Monday Morning Haskell

Haskell Production Checklist

Essential Libraries for writing Haskell applications

Data Structures

While these aren't necessarily specific to production tasks, you'll find as you write more sophisticated programs that you need to be able to incorporate these packages in order to get performant data structures. Also, many libraries for force you to use these in their APIs. So it pays to know them!

Text

hackage: text

Haskell's built-in String type has a number of problems. It is a pain to represent anything outside ASCII. It isn't especially efficient for large-scale string processing. It's just a list under the hood, which can occasionally be useful, but can sometimes require using an unintuitive API. Text is almost always a better choice for representing strings that will require processing or else that will be used by many different parts of your application. Text operations are subject to fusion, meaning the compiler can combine the operations to be more efficient and avoid unnecessary allocations of data. Text can be used in both strict and lazy forms.

If you're using Text (or ByteString below), you should be aware of the OverloadedStrings extension as well as the StringConv library for conversion.

Bytestring

hackage: bytestring

Bytestring is vey similar to Text in that it serves as a replacement for the String type. In fact, some of its API is identical to Text. Like Text, its operations are subject to fusion and it comes in strict and lazy variants. Unlike Text, ByteStrings are a lower level encoding of the string itself, and this makes them more suitable for serializing data. ByteString should be your go-to when passing information back and forth with another service (for instance if you're using an HTTP Request).

Vector

hackage: vector

Lists are the default container type in Haskell, and they're useful for a lot of functional programming problems. But many problems require operations that linked lists don't perform well, such as access by index or appending to the back. Vector mimics the function performed by arrays in languages such as C++ and Java. They allow constant time access by an integer index, and have reasonable amortized efficiency when appending elements to the front or to the back.

Containers

hackage: containers

Containers is a larger library containing a wide variety of important data structures. Whether you're applying advanced algorithms, or just need to do basic association tasks quickly, you'll probably need at least one of these. The four highlights are sets, hash sets, maps, and hash maps:

Sets (Data.Set) store a series of order-able items using a self balancing tree as the underlying representations.

Hash Sets (Data.HashSet) store a series of Hashable items in an un-ordered fashion. They use a hash table as the underlying representation.

Maps (Data.Map) store a key-value pairs with an ordering based on the key. Like sets, they use a tree as the underlying representation.

Hash Maps (Data.HashMap) store key-value pairs without an ordering, using a hash table as the underlying representation.

Array

hackage: array

Vector performs most of the functions performed by arrays in most other languages. Meanwhile, the array type extends this functionality and allows more flexibility. In particular, you can use any type corresponding to the Ix typeclass as the index type of your array, instead of being forced to use integers. This lets you do a few cool things. You can make the index bounds something other than *[0,(length - 1)]* if you're just using integers. But you can also have your index be a tuple of integers, for instance. This can provide a much more natural API if you're manipulating a 2-D game board. Instead of manipulating a vector of vectors, you can simply index by the tuple *(1,1)* and it will work as expected.

The Array package also include IO Arrays, which are truly mutable. This is necessary for certain algorithms like performing quick sort in-place.

Serialization

Aeson

hackage: aeson

Aeson is a JSON serialization and deserialization library. It provides two typeclasses, FromJSON and ToJSON that allow you to easily encode and decode your objects using JavaScript Object Notation.

XML

hackage: xml

Just as Aeson allows you to encode and decode your objects into JSON, the XML library allows you to encode them as XML.

Store

hackage: store

This library provides the Store typeclass that allows you to specify a compressed encoding of your data. It is necessary for use in conjunction with a couple different libraries.

Parsec

hackage: megaparsec, parsec, attoparsec

Parsec is a generic parsing library that comes in a number of different flavors. Parsec is the original version, which comes with a wide feature set that can parse any kinds of input. Attoparsec is a version which has a more limited feature set but significantly higher performance. In particular, Attoparsec is specialized to Bytestrings. Megaparsec seeks to improve on Parsec, maintaining the wide feature set and also providing better error messages.

Opt-Parse Applicative

hackage: optparse-applicative

This library treats parsing as an applicative task, rather than a monadic task. This leads to some combinators that are more intuitive. It doesn't have quite the same power as a monadic parsing library like parsec, but it's API is somewhat easier to deal with. It also has many convenient functions that allow you to parse command line arguments. You can use these to easily set-up sub-commands of your executables.

Streaming

Conduits

hackage: conduit
related: conduit-combinators, conduit-extra

Conduits is the first of our libraries that allows you to deal with streaming data. This means situations where you either do not know how much data you are receiving, or you want to avoid bringing all the data into memory at once. Conduits allow you to manipulate data as it comes in, pass it along to other sections of your code, and interleave effects among all of this. The library also emphases easy handling of resource cleanup.

Pipes

hackage: pipes

The Pipes library accomplishes many of the same tasks as Conduits. However, it was written with more of an emphasis on composability and semantically intuitive code. Like Conduits, Pipes offers extremely good performance and is well maintained.

