Question 5: 4 points (3 + 1) [Bonus Question]

a. An **independent set** in a Graph is defined as an independent set of vertices in a graph, no two of which are adjacent. That is, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two. The size of an independent set is the number of vertices it contains. Assume that independent set problem is in NP.

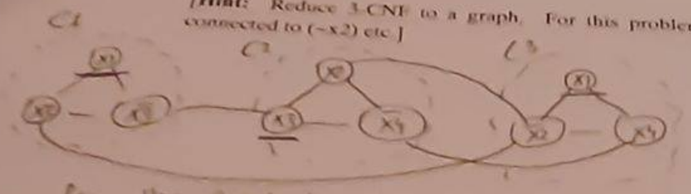
Assume the following 3-CNF which is NP-Complete.

$(x1 \vee x2 \vee \neg x3) \wedge (x2 \vee x3 \vee \neg x4) \wedge (x1 \vee \neg x2 \vee x4)$ where $\neg$ means not.

Show that Independent set in a graph corresponding to the 3-CNF is NP-Complete.

What is the size of the Independent set S (i.e. maximum number of nodes in S that are not adjacent)? Explain your answer.

*[**Hint:** Reduce 3-CNF to a graph. For this problem, you may consider a node (e.g. x2) is connected to (~x2) etc.]*



We have shown that solution S for 3CNF is solution of ISet and solution S for Independent Set is solution of 3CNF as well.

From the graph based on clauses above we can see that max size of independent set is (x_1 of C1, x_2 of C2 and x1 of C3). So, lets prove that these can be solution for 3CNF. $x1=1,\ x3=1$

$= (\underbrace{1 \vee x2 \vee \bar{0}}_{1}) \wedge (\underbrace{x2 \vee 1 \vee \bar{x}4}_{1}) \wedge (\underbrace{1 \vee \bar{x}2 \vee \bar{x}4}_{1}) = 1$ It is true

Also this x_1 of C1, x_2 of C2 and x1 of C3 are member of independent set. This is clear from the graph above. So. Solution for 3CNF is solution of Independent set aswell! - Max size of independent set for 3CNF is 3.

b. Can you use Bellman-Ford shortest path algorithm on undirected graph? Explain your answer. If your answer is yes, will you be able to deal with negative weight?

BF shortest path algorithm can be used on undirected graph because undirected graph is directed graph with 2 ways on each edge.

Generally BF can be used on graph with negative weight but cannot be used on graph with negative cycle. Negative weight on undirected graph means negative cycle. So. all weight has to be positive on undirected graph when we use BF.

(a)

Suppose that A and B , and that A ≤p B. Circle one answer for each statement below.

A reduced to B.

Also, briefly justify your answer.

1. If B is NP-Hard then A is NP-Hard True False

Justify:

ALL NP complete, NP, P problems can be reduced to NP hard. So not able to exactly define A.

2. If A is EXP-COMPLETE then B is EXP-COMPLETE True False

Justify:

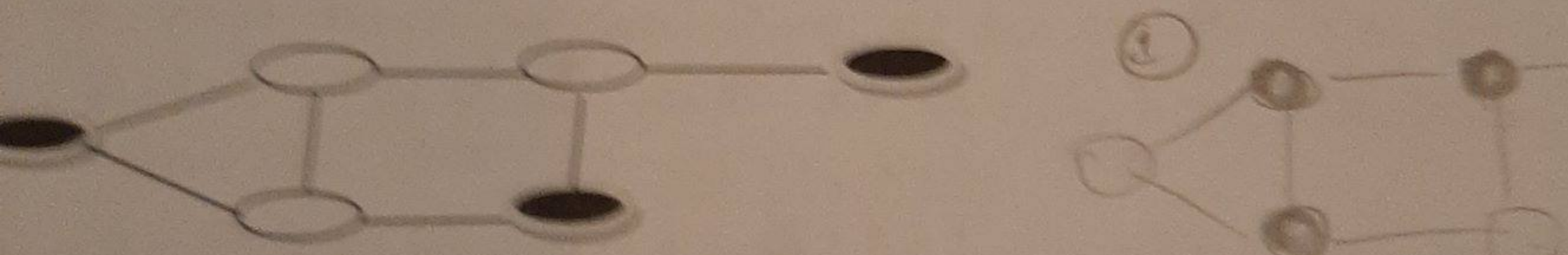
is EXP-complete means all

3. Does A ≤p B implies B ≤p A in general? True False

Justify:

For example NP problem A reduced to NP complete B. B Cannot be reduced to

(b) In graph theory, a **dominating set** for a graph G = (V, E) is a subset D of V such that every vertex not in D is adjacent to at least one member of D. An example of a Dominating set is the set of the 3 nodes shown as bold for the graph G shown below:



Show 2 other Dominating sets for the same graph.

oes Power Set problem belong to NP-Complete problem? Explain your answers.

