Benjamin Fondell

Homework 3

1.

Length i	1	2	3	4
value Pi	1	12	21	20
Pi/i	1	6	7	5

Consider a rod of length 4. For the values of length i above, let Pi be value or price of rod at length i and let Pi/i be the value Pi for individual section of i. By the greedy we would cut a rod of length 3 as this has the highest ration of value per segment of i. This leaves a rod of length 1. This results in a total value of 22 for the two segments which upon inspection of the table we see is suboptimal. The optimal way of cutting the rod would be into two sections of length 2 resulting in a total value of 24. This proves that the greedy strategy does not always determine the an optimal way of cutting rods.

2.

A. 1 + (4 x 3) + 2 + (3 x 4) + 2 = 29

B.

$$OPT[j] \quad \begin{cases} max\{OPT[j-1]+v_j, OPT[j-2]+v_j * v_{j-1}\} & \text{if } j \geq 2 \\ v^1 & \text{if } j = 1 \\ 0 & \text{if } j \quad 0 \end{cases}$$

C.

The asymptotic running time of the algorithm would be linear or Θ(n). The algorithm could be written such that it's runtime depends only on the number of values, n that are passed to it. The program would accept an array and the size of the array. The program would check for the cases that the size of the array is 0 and the size of the array is 1, returning if either are true. The program would then pass through the array from beginning to end comparing the maximum {sum or product} of adjacent values. Because the program requires only one pass through all values of the array it is possible to assume the program's runtime is bounded by Θ(n).

Benjamin Fondell

3.

A.

1. Program accepts an integer value that represents the change we are trying to make with cons. Program also accepts an array of denominations of coins.
2. Program creates an array to handle the minimum amount of coins for all values 2….n with n being the value in change passed to the program. (C[p]) for values of p from p…n
3. Program creates an array to handle the last used coin to make change for particular value of change. S[p]
4. Program handles base cases that p == <0 or p == 0. If either it sets the value C[0] = 0 and C[<0] = ∞.
5. Creates loop from values of p from 2….n
6. sets a int variable min to ∞, this will represent the minimum number of coins used to make change for each value p
7. Loops for values in array of coin denominations from 1…k
8. if value of p is greater than the value at denominations[index]
9. Then if the value held at C[p - denominations[index]] + 1 is less than min set min to that value
10. set the value of C[p] to min
11. set the value of S[p] to the coin at denomination[index]
12. loop through all values in denominations array
13. loop through for all values of p from p..n

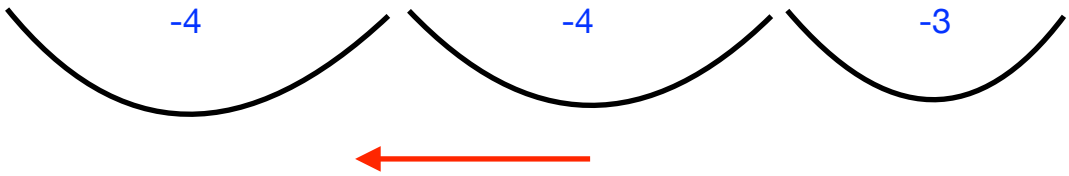
```
change( A, denom[])

int min
int coin
C[<0] = ∞
C[0] = 0
for p = 2 to A
        min = ∞
        for i = 1 to length(denom)
                if (p >= denom[i])
                        if (C[p - denom[i]] + 1 < min)
                                min = C[p - denom[i] + 1
                                        coin = i
        C[p] = min
        S[p] = coin
```

Benjamin Fondell

B.

p	0	1	2	3	4	5	6	7	8	9	10	11
C[p]	0	1	2	1	1	2	2	2	2	3	3	3
S[p]	0	1	1	3	4	1	3	3	4	1	3	3



Minimum coins: 3
Coins used: 3 4 4

C. The running time for this program would be O(Ak) where k is the length of the array of coin denominations. With the exception of the loops through an array from 1…A and the array of coin denominations all other operations are constant. Therefore the running time is bounded by the execution of passing through the two loops of indeterminate length.