Streaming

hackage: streaming

The Streaming library exposes an interface that is somewhat more intuitive and “list-like” than Conduits and Pipes. However it lacks bi-directional streaming which the other two libraries have. It does not have as large of an ecosystem behind it either. It is, however, still a very performant library.

Databases

Persistent

hackage: persistent
related: persistent-postgresql, persistent-sqlite, persistent-mongoDB, persistent-mysql

Persistent allows you to create a database schema using Template Haskell. The generated code will include Haskell definitions of your types as well as lenses to write queries on that data. You can perform many basic queries in a type-safe manner. It works with both SQL and No-SQL databases, though using a specific database platform will require additional packages, like persistent-postgresql, for example.

Esqueleto

hackage: esqueleto

Esqueleto essentially works as a layer on top of a Persistent schema. It depends on all the same lenses that you get from using the Template Haskell approach. It provides an EDSL for composing a more diverse array of SQL queries than Persistent alone. In particular, you can perform type-safe joins using Esqueleto.

Beam

hackage: beam

Beam is another great database library with intuitive syntax for defining your database types and tables. Unlike Persistent, it seeks to avoid using Template Haskell and EDSLs, instead relying on type families and generics. But it still provides the same kind of type safety you'll get with Persistent. Since it doesn't use Template Haskell, there's a bit more boilerplate code involed, but the volume is quite manageable.

Opaleye

hackage: opaleye

Opaleye is more specialized than Beam and Persistent. In particular, it is only written for Postgres databases. Like Persistent, it makes use of Template Haskell for automatically generating boilerplate code around tables. It also uses an EDSL for making type-safe SQL queries.

Redis

hackage: hedis

The Haskell bindings for Redis allow you to interact with a Redis datastore that is either running on your local machine or running remotely. Most of the main Redis commands are available. Note this library works almost entirely through bytestrings, so be sure you know how to serialize your data!

HTTP API/Web Programming

Servant

hackage: servant
related: servant-server, servant-client, servant-docs, servant-blaze

Servant uses advanced type-level machinery to allow you to create a type-safe web API. You can specify endpoints for your API using various combinators, and then write handler functions using the corresponding types. Your handler functions can typically be in terms of the business types of your application, and your handler code is generally doesn't need to do any de-serialization or handling of raw requests.

Servant has many related libraries (the above list is non-exhaustive). These help with tasks such as generating client functions for your API, generating documentation and so on.

Spock

hackage: Spock

Spock is a simple Web API library that is a lighter-weight alternative to Servant. It allows you to specify routes using a similar syntax to Ruby's Sinatra framework. This system is more accessible overall than Servant's type-level approach, but not quite as powerful. It comes with built-in support for items like databases and session handling, so it is a great tool for getting something up and running quickly.

Snap

hackage: snap

Snap aims to be a full-stack web library for Haskell. It provides HTTP routing just like Spock and Servant, but also has tools for front-end templates as well. The library allows you to use modularized pieces of functionality, called Snaplets, to separate concerns within your application.

Yesod

hackage: yesod

Yesod is a heavy duty full stack web library. Like Snap, it combines backend routing, database functionality, and frontend templating. Unlike Snap, Yesod relies heavily on Template Haskell and DSLs in order to accomplish these tasks. The result is a system that has an incredible ability to catch problems at compile time, but it somewhat less accessible to beginners.

Frontend Web

GHCJS (Reflex FRP)

link: reflex-frp

As we saw above, both Snap and Yesod have template options for writing your frontend web pages. But they don't quite give the same level of control over the dynamics of the page that you would get from using Javascript.

GHCJS is an alternative compiler to GHC that allows you to compile your Haskell code into Javascript to run in the browser. Reflex FRP is the most popular library right now for use with GHCJS. It uses the paradigm of Functional Reactive Programming to allow you to deal with a dynamic web page in a pure functional way.

It's extremely cool to use. But the drawback is that it doesn't work too well with Stack or Cabal just yet. You'll want to use the Nix package manager instead, which has its own learning curve. Overall, it is still a great option and a neat tool to learn.

Machine Learning

Tensor Flow

hackage: tensorflow
related: tensorflow-proto, tensorflow-opgen, tensorflow-logging, tensorflow-core-ops (others)

The Tensor Flow bindings for Haskell allow you to hook into the low level C APIs created by Google for fast, distributed computations. You can build up a “graph” of computations, stored as Tensors. Then all these computations are evaluated at once at a lower level. Note that to get even basic functionality, you'll need to include a number of the “related” libraries. At time of writing, it is definitely easier to access this library from Github rather than Hackage.

Grenade

hackage: grenade

Grenade is a library specifically directed towards deep neural networks. It allows you to build networks that combine many different types of layers, including activation functions, dense layers, and convolutional layers. It also allows you to do all this in a type-safe way with dependent types.

HMatrix

hackage: hmatrix

HMatrix is a library for efficient numerical computations. It allows you to do common operations like matrix multiplication and addition very quickly. It is used by Grenade under the hood.