A4: Instructions

February 1, 2019

1 General Instructions

You are given five files in this assignment as starters. You should not edit the following three files:

- `List`: this should be a familiar implementation of lists by now. The method `MakeList` can be used to create large random lists for testing. There are examples of its use in `List` and `DPTest`.
- `Pair`: this is a simple class that can be used to group two values.
- `Coin`: again a fairly simple class to represent an abstraction of a "coin."

You will notice that three files above override `toString`, `equals`, and `hashCode` explicitly. The new `toString` methods are there for convenience only. The `equals` and `hashCode` methods are however absolutely essential for our assignment. To understand why, note that the default implementation of `hashCode` will likely return *different* hashes for two different objects, i.e.,

```
new A().hashCode() == new A().hashCode()
```

is usually false. Indeed, the documentation for `hashCode` in `java.lang.Object` says:

> As much as is reasonably practical, the hashCode method defined by class Object does return distinct integers for distinct objects.

Since we are using hash tables to store results of subproblems, we want to make sure that two representations of the same subproblem hash to the same value. So any two lists containing 1,2,3 should hash to the same value. For sanity, we also override the `equals` to reflect this notion of identity.

After making yourself familiar with the above three files, all your work will be in the following two files:

- `DP`
- `DPTest`

We are providing quite a few tests. You *must* write additional tests for `ffib`, `coinChange` and its memoized version, `partition` and its memoized version, `minDistance` and its memoized version, `lcs` and its memoized version. Your tests should be a mix of correctness tests and timing tests.

2 Fibonacci

You are given the usual recursive definition of `fib` and its memoized version. There is also a direct definition:

$$\mathit{ffib}(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Implement this definition and test it for both correctness and speed. For really large values of n, how does this compare to the memoized version? Do you think this implementation is $\mathcal{O}(1)$?

3 Coins

You are given the solution to this problem. Make sure you understand it. Write several additional test cases.

4 Partition

You are given a list of numbers and a target sum. Your job is to determine whether there is a subsequence of the list that sums to the given target. A subsequence of a list contains some of the list of the elements in the same order as they appear in the original list. So for example, if the given list is:

```
[1,5,2,4,8,3]
```

then `[1,8]` is a subsequence but `[8,1]` is not.

5 Minimum Distance

You are given two lists of DNA bases and you need to determine how "close" they are to each other. The definition of "close" is determined by the following notion of cost calculated as you traverse the two lists comparing elements:

- if the two elements under consideration match the cost is 0;
- if the two elements do not match, then you have three options:
  - you could add a cost of 1 for a mismatch and continue calculating the cost with the remainders of the two lists;
  - you could shift one of the lists by one position but paying a cost of 2.

For example, if the two lists were:

```
[T,A,T,A,C]
[A,T,A,C]
```

When comparing the first two elements, you could decide there is a mismatch paying a cost of 1, then you pay again for the next two elements, and again for the ones after that. Or you could pay a cost of 2 initially shifting the bottom list to make a perfect match.

Your goal is to calculate the minimum cost.

6 Longest Common Subsequence

In this last problem, you are given two lists of characters and your job is find the long common subsequence. For example, if the two lists are:

```
[a,b,c,d,a,a,a,b,a,c]
[a,a,a,a,b,d,d,a,c]
```

There are several common subsequences like `[b,c]`, `[b]`, `[a,a,a,a,a]`. Your job is to compute the longest such subsequence.