BLG 336E Final Spring 2018 Name and Student ID:____________________

BLG 336E ANALYSIS OF ALGORITHMS II
FINAL – JUNE 4, 2018, 09:00-11:00 PM (2hours)

Q1 (25pt)	Q2 (25pt)	Q3 (25pt)	Q4 (25pt)	Total (100 pt)

On my honor, I declare that I neither give nor receive any unauthorized help on this exam.

Student Signature:____________________

Write your name on each sheet. Write your answers neatly (in English) in the space provided for them. You must show all your work for credit. Books and notes are closed. Good Luck!

Q1. [25pts]
Assume you are given n different radio frequencies, each may have **pairwise interferences**. You need to find the maximum number of frequencies that **do not** interfere with each other.

Q1a) [10pt] Present the formal representation of this problem considering the existing problem classes? Note that, you do not need to provide a solution.

The radio frequencies → nodes
5
interferences → edges

5
The problem is finding the max. number of nodes without interferences.

Q1b) [15pt] Is this problem NP-complete? Please justify your answer in detail.

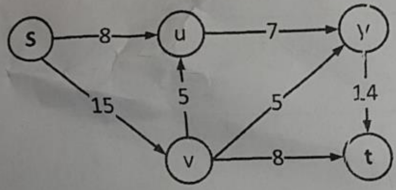
– It's decision version is polynomial. (polynomial certifier)
8
Given a solution it takes polynomial number of steps to determine if it's a solution
(NP)

– It can be polynomially reduced to Max. Independent set problem.
$X \leq_p Y$

7

Q2. [25pts]

Q2a) [15pts] The directed graph below shows a flow network. The capacities are given on each edge. Find the **Max-flow Min-cut s-t** flow on this graph by **presenting all the steps of the algorithm** that finds the **optimal flow value**. Please number each step and show the intermediate structure of the graph.



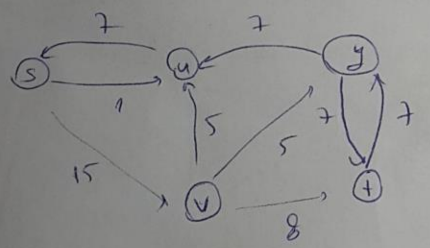
step-by-step flow values -5

residual graph -10

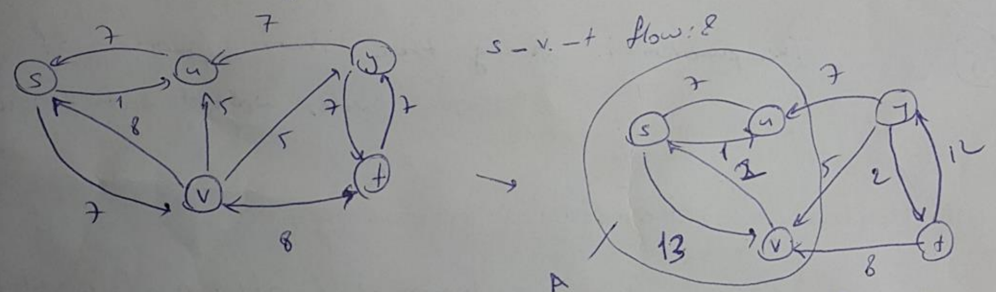
conversation of flow error -10

Residual Graph

Flow: 7 s-u-y-t



s-v.-t flow: 8



A

s-v-y-t flow: 5

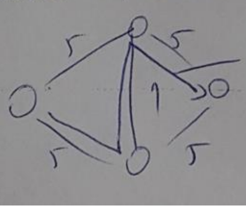
Q2b) [5pts] For the final flow found in (a)

The value of the maximum flow	20 (1)
The nodes in partition A (s∈ A) of the minimum cut (A,B)	s, u, v (3)
The capacity of the cut	20 (1)

Q2c) [5pts] Please explain the effects of selecting a bad augmenting path by this algorithm with an example?

The number of iterations will be higher. (3) (2)

a bad augmenting path example



Q3)[25pt]
Let A be an array of integers of size n, where $A[1]<A[2]<...<A[n]$.

