 = 1
list := make([]int64, x)
list[0] = 1
for ; count != x; { 
    list[count] = (number + count)
    number += count
    count++
} return list

// Queue function from http://rosettacode.org/wiki/Queue/Definition#Go
// the zero object is a valid queue ready to be used.
// int queue
// items are pushed at tail, popped at head.
// tail = -1 means queue is full
func (q *Queue) Push(x Node) { switch {
// buffer full, reallocate.
case q.tail < 0:
    next := len(q.b)
    bigger := make([]Node, 2*next)
    copy(bigger[copy(bigger, q.b[q.head:])], q.b[:q.head])
    bigger[next] = x
    q.b, q.head, q.tail = bigger, 0, next+1
// zero object, make initial allocation.
case len(q.b) == 0:
    q.b, q.head, q.tail = make([]Node, 10), 0, 1
    q.b[0] = x
// normal case
default:
    q.b[q.tail] = x
    q.tail++
    if q.tail == len(q.b) { 
        q.tail = 0
    } 
    if q.tail == q.head { 
        q.tail = -1
    }
}
}
func (q *Queue) Pop() (Node, bool) { 
    if q.head == q.tail { 
        z := Node{}
        return z, false
    }
    r := q.b[q.head]
    if q.tail == -1 { 
        q.tail = q.head
    }
    q.head++
    if q.head == len(q.b) { 
        q.head = 0
    }
    return r, true
}
func (q *Queue) Empty() bool { 
    return q.head == q.tail
}
8.6 Implemented code in Java

Node.java

```java
public class Node {
    long value;
    int[] neighbours; boolean visited;

    // A node in the graph. Initializes two edges to neighbouring nodes
    // a value for the node and sets visited to false
    public Node(long value, int left, int right) {
        this.value = value;
        this.neighbours = new int[2];
        this.neighbours[0] = left;
        this.neighbours[1] = right;
        this.visited = false;
    }
}
```

testBfs.java

```java
import java.util.LinkedList;
import java.util.Queue;

public class testBfs {
    // number of runs for the benchmark
    public static final int NUMBER_OF_RUNS = 100;
    // size of the graph
    public static final int SIZE = 10000;

    public static void main(String[] args) {
        long listT = 0;
        long graphT = 0;
        long bfsT = 0;

        // the benchmark loop, times the different parts of the program
        // with System.nanoTime()
        for (int i = 0; i < NUMBER_OF_RUNS; i++) {
            long l0 = System.nanoTime();
            int[] list = makeList(SIZE);
            long l1 = System.nanoTime();

            long g0 = System.nanoTime();
            Node[] graph = makeGraph(list);
            long g1 = System.nanoTime();

            long b0 = System.nanoTime();
            boolean search = bfs(graph, list[SIZE - 1]);
            long b1 = System.nanoTime();

            listT += (l1 - l0);
            graphT += (g1 - g0);
            bfsT += (b1 - b0);
        }

        System.out.println("Make list: "+ listT / NUMBER_OF_RUNS);
        System.out.println("Make graph: "+ graphT / NUMBER_OF_RUNS);
```

System.out.println("Do bfs: " + bfsT/NUMBER_OF_RUNS);
System.out.println("Everything: " + (listT+graphT+bfsT)/NUMBER_OF_RUNS);

//The breadth first search algorithm
public static boolean bfs(Node[] graph, int searchValue) {

LinkedList<Node> q = new LinkedList();
Node currentNode = graph[0];
q.add(currentNode);

boolean notEmpty = true;
while(notEmpty==true) {

    currentNode = q.remove();
    if (searchValue == currentNode.value) {
        //System.out.println(currentNode.value);
        return true;
    }
    if (graph[currentNode.neighbours[0] - 1].visited == false) {
        graph[currentNode.neighbours[0] - 1].visited = true;
        q.add(graph[currentNode.neighbours[0] - 1]);
    }
    if (graph[currentNode.neighbours[1] - 1].visited == false) {
        graph[currentNode.neighbours[1] - 1].visited = true;
        q.add(graph[currentNode.neighbours[1] - 1]);
    }

    return false;
}

//Creates a list with values for the graph
public static int[] makeList(int x) {
    int number = 1;
    int count = 1;
    int[] list = new int[x];

    for (;count!=x;)
        list[count] = (number+count);
    number += count;
    count++;

    return list;
}

//Initializes the graph
public static Node[] makeGraph(int[] list) {
    Node[] graph = new Node[list.length*2];
    Node base = new Node(1,2,3);
    graph[0] = base;

    int j = 3;
    for (int i=1;i<list.length; i++) {
        graph[i] = new Node(list[i], j, j+1);
        j+=2;
    }
    for (int i=list.length; i<list.length*2; i++) {
        graph[i] = new Node(0,0,0);
    }
    return graph;
}