Emily Crilley

Analysis of Algorithms
Homework 5

1. A1 - 5 × 10, A2 - 10 × 3, A3 - 3 × 12, A4 - 12 × 5, A5 - 5 × 50, A6 - 50 × 6
   a. Optimal Parenthesization: (((((([5,10] [10,3]) [3,12]) [12, 5]) [5,50]) [50,6])
   Table:

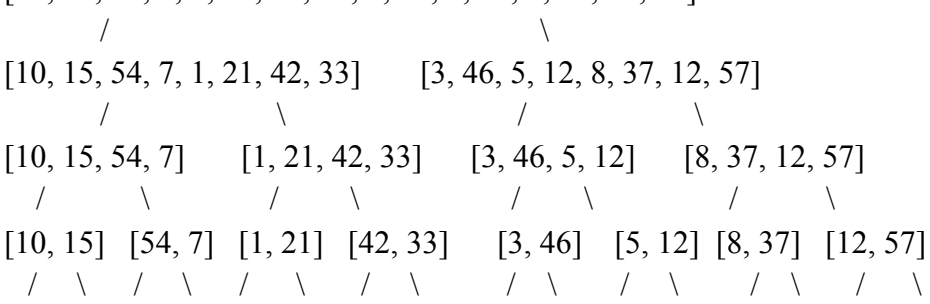
0	150	330	405	1655	2010
0	0	360	330	2430	1950
0	0	0	180	930	1770
0	0	0	0	3000	1860
0	0	0	0	0	1500
0	0	0	0	0	0

   b.

   parenthesis(1) = 0
   parenthesis(2) = 1
   parenthesis(3) = 2
   parenthesis(n + 1) = n - 1

   Where n = 2
   parenthesis (2+1) = 2- 1 = 2
   parenthesis(3) = 2

2.
   a. A recursive matrix-chain would be the best choice, as the recursive chain has $\frac{1}{2}n^2$ unique checks. Enumerating through all the ways would result in $(n-1)^{n-1}$ unique checks.
   b. Having to allocate additional memory to an algorithm would worsen its time complexity.



[10] [15] [54] [7] [1] [21] [42] [33] [3] [46] [5] [12] [8] [37] [12] [57]

\ / \ /

[1, 7, 10, 15, 21, 33, 42, 54] [3, 5, 8, 12, 12, 37, 46, 57]

\ /

[1, 3, 5, 7, 8, 10, 12, 12, 15, 21, 33, 37, 42, 47, 54, 57]

c. This problem exhibits optimal structure. The maximization of scalar multiplications will ensure the best multiplication is the higher number instead of the lower one.

3. See hw5.py
4.
   a. knapsack(items, weight):
      if items is empty or weight is 0:
         return 0
      else:
         i = max(i.value/i.weight, i::items)
         i.value + knapsack(items - i, weight - items.weight)
   b. See hw5.py
   c. See hw5.py
5.
   a. e(S, M, i, j):
      if S == [] or i > j:
         return M
      else:
         return e(S, M-(len(S[i]) + 1), i+1, j)
   b. See printPar.py
   c. bl(S, M, i, j):
         if M < 0:
            return infinity
         else:
            return e(S, M, i, j)
   d. See printPar.py
   e. mb(S, M):
      while e(S, M, 0, infinity) >= 0:
         bl(S, M, 0,infinity)
      bl(S[: mb' (S, M, 0)], M, 0, mb' (S, M)) + mb(S[: mb' (S, M, i)])
   f. $mb'(S, M, i)$:
      if i > |S| or e(S[:i], M, 0, infinity) < 0:
         return i -1
      else:

$mb'(S,\ M,\ i+1)$

6. d values:

s	t	x	y	z
∞	∞	∞	∞	0
2	∞	7	∞	0
2	5	7	9	0
2	5	6	9	0
2	4	6	9	0

π *values* :

s	t	x	y	z
NIL	NIL	NIL	NIL	NIL
z	NIL	z	NIL	NIL
z	x	z	s	NIL
z	x	y	s	NIL
z	x	y	s	NIL

Weight of the edge at 4:
d Values:

s	t	x	y	z
0	∞	∞	∞	∞
0	6	∞	7	∞
0	6	4	7	2
0	2	4	7	2
0	2	4	7	-2

π *values* :

s	t	x	y	z
NIL	NIL	NIL	NIL	NIL

NIL	s	NIL	s	NIL
NIL	s	y	s	t
NIL	x	y	s	t
NIL	x	y	s	t