Q3a) [20pts] Write the pseudocode of an **efficient** algorithm (the worst case complexity is better than O(n)) to find an i such that $A[i]=i$ provided that such an i exists. If more than such i exists, finding any one of them is acceptable.

Divide & Conquer approach.

Note that A[i] could be negative, there could be multiple values of the same A[i].

Three cases:

i==A[i]: done

i<A[i]: check if A[A[i]]==A[i] .. e.g.

i:	3	4	5	6	7	8
A[i]	3	3	8	8	8	8

else search in the portion before i.

i>A[i]: check if A[i]>0 & A[A[i]]==A[i] e.g.

i:	3	4	5	6	7
A[i]:	3	3	3	5	7

else search in the portion after i.

```
IFindAi(i, A, mini, maxi){
  if maxi<mini return (false,-1,-1)
  else if maxi==mini && mini<n & mini>,1
     if A[mini]==mini return (true, mini, mini)
  else if i==A[i] return (true, i, i)
  else if i>A[i]
     if A[i]>0 && A[A[i]]==i  return (true, A[i], A[i])
     else return (IfindAi( i+⌊(maxi-i)/2⌋, A, i+1, maxi)
  else if i<A[i]
     if A[A[i]]==A[i] return (true, A[i], A[i])
     else return (IfindAi( i-⌊(mini-i)/2⌋, A, mini, i-1)
}
```

Initial call with: IFindAi(⌊n/2⌋, A, 1, n)

Q3b) [5pts] What is the worst case complexity of your algorithm? O($\log_2 n$)
Give the proof for the complexity.

$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$

$T(n) = c(\log_2 n + 1)$

$= O(\log_2 n)$

		h=
$T(n)$	c	0
$T(\frac{n}{2})$	c	1
$T(\frac{n}{4})$	c	
$T(1)$	c	$\log_2 n$
+	$c(\log_2 n + 1)$	

$n/2^k = 1 \Rightarrow n = 2^k,\ k = \log_2 n$

BLG 336E Final Spring 2018 Name and Student ID: SOLN

Q4)[25pt]

The following items are available at the grocery store:

\# items : K=4
weight limit = W=10

	Milk (M)	Cheese (C)	Egg (E)	Yoghurt (Y)	
Price	2 TL	3 TL	1 TL	6 TL	→ weight, w(i), i=1..K
ProteinValue	1	5	3	2	→ value, v(i), i=1..K

You want to buy a set of unique items with the maximum total protein content. You only have 10TL.

Q4a) [15pts] Write down the pseudocode of a dynamic programming algorithm to solve this problem.

```
Knapsack Problem
M[K,W] // Used for memoization, M[i,j]: value when item i & weight limit j is considered.
M[0,:] = M[:,0] = 0 // First row and col zeros.
Pred[K,W] = [[0,0]] // shows location of predecessor positions in M for backtracking the solution.
for i=1..K
  for j=1..W
    w = weight(i), v = value(i)
    if w <= j // enough capacity?
      if M[i-1, j-w] + v >= M[i-1, j]
        M[i,j] = M[i-1, j-w] + v; Pred(i,j) = [i-1, j-w]
      else M[i,j] = M[i-1, j]; Pred(i,j) = [i-1, j]
    else M[i,j] = M[i-1, j]; Pred(i,j) = (i-1, j)
```

Q4b)[10pts] Give the steps of your algorithm and the final solution:

```
Itemset = {}
i=K, j=L, p=Pred(i,j)
for i=K down to 1
  if p.getJ != j
    Itemset = Itemset ∪ {j}
  p = Pred(p.getI, p.getJ)
Return (M[K,W], Itemset)
```

weight / value		0	1	2	3	4	5	6	7	8	9	10
	{}	0	0	0	0	0	0	0	0	0	0	0
2,1	{M}	0	0	1	1	1	1	1	1	1	1	1
3,5	{M,C}	0	0	1	5	5	6	6	6	6	6	6
1,3	{M,C,E}	0	3	3	5	8	8	9	9	9	9	9
6,2	{M,C,E,Y}	0	3	3	5	8	8	9	9	9	9	10

Buy: Cheese, Egg, Yoghurt Total Cost: 10 Total ProteinValue: 10