5. A.

F(A)						
Amount	Run 1	Run 2	Run 3	Time(Average of Three Runs)		
3	1.70E-05	1.60E-05	2.00E-05	1.77E-05		
6	2.50E-05	2.30E-05	1.70E-05	2.17E-05		
9	3.00E-05	2.10E-05	1.90E-05	2.33E-05		
12	3.20E-05	2.30E-05	2.50E-05	2.67E-05		
15	3.80E-05	4.00E-05	3.40E-05	3.73E-05		
18	4.20E-05	3.80E-05	3.80E-05	3.93E-05		
21	5.10E-05	8.80E-05	6.40E-05	6.77E-05		
24	5.20E-05	4.20E-05	5.00E-05	4.80E-05		
27	5.60E-05	6.20E-05	5.70E-05	5.83E-05		
30	5.80E-05	5.60E-05	6.10E-05	5.83E-05		

Benjamin Fondell

F(n)

n coins	Run 1	Run 2	Run 3	Run 4	Run 5	Time (Average of Five Runs)
3	0.000123	0.000197	0.00015	0.00015	0.000102	0.000144
4	0.000138	0.00022	0.000167	0.000167	0.000116	0.000162
5	0.0002	0.000276	0.000172	0.000172	0.000194	0.000203
6	0.000188	0.00038	0.000188	0.000188	0.000223	0.000233
7	0.000421	0.000323	0.000335	0.000335	0.000193	0.000321
8	0.000224	0.000331	0.000238	0.000238	0.000178	0.000242
9	0.000273	0.000437	0.000342	0.000342	0.000213	0.000321
10	0.000326	0.000447	0.00042	0.00042	0.000224	0.000367
11	0.000332	0.000454	0.000397	0.000397	0.000316	0.000379
12	0.000406	0.00045	0.000345	0.000345	0.000297	0.000369

F(nA)

n*A	Run 1	Run 2	Run 3	Run 4	Run 5	Time(Average of Five Runs)
n=3 ,A=3	5.30E-05	2.60E-05	4.30E-05	2.00E-05	2.50E-05	3.34E-05
n=4, A=6	2.10E-05	2.00E-05	1.90E-05	5.30E-05	2.80E-05	2.82E-05
n=5, A=9	3.90E 05	3.90E 05	3.70E 05	3.20E 05	3.70E 05	3.68E 05
n=6, A=12	3.60E-05	3.90E-05	3.10E-05	4.20E-05	4.20E-05	3.80E-05
n=7, A=15	4.80E-05	4.00E-05	7.10E-05	4.00E-05	4.30E-05	4.84E-05
n=8, A=18	7.20E-05	4.80E-05	4.80E-05	0.000225	4.60E-05	8.78E-05
n=9, A=21	8.80E 05	8.00E 05	7.20E 05	6.60E 05	5.80E 05	7.28E 05
n=10, A=2	9.80E-05	9.70E-05	7.20E-05	9.90E-05	8.20E-05	8.96E-05
n=11, A=2	0.000108	8.90E-05	0.000144	0.000124	0.000122	1.17E-04
n=12, A=3	0.000133	0.000127	0.000141	0.00014	0.000107	1.30E-04

Data Selection and Runtime Collection:

Data Selection F(A):

Data was selected for A using multiples of 3 from 3….30. This allows for a consistent runtime increase as values of A increased. With n values of [1 , 5 , 10 , 25] no coin was able to shortcut the value to make change consistently with the same denomination.

Data Selection F(n):

Data was selected for n by incrementing the value of n by one for every test call to the change making DP algorithm. The n values followed a sequence of 3,4,5…12. Each list of coins for n followed the sequence, [1,2,3] , [1,2,3,4] , [1,2,3,4,5] …. These values were chosen to represent a consistent increase in values for n so the data collected for time would be proportional to the increase in value for n.

