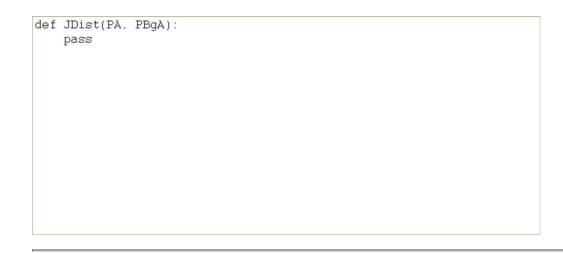
Part 1: Creating a joint distribution

Write the JDist function, which will represent the joint probability distribution for two variables A and B. The function is given as inputs PA, which is a DDist representing the probability distribution for A, and PBgA, which is a function that takes a value of A as input and returns a DDist representing the conditional probability of B given A (i.e., P(B | A)). Consider the following example:

JDist should return a DDist whose items are tuples of the form (a,b), with values that are the corresponding joint probabilities. In the tuple (a,b), a is an element in the support of P(A) and b is an element in the support of $P(B \mid A)$

JDist is in the dist module, so it has access to DDist directly. So, in your solution, you do not need to use dist.DDist; you can just DDist.



Part 2: Implement marginalization

Add the marginalizeOut(self, index) method to the DDist class; index is an integer specifying which of the variables to marginalize out; if index is 0, then marginalize out the first variable, if index is 1, then marginalize out the second variable. The method should return a DDist.

Continuing the example in the previous part of this problem:

>>> JDist(disease, PTgD).marginalizeOut(0)
DDist(posTest: 0.540000, negTest: 0.460000)

The following function is already defined:

```
def removeElt(items, i):
    result = items[:i] + items[i+1:]
    if len(result) == 1:
        return result[0]
    else:
        return result
```

This removes the *i*th entry from a tuple. However, if the resulting tuple has a single element, it just returns that element, instead of a tuple of one element.

The following function is also defined:

```
def incrDictEntry(d, k, v):
    if d.has_key(k):
        d[k] += v
    else:
        d[k] = v
```

This increments the key k in dictionary d by the value v. If the key does not already exist, it behaves as if the key existed with the value 0.

```
class DDist:
    def __init__(self, dictionary):
        self.d = dictionary
    def prob(self, elt):
        if self.d.has_key(elt):
            return self.d[elt]
        else:
            return 0
    def support(self):
        return [k for k in self.d.keys() if self.prob(k) > 0]
    def marginalizeOut(self, index):
        pass
```

Part 3: Implement conditioning

When we condition on one of the random variables having a particular value, that variable becomes irrelevant and need not be represented. Conditioning effectively selects a row or a column of the joint distribution and normalizes it appropriately. Implement the conditionOnVar(self, index, value) method of the DDist class; index is 0 or 1, determining which variable is being specified, and value gives the value that variable is being assumed to have. The method should return a DDist over the variable

that was left unspecified. Continuing the example:

>>> JDist(disease, PTgD).conditionOnVar(1, 'posTest')
DDist(noDisease: 0.833333, disease: 0.166667)

The removeElt function is also defined for this problem.

```
class DDist:
    def __init__(self, dictionary):
        self.d = dictionary
    def prob(self, elt):
        if self.d.has_key(elt):
            return self.d[elt]
        else:
            return 0
    def support(self):
        return [k for k in self.d.keys() if self.prob(k) > 0]
    def conditionOnVar(self, index, value):
        pass
```

MIT OpenCourseWare http://ocw.mit.edu

6.01SC Introduction to Electrical Engineering and Computer Science Spring 2011

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.