Maciej Medyk
Z-15197421

Assignment 3

Posted on April 5, due on April 19

- Maximum total is 40 points. Each problem is worth 10 points.

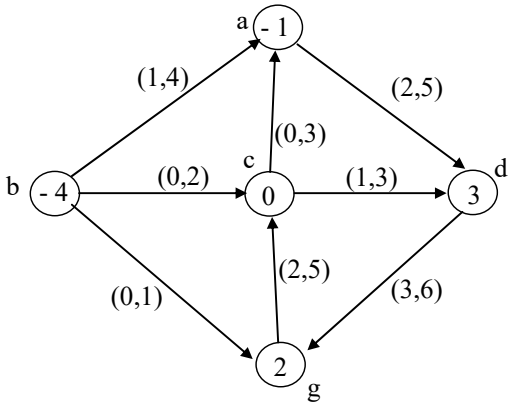
Problem 1.

We solved in class the "Weighted Interval Scheduling" problem using dynamic programming (see PPT presentation on Dynamic Programming). Let us assume that the n requests have start times s[1..n] and finish times f[1..n]. Assume that $f[1] \leq f[2] \leq f[3] \leq \ldots \leq f[n]$.

Use pseudocode to write an algorithm which computes the values p(i) for each request i. The output of your algorithm is an array p[1..n] containing p(i) value for each request i. What is the RT of your algorithm, as a function of n?

Problem 2. (Circulation with demands and lower bounds)

Find whether there is a feasible circulation for the flow-network below. Show your work for full credit, similar to the example done in class.



Problem 3. Greedy Algorithms

Let us consider a system of coins with n denominations $\{c_1, c_2, \ldots, c_n\}$ such that $c_i \geq 2c_{i-1}$ for i = 2, 3, .., n. Does the greedy algorithm optimally solve the change-making problem for this system of coins? Either prove that greedy always yields an optimal solution or give an example for which greedy does not compute the optimal solution.

(*continued next page*)

Problem 4 (Dynamic Programming)

What is an optimal alignment for MOAT and BOAST ?

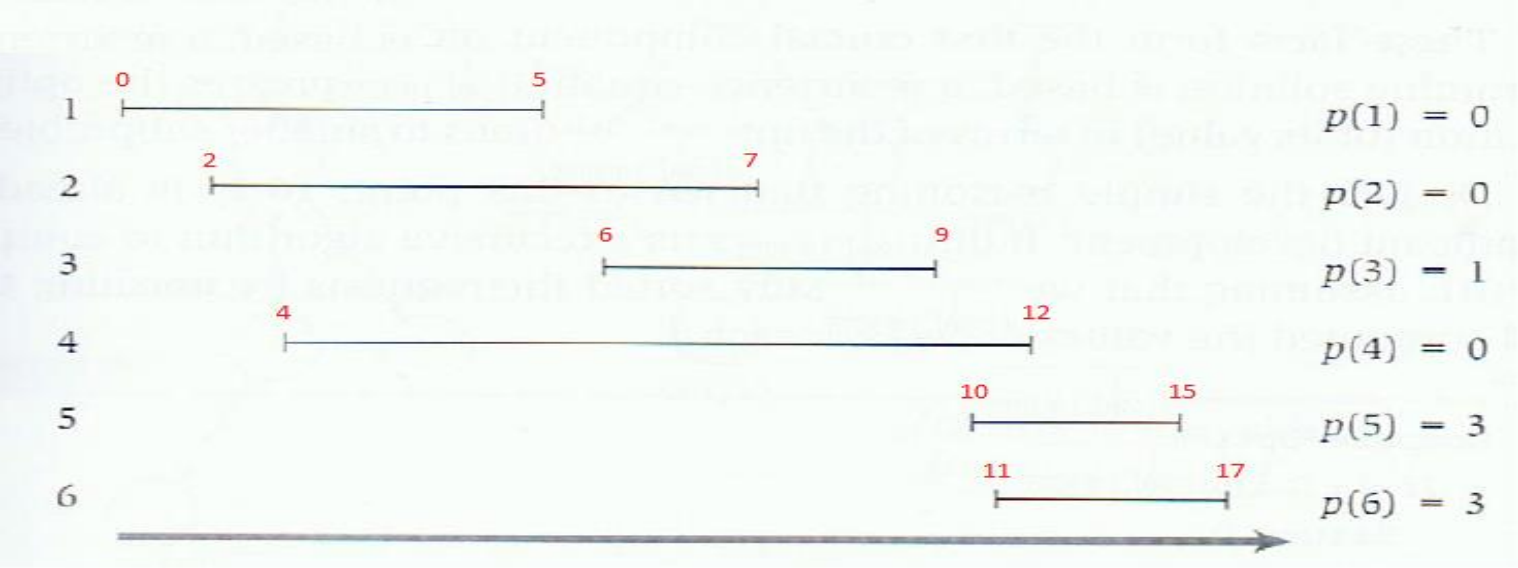
Assume that $\delta = 3$ and consider the following matching/mismatching costs:

	A	B	M	O	S	T
A	0	1	2	3	2	3
B		0	4	3	1	4
M			0	5	1	3
O				0	4	3
S					0	2
T						0

Fill out the table A and write the optimal alignment obtained.

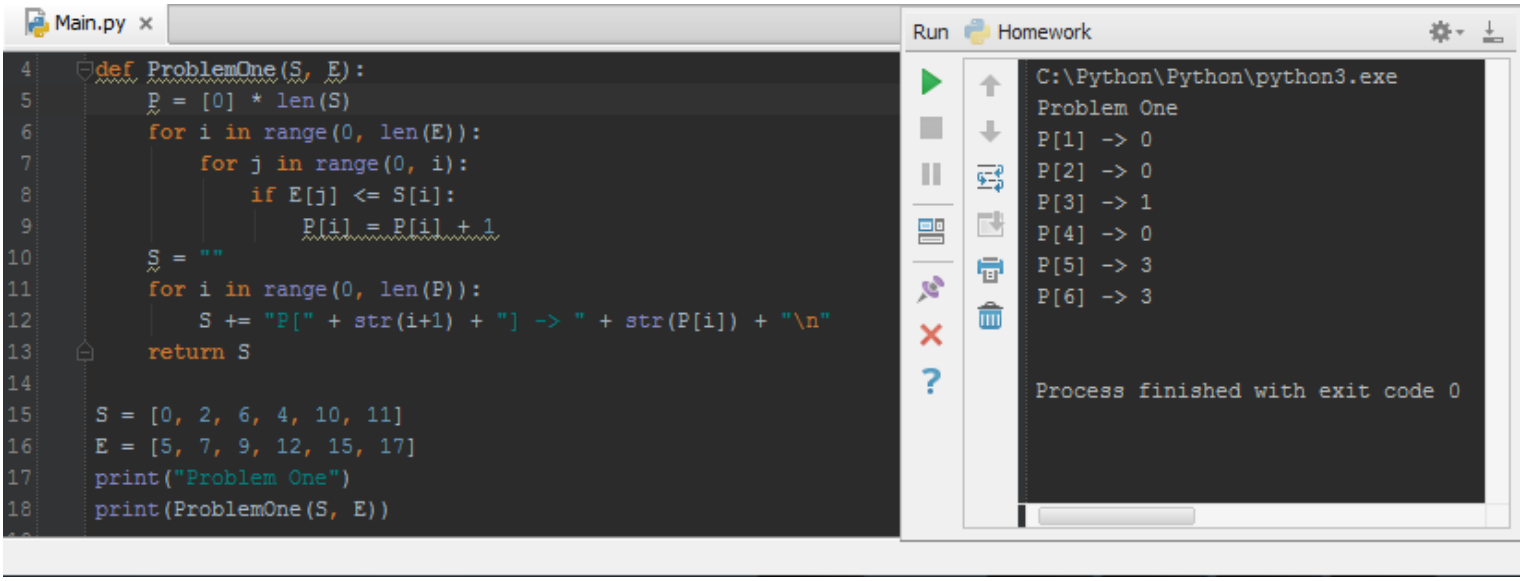
Maciej Medyk – COT6405 – Homework 3

Problem 1

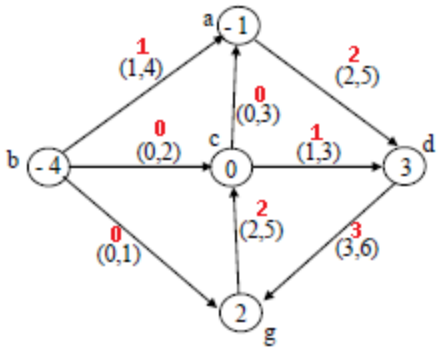


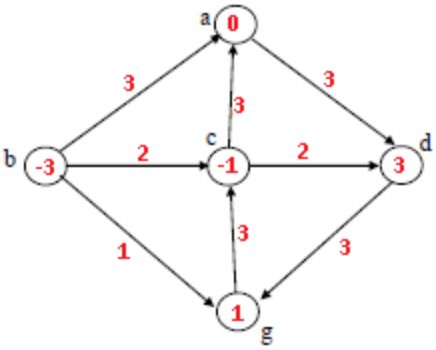

```
S = [0, 2, 6, 4, 10, 11]
E = [5, 7, 9, 12, 15, 17]
P = [0, 0, 0, 0, 0, 0]

for i in range(0, len(E)):              // RT (n)
    for j in range(0, i):               // RT (n^2)
        if E[j] <= S[i]:                // RT (n^2)
            P[i] = P[i] + 1             // RT (n^2)

print(P)
```

Running Time is O (n^2)