Benjamin Fondell

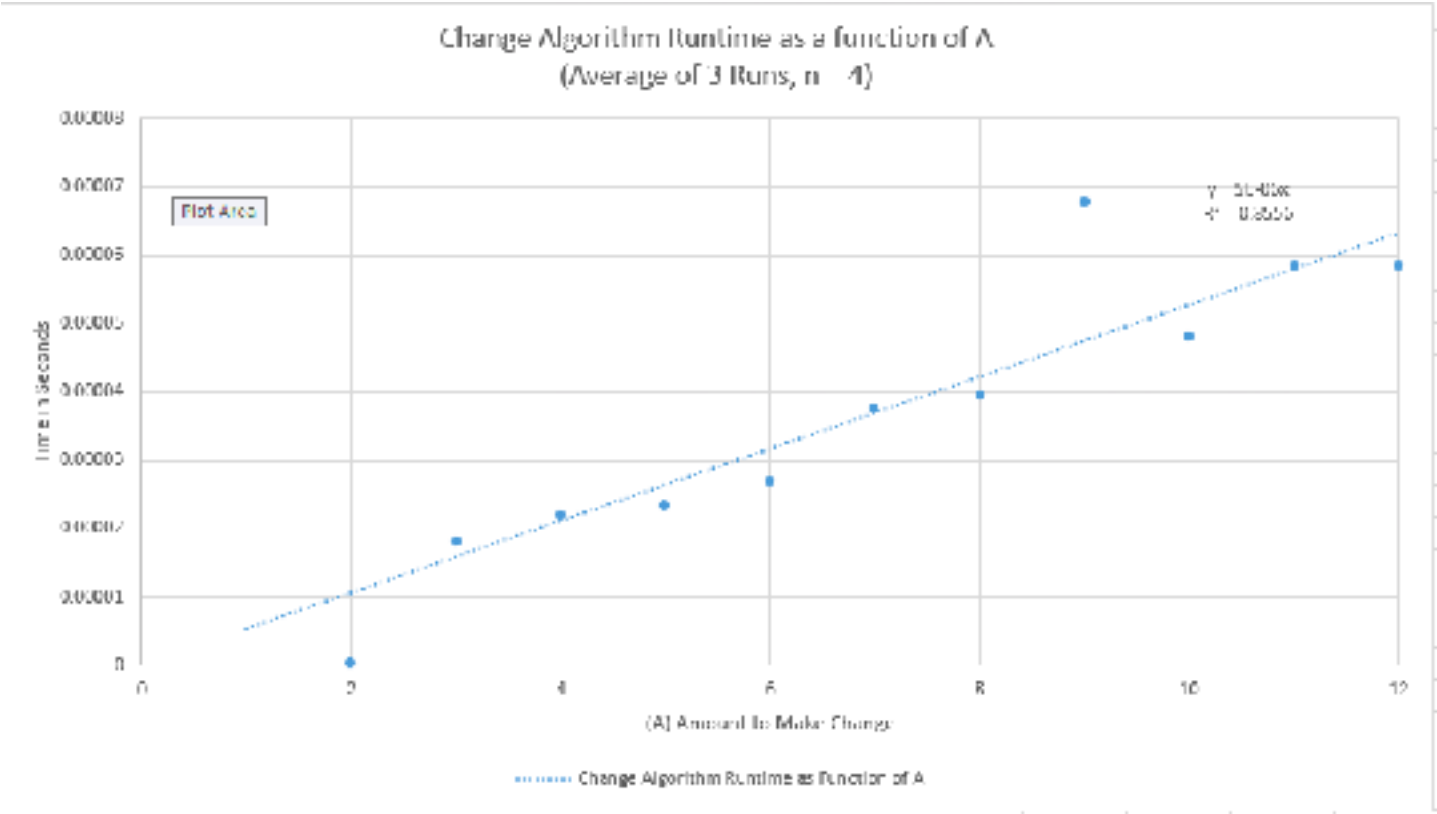
Data Selection F(nA):