Power Set doesn't belong to NP-Complete. Because, it can b
Exp time using the Dynamic programming and it can be
O time

Question 3: Contd.

b. Consider the following graph. Start at node A. Edges are processed in the following order (B,E), (A,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D). Find all shortest path distances to all nodes using Bellman-Ford algorithm. Show all steps clearly.

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞

Since, number of V is 5, it will iterate 4 times (5-1) and so on.
We get following distances when all edges are processed first time:
- distance of BE, BB, BD and A, B are passed
- result of AC
- result of DC, BC and ED

Overall, distance of 3 edges

Second iteration

A	B	C	D	E
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1

Result after second iteration

Third iteration and Fourth iteration is same as below

A	B	C	D	E
0	-1	2	-2	1

```
Bell Man Ford (G, w, s)
Initialize Single-Source (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        Relax (u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return false
```

So result is. Distance from A to all other nodes

	Distance	Prev Vertex
A	0	0
B	-1	A
C	2	B
D	-2	E
E	1	B

Question 4: 11 points (6 + 3 + 2)

(a)

Suppose that A and B , and that A ≤p B. Circle one answer for each statement below.

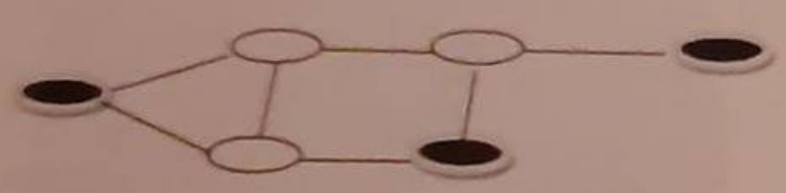
Also, briefly justify your answer.

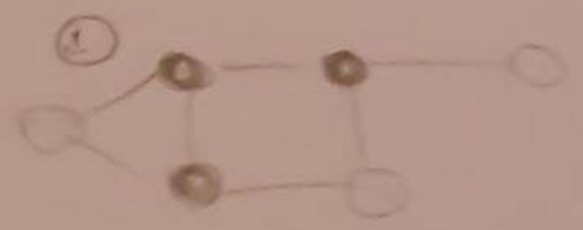
1. **If B is NP-Hard then A is NP-Hard** True **False**
Justify -

2. **If A is EXP-COMPLTE then B is EXP-COMPLTE** **True** False
Justify –

3. **Does A ≤p B implies B ≤p A in general?** True **False**
Justify -

(b) In graph theory, a **dominating set** for a graph G = (V, E) is a subset D of V such that every vertex not in D is adjacent to at least one member of D. An example of a Dominating set is the **set of the 3 nodes shown as bold** for the graph G shown below.





Show 2 other Dominating sets for the same graph.

(c) Does Power Set problem belong to NP-Complete problem? Explain your answers.

No, Power Set doesn't belong to NP-Complete. Because, it can be solved in Exp time using the Dynamic programming and it can be verified at exp time.

s: 12 points (8 + 4)

The shortest path algorithm using Dijkstra is shown in the pseudo code below. Show that the running time is $O(|E| \log |V|)$. Assume Adjacency List to represent the graph. **Explain your answer clearly by considering running time for each line and any assumptions used.** Assume implementation using a Heap.

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

S

→ O(V)
→ O(1)
→ O(lg V) - due to heapification.
→ V times
→ V · lg V → Heapification required
→ V(ΔE · lg V) = E · lg V - explain below.

1. Every time we enter line 7, we get small number of nodes (verteces) as neighbor.
2. Hence it is accounts for ΔE.
3. Hence it takes $|V| \cdot (\Delta E \cdot \lg V)$

Since total running time is $O(|V| \cdot \lg V + |E| \cdot \lg V) = O((|V| + |E|) \cdot \lg V)$

Hence number of edges is higher than number of verteces $O(E \cdot \lg V)$

total running time of Dijkstra shortest path algorithm.