Homework 5

1) Coin-Row Problem

C = [5, 1, 2, 10, 6]

F0 = 0, f1 = 5, fi = max (ci + fn-2, fi-1)

i = 2: max (1 + F(0), F(1)) => max (1 , 5) == 5 cents

i = 3: max (2 + F(1), F(2)) => max (7 , 5) == 7 cents

i = 4: max (10 + F(2), F(3)) => max (15 , 7) == 15 cents

i = 5: max (6 + F(3), F(4)) => max (13, 15) == 15 cents

Max change we can get is 15 cents picking the 1st and 4th coins

2) Change-Making Problem (find all solutions)

N=9, C = [1, 3, 5]

N0 = 0, i 1>n, j 1>m and I > dj, temp = min (n(i-dj), temp)

i = 1: N(i) = temp +1 => 1 coin

j = 1: temp = min(N(i – dj), temp) => min(0 , inf) => 0

j = 2: temp = min(N(i – dj), temp) => min(na , 0) => 0

j = 3: temp = min(N(i – dj), temp) => min(na , 0) => 0

i = 2: N(i) = temp +1 => 2 coins

j = 1: temp = min(N(i – dj), temp) => min(N(1) , inf) => 1

j = 2: temp = min(N(i – dj), temp) => min(na , 1) => 1

j = 3: temp = min(N(i – dj), temp) => min(na , 1) => 1

```
i = 3: N(i) = temp +1 => 1 coin
    j = 1: temp = min(N(i – dj), temp) => min(N(2) , inf) => 2
    j = 2: temp = min(N(i – dj), temp) => min(N(0) , 2) => 0
    j = 3: temp = min(N(i – dj), temp) => min(na , 0) => 0

i = 4: N(i) = temp +1 => 2 coins
    j = 1: temp = min(N(i – dj), temp) => min(N(3) , inf) => 1
    j = 2: temp = min(N(i – dj), temp) => min(N(1) , 1) => 1
    j = 3: temp = min(N(i – dj), temp) => min(na , 1) => 1

i = 5: N(i) = temp +1 => 1 coin
    j = 1: temp = min(N(i – dj), temp) => min(N(4) , inf) => 2
    j = 2: temp = min(N(i – dj), temp) => min(N(2) , 2) => 2
    j = 3: temp = min(N(i – dj), temp) => min(N(0) , 2) => 0

i = 6: N(i) = temp +1 => 2 coins
    j = 1: temp = min(N(i – dj), temp) => min(N(5) , inf) => 1
    j = 2: temp = min(N(i – dj), temp) => min(N(3) , 1) => 1
    j = 3: temp = min(N(i – dj), temp) => min(N(1) , 1) => 1

i = 7: N(i) = temp +1 => 3 coins
    j = 1: temp = min(N(i – dj), temp) => min(N(6) , inf) => 2
    j = 2: temp = min(N(i – dj), temp) => min(N(4) , 2) => 2
    j = 3: temp = min(N(i – dj), temp) => min(N(2) , 2) => 2

i = 8: N(i) = temp +1 => 2 coins
```

j = 1: temp = min(N(i – dj), temp) => min(N(7) , inf) => 3

j = 2: temp = min(N(i – dj), temp) => min(N(5) , 3) => 1

j = 3: temp = min(N(i – dj), temp) => min(N(3) , 1) => 1

i = 9: N(i) = temp +1 => 3

j = 1: temp = min(N(i – dj), temp) => min(N(8) , inf) => 2

j = 2: temp = min(N(i – dj), temp) => min(N(6) , 2) => 2

j = 3: temp = min(N(i – dj), temp) => min(N(4) , 2) => 2

Answer: it will take 3 coins for 9 cents back

3) Coin-Collecting Problem

a) In words, how is different from book?

With the addition of the inaccessible squares it adds an additional check that would need to be performed to see if the path can be taken in a way.

