Andrew Letz

CIS 315

Professor Wilson

Assignment #5

Question 1

A_1 = 5 x 10, A_2 = 10 x 3, A_3 = 3 x 12, A_4 = 12 x 5, A_5 = 5 x 50, A_6 = 50 x 6

Using the recursive definition m[i, j] to find m[1, n]:

m[1,1], m[2,2], m[3,3], m[4,4], m[5,5], m[6,6] = 0

---

m[1,2] = m[1,1] + m[2,2] + p0*p1*p2 = **150**

m[2,3] = m[2,2] + m[3,3] + p1*p2*p3 = **360**

m[3,4] = 0 + p2*p3*p4 = **180**

m[4,5] = 0 + p3*p4*p5 = **3000**

m[5,6] = 0 + p4*p5*p6 = **1500**

---

m[1,3] = min(m[1,1] + m[2,3] + p0*p1*p3, m[1,2] + m[3,3] + p0*p2*p3) = **330**

m[2,4] = min(m[2,2] + m[3,4] + p1p2*p4, m[2,3] + m[4,4] + p1*p3*p4) = **330**

m[3,5] = min(m[3,3] + m[4,5] + p2*p3*p5, m[3,4] + m[5,5] + p2*p4*p5) = **930**

m[4,6] = min(m[4,4] + m[5,6] + p3*p4*p6, m[4,5] + m[6,6] + p3*p5*p6) = **1860**

---

m[1,4] = min(m[1,1] + m[2,4] + p0*p1*p4, m[1,2] + m[3,4] + p0*p2*p4,
            m[1,3] + m[4,4] + p0*p3*p4) = **405**

m[2,5] = min(m[2,2] + m[3,5] + p1*p2*p5, m[2,3] + m[4,5] + p1*p3*p5,
            m[2,4] + m[5,5] + p1*p4*p5) = **2430**

m[3,6] = min(m[3,3] + m[4,6] + p2*p3*p6, m[3,4] + m[5,6] + p2*p4*p6,
            m[3,5] + m[6,6] + p2*p5*p6) = **1770**

---

m[1,5] = min(m[1,1] + m[2,5] + p0*p1*p5, m[1,2] + m[3,5] + p0*p2*p5,
            m[1,3] + m[4,5] + p0*p3*p5, m[1,4] + m[5,5] + p0*p4*p5) = **1655**

m[2,6] = min(m[2,2] + m[3,6] + p1*p2*p6, m[2,3] + m[4,6] + p1*p3*p6,
            m[2,4] + m[5,6] + p1*p4*p6, m[2,5] + m[6,6] + p1*p5*p6) = **1950**

---

m[1,6] = min(m[1,1] + m[2,6] + p0*p1*p6, m[1,2] + m[3,6] + p0*p2*p6,
m[1,3] + m[4,6] + p0*p3*p6, m[1,4] + m[5,6] + p0*p4*p6,
m[1,5] + m[6,6] + p0*p5*p6) = **2010 < final minimum flops**

0	150	330	405	1655	2010
	0	360	330	2430	1950
		0	180	930	1770
			0	3000	1860
				0	1500
					0

---

s table computed by MATRIX-CHAIN-ORDER:

s	**1**	**2**	**3**	**4**	**5**	**6**
1	0	1	2	2	4	2
2		0	2	2	2	2
3			0	3	4	4
4				0	4	4
5					0	5
6						0

PRINT-OPTIMAL-PARENS(s, 1, 6) would return:

(A1*A2)((A3*A4)(A5*A6)) with a flop count of 2010

Question 2

(a) M(i) = maximum number of robots killed up to second i

(b) M(0) = 0

M(1) = min(x_1, f(1))

```
M(i) = max {for all i > j >= 0} of ( M(j) + min(x_i, f(i - j)))
```

(c) x = [x_1, x_2, … x_n], f = [f(1), f(2), … f(n)], sol = []//initialized to 0s

```
def maxEmp(x, f, i):
      for (i = 2; i <= n; i++):
            maxArray = []
            for (j = i - 1; j >= 0; j--):
                  maxArray.append(sol[j] + min(x_i, f[i - j]))
            sol[i] = max(maxArray)
      return sol[i]
```

(d) Time complexity: $O(n^2)$, Space complexity: $O(n)$

Question 3

(a) L(i, j) = longest antidromic subsequence from s_i to s_j

(b)

$$L(i, j) = \begin{cases} 0 & \text{if } i == j \\ \max(L(i + 1, j), L(i, j - 1)) & \text{if } s_i == s_j \\ 2 + L(i + 1, j - 1) & \text{if } s_i \mathrel{!}= s_j \end{cases}$$

(c) string s = s_1, s_2, … s_n, sol = [][] // initialized to 0s

```
def LAS(s, i, j):
      if (i == j):
            sol[i][j] = 0
      for (k = 2; k < j + 1; k++):
            for (y = 0; y < (j - k + 1); y++):
                  z = y + k - 1
                  if (y == z):
                        sol[y][z] = 0
                  elif(s[y] == s[z]):
                        sol[y][z] = max(sol[y+1][z], sol[y][z-1])
                  else:
                        sol[y][z] = 2 + sol[y+1][z-1]
      return sol[i][j]
```

(d) Time complexity: $O(n^2)$, Space complexity: $O(n^2)$

Question 4

```
d = [d1, d2, … dn]
C = [] // Minimum number of coins at each point, D = [] // denoms used
def iterCoin(T, d):
      n = length(d)
      C[0] = 0
      for (t = 1; t++; t <= T):
            min = INT_MAX
            min_d = 0
            for (j = 1; j <= n; j++): // n number of denominations
                  temp = C[t - d[j]]
                  if (temp > 0 && temp < min):
                        min = 1 + C[t - d[j]]
                        min_d = d[j]
            D.append(min_d) // add what coin we used to the D array
            C[t] = min
      print("Minimum number of coins to make {} is {}".format(T, C[T]))
      print("Used a
      for each coin in D: // print each coin denomination we used
            print(" {} coin,".format(coin))
```