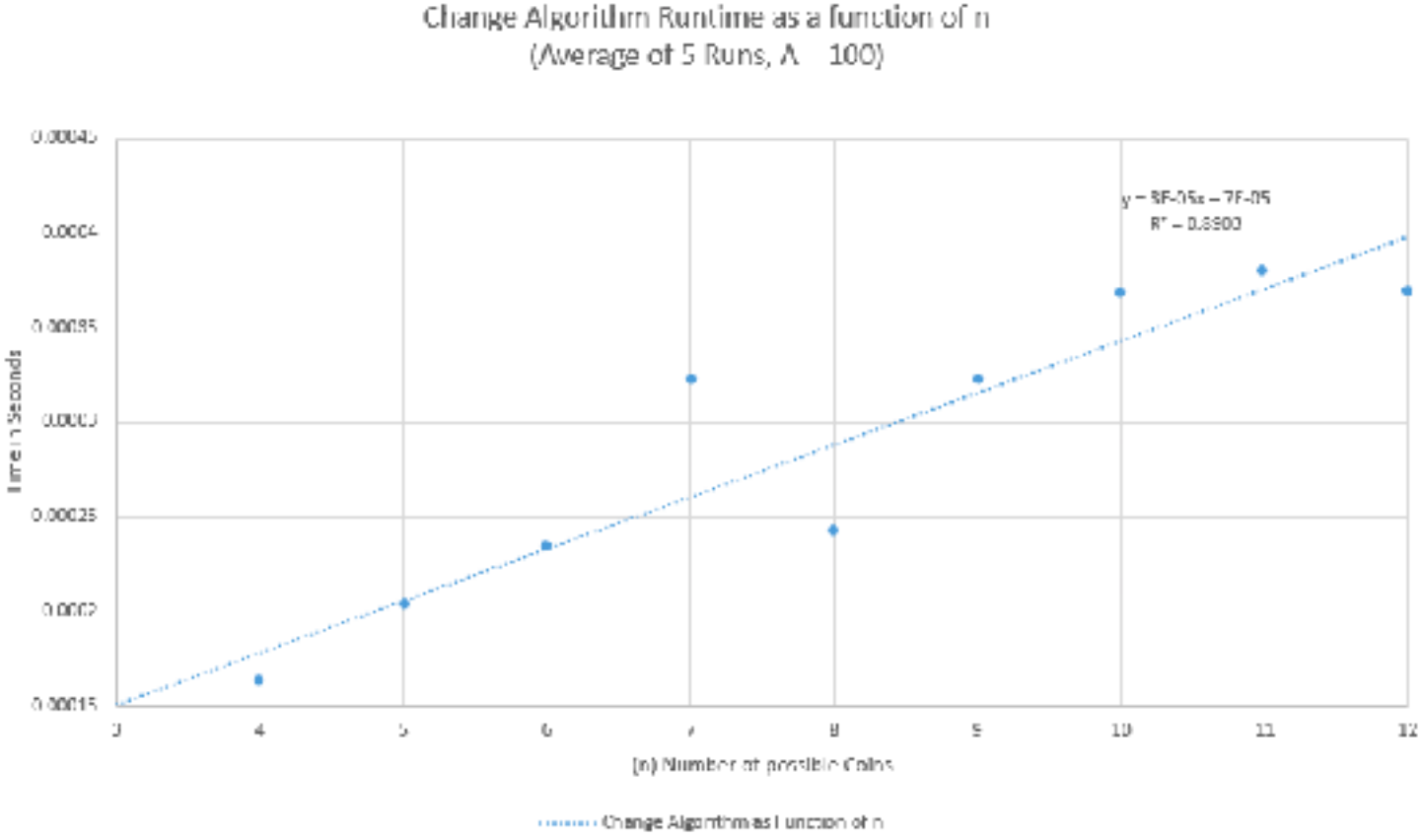
Data selected for nA was the same as the data chosen for A and n. This data was selected to represent increase in runtime for increases in both variables A and n.

