BCH 571 Bioinformatics for Life Scientists

Lab 8

In class, I presented the ImprovedBreakPointReversalSort algorithm:

```
ImprovedBreakpointReversalSort(π)
  while b(π) > 0
    if π has a decreasing strip
      choose ρ(i,j) such that b(π · ρ(i,j)) is minimized
    else
      choose ρ(i,j) that flips an increasing strip
    output ρ(i,j)
    π ← π · ρ(i,j)
    output π
  return
```

where π is a permutation, ρ(i,j) is a reversal, and b(π) is the number of breakpoints in π. Remember that π is augmented with a 0 at the start and n+1 at the end.

Part 1:
The step “choose ρ(i,j) such that b(π · ρ(i,j)) is minimized” is not detailed. Write a function (called minimizebreakpoint) that returns a tuple of i and j for which b(π · ρ(i,j)) is minimized. The input is π, which should be provide as a list.

It will be convenient to write a function that determines the number of breakpoints (called breakpoints), given π, and a function that determines π · ρ(i,j) (called reversal), given π, i, and j.

Part 2:
With minimizebreakpoint in hand, ImprovedBreakpointReversalSort is straight-forward to implement. Implement ImprovedBreakpointReversalSort with input π, a list.

It will be convenient to write a function (called hasdecreasingstrip) that returns a Boolean that indicates TRUE if there is a decreasing strip in π.

Try running with π = (0 8 7 6 1 3 4 2 5 9) (where π is already augmented with 0 and 9=n+1.).