It also forces (a bit but not entirely) that you need to start at the start and go down in a bottom up approach rather than starting at the end and going backwards and terminating the sequence if no move is possible.

b) Adjust pseudo code to follow new rule

```
AdjustCoinCollect(C[1…n, 1…m])
   F[1,1] = C[1,1]
   For j = 2 to m  //rows
      If C[1,j] is inaccessible //check col
            Break          // don't keep going down the cols
      else
      $F[1,j] = F[1,j-1] + C[1,j]$
For i = 2 to n   //cols
  If C[I, j] is inaccessible // check current
      Break
  else
      $F[i,1] = F[i-1,1] + C[i,1]$
  For j = 2 to m
      If C[i,j+1] is inaccessible // check col
          Break        // don't keep going down the cols
      else
          $F[i,j] = max(F[i-1,j], F[i,j-1] ) + C[i,j]$
Return F[n,m]
```

c) Use to solve example and fill out two grids

🟩	X		o		
🟩 o			X	o	
🟩	🟩 o		X	o	
	🟩	🟩	🟩 o	🟩	🟩 o
X	X	X		o	🟩

🟩	X				
🟩 o			X	o	
🟩	🟩 o		X	o	
	🟩	🟩	🟩 o		o
X	X	X	🟩	🟩 o	🟩

4) Knapsack (bottom-up DP)

W = 6, { 3:25 , 2:20 , 1:15 , 4:40 , 5:50 }

F (i-1,j)   if j-wi < 0

Max(f(i-1,j), vi + f(i-1, j-wi).   If j-wi > 0

	j = 0	1	2	3	4	5	6
i = 0	0	0	0	0	0	0	0
1	0	0	0	25	25	25	25
2	0	0	20	20	20	45	45
3	0	15	20	35	35	45	60
4	0	15	20	20	40	55	60
5	0	15	20	20	40	55	65

5) Optimal Binary Search Trees

C(i,j) = min for k of c(i,k-1) + c(k+1,j) + sum + Ps

Main:

0	.1	.4	1.1	1.7
	0	.2	.8	1.4
		0	.4	1.0
			0	.3
				0

Example math:

C(2,3) = min (k=2 or k=3) + sum Ps

K=2 => c(2,1) + c(3,3) + [.2+.4] => 0 + .4 + .6 = 1.0

K=3 => c(2,2) + c(4,3) + [.2+.4] => .2 + 0 +.6 = .8

Min is k = 3 so put in root table

Root:

0	1	2	3	3
	0	2	3	3
		0	3	3
			0	4
				0

6) Warshall’s algo

Start

0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

R1

0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

R2

0	1	1	0
0	0	1	0
0	0	0	1
0	0	0	0

R3

0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

R4 = Transitive Closure

0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

7) Floyd's algo

Start Direction Route

0	2	inf	1	8	X	A	B	C	D	E
6	0	3	2	Inf	X	A	B	C	D	E
Inf	Inf	0	4	Inf	X	A	B	C	D	E
Inf	Inf	2	0	3	X	A	B	C	D	E
3	Inf	inf	Inf	0	x	A	B	C	D	E

Iter 1 Direction Route

0	2	Inf	1	8	X	A	B	C	D	E
6	0	3	2	14	X	A	B	C	D	A
Inf	Inf	0	4	inf	X	A	B	C	D	E
Inf	Inf	2	0	3	X	A	B	C	D	E
3	5	inf	4	0	x	A	A	C	A	E

Iter 2 Direction Route

0	2	5	1	8	X	A	B	B	D	E
6	0	3	2	14	X	A	B	C	D	A
inf	inf	0	4	inf	X	A	B	C	D	E
inf	inf	2	0	3	X	A	B	C	D	E
3	5	8	4	0	x	A	A	B	A	E

Iter 3 Direction Route

0	2	5	1	8	X	A	B	B	D	E
6	0	3	2	14	X	A	B	C	D	A
inf	inf	0	4	inf	X	A	B	C	D	E
inf	inf	2	0	3	X	A	B	C	D	E
3	5	8	4	0	x	A	A	B	A	E

Iter 4 Direction Route

0	2	3	1	4	X	A	B	D	D	D
6	0	3	2	5	X	A	B	C	D	D
inf	inf	0	4	7	X	A	B	C	D	D
Inf	Inf	2	0	3	X	A	B	C	D	E
3	5	6	4	0	x	A	B	D	A	E

Final iter Direction Route

0	2	3	1	4	X	A	B	D	D	D
6	0	3	2	5	X	A	B	C	D	D
10	12	0	4	7	X	E	E	C	D	D
6	8	2	0	3	X	E	E	C	D	E
3	5	6	4	0	x	A	B	D	A	E

8) Edit Distance Problem

a) Pseudo code

b) Best/Worst cases

c) Build or explain in detail