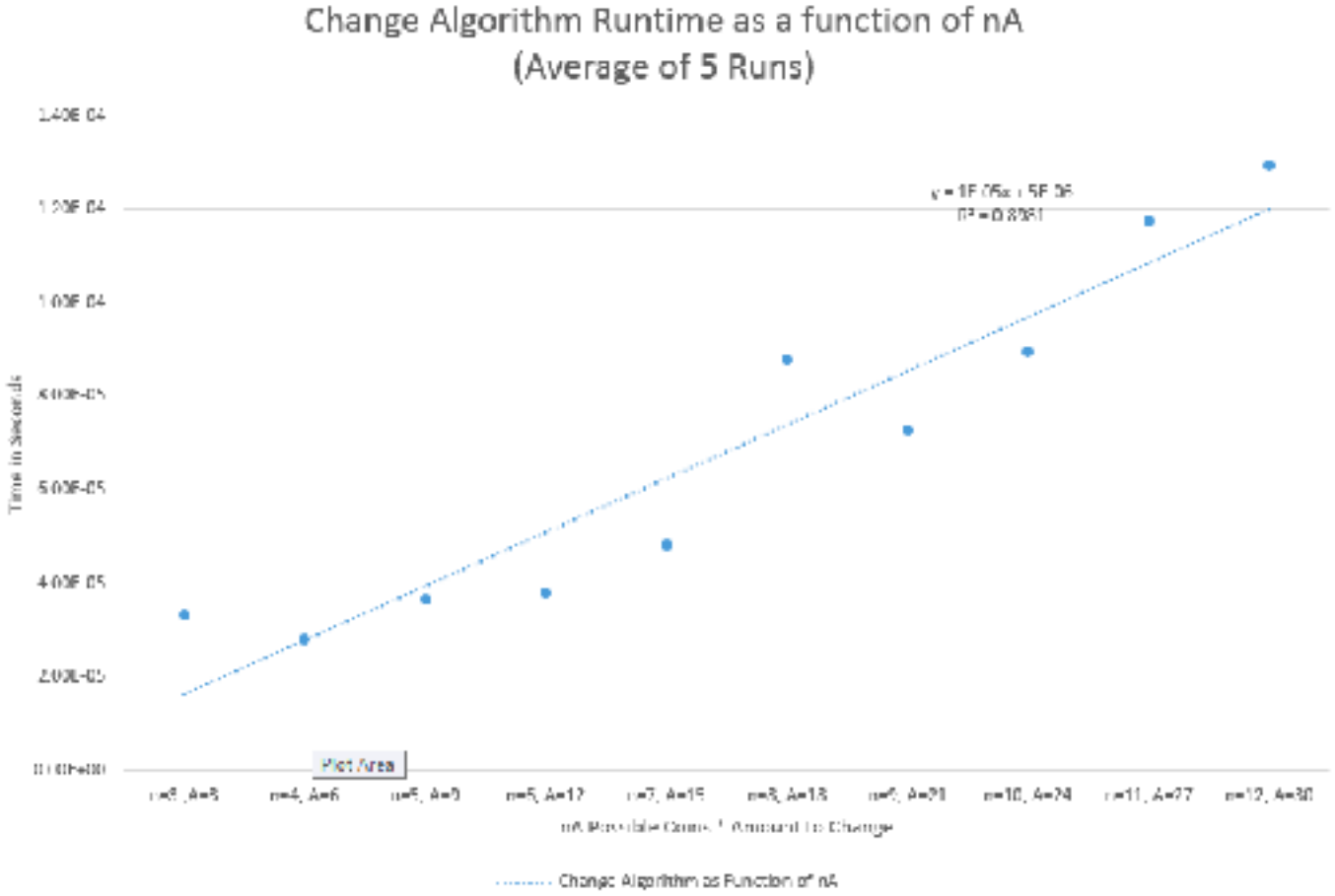
Runtime Collection:

Running times were collected by creating a text file with these values and passing them to the program. Time was then collected before and after the call to the change making DP function. The difference

B.




Change Algorithm Runtime as a function of n
(Average of 5 Runs, A 100)
Time in Seconds
(n) Number of possible Coins
Change Algorithm as Function of n


Change Algorithm Runtime as a function of nA
(Average of 5 Runs)
Time in Seconds
Plot Area
Change Algorithm as Function of nA

b cont'd)

The results of the above graphs are indicative of a runtime of O(nA). This is true because for every value of A the algorithm must cycle through all values of n. Thus we can say that the runtime is bounded by O(nA) and can be considered multilinear assuming that either n or A is constant.