# Review for Final – E.K 8/10/21

## Data Structures

1. **Give one advantage of a Linked List, as compared to an array.**

Ans. Linked List has much faster Insert and Remove operations than an Array – O(1) for Linked List and O(n) for Arrays.

Another one – Linked List allows more flexible memory management than an Array. It does not require large contiguous memory blocks, and needs no preallocation.

**2. Data Structures – Hashing 1**

Insert the following sequence of numbers 23, 46, 12, 21, 75, 5, 3 into a hash table of size 9 using h(x) = x%9 as a hash function. % means Mod. Use Chaining with Linked List to avoid collision.

Ans.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| | 46 | | 12,21,<br>75,3 | | 23,5 | | | |

3.

Suppose that we store $n$ keys in a hash table of size $m = n^2$ using a hash function $h$ randomly chosen from a universal class of hash functions. Then, the probability is less than $1/2$ that there are any collisions.

**3. Data Structures – BST, Red-Back tree and Heaps**

- **Focus more on Red-Black Tree and Heaps**

**Graphs – Basics, DFS, BFS, MST and Shortest paths**

- **Focus more on DFS / BFS and Shortest paths**
- Sample questions / problems as covered in class and in labs
- More problems from these materials

## Hard Problems

4. Suppose A and B, and that A -> B (A is Reduceable to B in Poly time). Circle one answer for each statement below.

If A is NP-Complete and B is in NP then B is NP-Complete.

True False

If B is P then A is P.

True False

If B is NP-hard then A is NP-hard.

True False

**Reduction -1**

5. 3-CNF to Clique - To be done in the Class

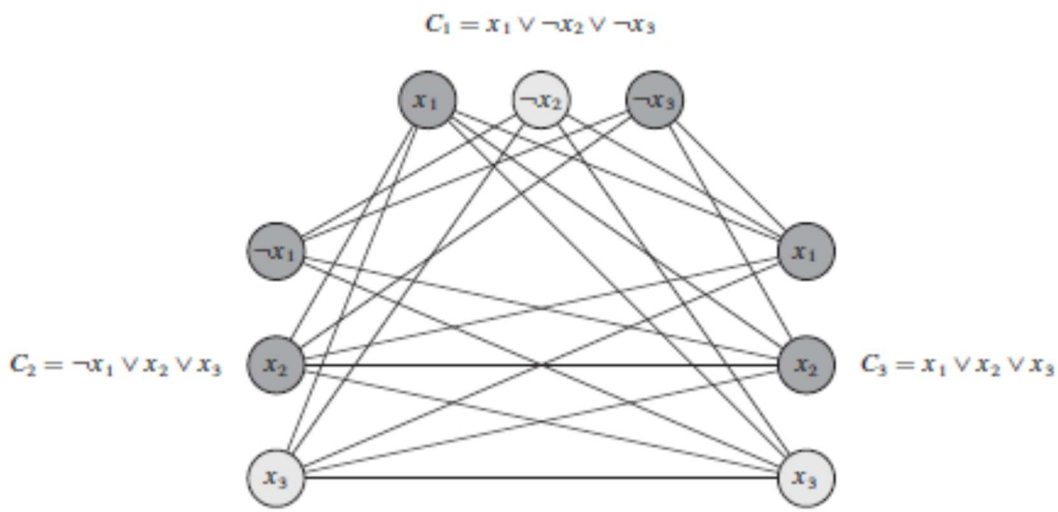



**Figure 34.14** The graph $G$ derived from the 3-CNF formula $\phi = C_1 \wedge C_2 \wedge C_3$, where $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2 = (\neg x_1 \vee x_2 \vee x_3)$, and $C_3 = (x_1 \vee x_2 \vee x_3)$, in reducing 3-CNF-SAT to CLIQUE. A satisfying assignment of the formula has $x_2 = 0$, $x_3 = 1$, and $x_1$ either 0 or 1. This assignment satisfies $C_1$ with $\neg x_2$, and it satisfies $C_2$ and $C_3$ with $x_3$, corresponding to the clique with lightly shaded vertices.

