**Problem 3-1.**

1. f. .

By definition, for any input, since the approximation ratio for a maximization algorithm is specified as the minimum, , or .

2. .

By definition, for any input, since the approximation ratio for a minimization algorithm is specified as the maximum, , or .

**Problem 3-2.**

1. u0 is an initial sequence; u1 is the resulting sequence; p(i, j) is the reversal that have been chosen; b(u1) is the number of breakpoints that remain after performing the reversal.

Step 1

u0 = 0 3 4 6 5 8 1 7 2 9, p(6, 7)

u1 = 0 3 4 6 5 8 7 1 2 9, b(u1) = 5

Step 2

u0 = 0 3 4 6 5 8 7 1 2 9, p(1, 7)

u1 = 0 1 7 8 5 6 4 3 2 9, b(u1) = 4

Step 3

u0 = 0 1 7 8 5 6 4 3 2 9, p(2, 8)

u1 = 0 1 2 3 4 6 5 8 7 9, b(u1) = 3

Step 4

u0 = 0 1 2 3 4 6 5 8 7 9, p(5, 6)

u1 = 0 1 2 3 4 5 6 8 7 9, b(u1) = 2

Step 5

u0 = 0 1 2 3 4 5 6 8 7 9, p(7, 8)

u1 = 0 1 2 3 4 5 6 7 8 9, b(u1) = 0

2.

Step1:

0 3 4 6 5 8 7 1 2 9, b(u)=7, p(6,7)

Step2:

0 3 4 6 5 2 1 7 8 9, b(u)=5, p(3,6)

Step3:

0 3 4 1 2 5 6 7 8 9, b(u)=4, no decreasing strips

3.

The sequence found by ImprovedBreakpointReversalSort() is one of the shortest.

**Problem 3-3.**

Step 1:

0 1 4 5 2 3 6, p(2,4), b(u)=3, no decreasing strips

0 1 2 5 4 3 6, b(u)=2

Step 2:

0 1 2 5 4 3 6, p(3, 5)

0 1 2 3 4 5 6, b(u) = 0

**Problem 3-4.**

, ,

, breakpoints: 12, 24, 35

, breakpoints: 24, 35

, breakpoints: 14, 25

**Problem 3-5.**

DPCHANGE\_2(M,c,d)

1 bestNumCoins[0] := 0

2 for m := 1 to M

3 bestNumCoins[m] := *inf*

4 corrComboCoins[m] := *empty-sequence*

5 for i := 1 to d

6 if m >= c[i]

7 if bestNumCoins[m−c[i]] + 1 < bestNumCoins[m]

8 bestNumCoins[m] := bestNumCoins[m−c[i]] + 1

9 corrComboCoins[m] := *merge*(c[i], corrComboCoins[m−c[i]])

10 return bestNumCoins [M], corrComboCoins[M]