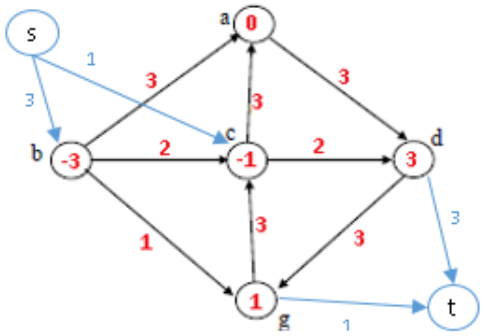
Problem 2

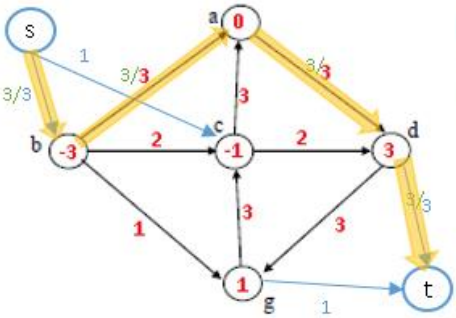


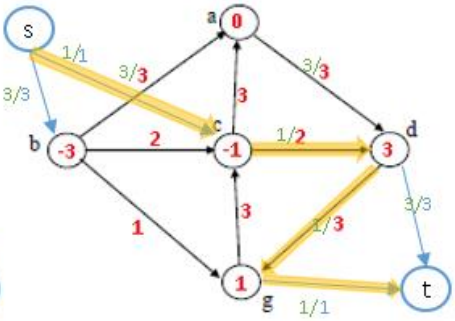


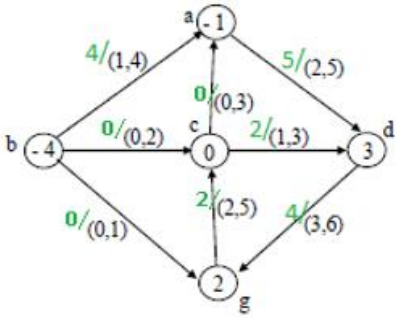

```
Vertex A = (-1 - 1 + 2) =  0
Vertex B = (-4 - 0 + 1) = -3
Vertex C = ( 0 - 2 + 1) = -1
Vertex D = ( 3 - 3 + 3) =  3
Vertex G = ( 2 - 3 + 2) =  1

path = < S* , b , a , d , t* > = 3
path = < S* , c , d , g , t* > = 1
```

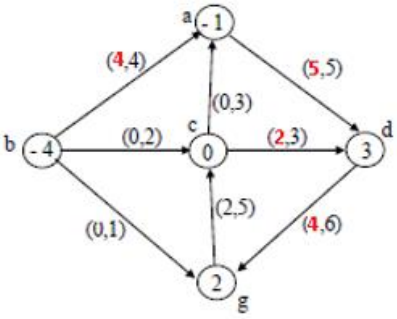
```
Edge BA = 4 - 1 = 3
Edge BC = 2 - 0 = 2
Edge BG = 1 - 0 = 1
Edge AD = 5 - 2 = 3
Edge CA = 3 - 0 = 3
Edge CD = 3 - 1 = 2
Edge DG = 6 - 3 = 3
Edge GC = 5 - 2 = 3
```











