Packing Albums into DVDs
An Exercise for Lazy Functional Programmers

Norman Ramsey
Tufts University

September 5, 2012

The purpose of this exercise is to help you practice the techniques of lazy functional programming as described by John Hughes in his seminal paper *Why Functional Programming Matters*. The problem is as follows:

- I want to archive my music collection on DVD. An album takes up 300–600MB, but a DVD holds 4,700,000B. I want to pack as many albums as possible into a DVD.

  If two different packings use the same number of DVDs, I prefer the one that leaves the most amount of free space on the least full DVD, so that all the other DVDs are as full as possible.

- The problem is NP-hard, but basic step is to use a greedy heuristic:

  1. Put the albums in a finite list. (You may want to try various orders.)
  2. Start with an infinite list of empty DVDs.
  3. Run this algorithm:

     ```
     repeat
       take the first album from the list
       put the album in the first DVD that has room for it
     until there are no more albums on the list
     ```

     Burn all the nonempty DVDs.

Implement the standard greedy algorithm by writing a function

```
pack :: [Album] -> [DVD]
```

where

```
type Album = Album { name :: String, size :: Integer }
type DVD = [Album]
```

Decompose your solution into separate functions as described by Hughes.

You can find a sample list of albums at `http://www.cs.tufts.edu/comp/150GIT/Albums.hs`.