Problem 3

Main.py

```
def ProblemThree(D, A):
    T = A
    R = [0] * len(D)
    for i in reversed(range(0, len(D))):
        while A >= D[i]:
            A = A - D[i]
            R[i] = R[i] + 1

    S = "Change for : " + str(T) + '\n'
    S += " D -> C\n"
    for i in reversed(range(0, len(R))):
        if(len(str(D[i])) == 2):
            S += str(D[i]) + " -> " + str(R[i]) + '\n'
        if(len(str(D[i])) == 1):
            S += " " + str(D[i]) + " -> " + str(R[i]) + '\n'
    return S

A = 21
print("\nProblem Three")
D = [1, 2, 4, 8, 16]    # case one all Ci = 2C(i-1)
print("Optimal solution")
print(ProblemThree(D, A))
D = [1, 3, 7, 15, 31]   # case two all Ci + 1 = 2C(i-1)
print("Optimal solution")
print(ProblemThree(D, A))
D = [1, 2, 5, 10, 25]   # united states currency
print("Optimal solution")
print(ProblemThree(D, A))
D = [1, 7, 15, 32, 64]   # one that fails
print("Not optimal solution as 3 coins of 7 will yield 21")
print(ProblemThree(D, A))
```

Run Homework

```
C:\Python\Python\python3.exe

Problem Three
Optimal solution
Change for : 21
 D -> C
16 -> 1
 8 -> 0
 4 -> 1
 2 -> 0
 1 -> 1

Optimal solution
Change for : 21
 D -> C
31 -> 0
15 -> 1
 7 -> 0
 3 -> 2
 1 -> 0

Optimal solution
Change for : 21
 D -> C
25 -> 0
10 -> 2
 5 -> 0
 2 -> 0
 1 -> 1

Not optimal solution as 3 coins of 7 will yield 21
Change for : 21
 D -> C
64 -> 0
32 -> 0
15 -> 1
 7 -> 0
 1 -> 6

Process finished with exit code 0
```

Greedy fails here to be optimal

Optimal solution is 3 coins of 7 yielding 21

In the algorithm I run various cases to analyze and in most cases greedy algorithm does work well; however, fourth case shows that greedy algorithm does not yield optimal solution when denominations are [1, 7, 15, 32, 64] when trying to provide change for 21. According to greedy algorithm we would get 1 coin of 15 and 6 coins of 1 which are total of 7 coins. The optimal solution would have been 3 coins of 7 which would also yield 21 at cost of 3 coins.

Problem 4

4	T	12 ↑	10 ↑	10 ↗	7 ↑	6 ↗	7 ↗
3	A	9 ↑	7 ↗	7 ↑	4 ↗	7 →	10 →
2	O	6 ↑	6 ↗	4 ↗	7 →	10 →	13 ↗
1	M	3 ↑	4 ↗	7 →	8 ↗	10 ↗	13 →
0	_	0	3 →	6 →	9 →	12 →	15 →
		_	B	O	A	S	T
		0	1	2	3	4	5

Letters		Cost	Matrix		↗	→	↑
B	M	4	1	1	4	6	6
B	O	3	1	2	6	9	7
B	A	1	1	3	7	12	9
B	T	4	1	4	13	15	10
O	M	5	2	1	8	7	9
O	O	0	2	2	4	9	10
O	A	3	2	3	9	10	7
O	T	3	2	4	10	13	10
A	M	2	3	1	8	10	12
A	O	3	3	2	10	7	11
A	A	0	3	3	4	10	10
A	T	3	3	4	10	13	7
S	M	1	4	1	10	11	15
S	O	4	4	2	12	10	13
S	A	3	4	3	9	7	13
S	T	2	4	4	6	10	10
T	M	3	5	1	15	13	18
T	O	3	5	2	13	13	16
T	A	3	5	3	13	10	16
T	T	0	5	4	7	9	13

M	O	A	_	T
B	O	A	S	T
4	0	0	3	0
Optimal Alignment Cost = 7				