Function using 3-CNF is

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Conditions to form the graph:

- $v_i^r$ and $v_j^s$ are in different triples, that is, $r \neq s$, and
- their corresponding literals are ***consistent***, that is, $l_i^r$ is not the negation of $l_j^s$.

Proof:

We must show that this transformation of $\phi$ into $G$ is a reduction. First, suppose that $\phi$ has a satisfying assignment. Then each clause $C_r$ contains at least one literal $l_i^r$ that is assigned 1, and each such literal corresponds to a vertex $v_i^r$. Picking one such "true" literal from each clause yields a set $V'$ of $k$ vertices. We claim that $V'$ is a clique. For any two vertices $v_i^r, v_j^s \in V'$, where $r \neq s$, both corresponding literals $l_i^r$ and $l_j^s$ map to 1 by the given satisfying assignment, and thus the literals

cannot be complements. Thus, by the construction of $G$, the edge $(v_i^r, v_j^s)$ belongs to $E$.

Conversely, suppose that $G$ has a clique $V'$ of size $k$. No edges in $G$ connect vertices in the same triple, and so $V'$ contains exactly one vertex per triple. We can assign 1 to each literal $l_i^r$ such that $v_i^r \in V'$ without fear of assigning 1 to both a literal and its complement, since $G$ contains no edges between inconsistent literals. Each clause is satisfied, and so $\phi$ is satisfied. (Any variables that do not correspond to a vertex in the clique may be set arbitrarily.) ■

**Reduction -2**

6. Clique to Vertex Cover - To be done in the Class

(Next Page)

# Clique →

# Vertex Cover

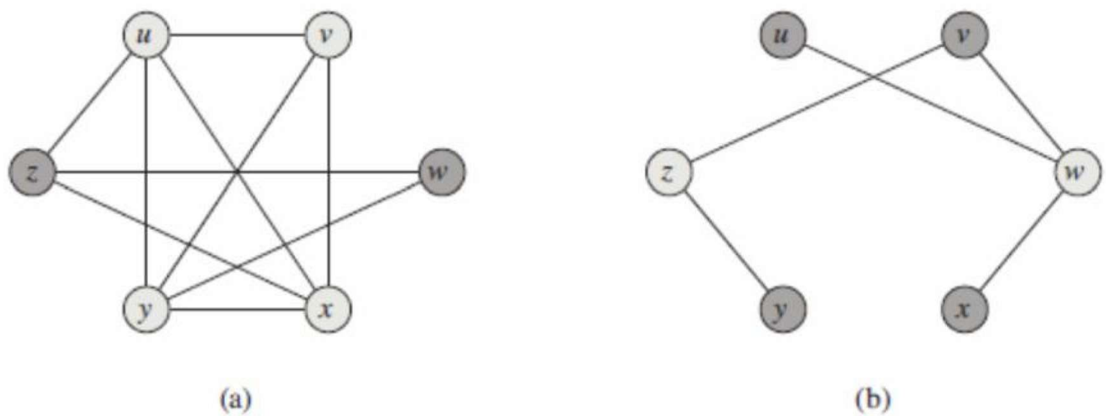



**Figure 34.15** Reducing CLIQUE to VERTEX-COVER. **(a)** An undirected graph $G = (V, E)$ with clique $V' = \{u, v, x, y\}$. **(b)** The graph $\overline{G}$ produced by the reduction algorithm that has vertex cover $V - V' = \{w, z\}$.

# Clique → Vertex Cover: Problem

Suppose that $G$ has a clique $V' \subseteq V$ with $|V'| = k$. We claim that $V - V'$ is a vertex cover in $\overline{G}$. Let $(u, v)$ be any edge in $\overline{E}$. Then, $(u, v) \notin E$, which implies that at least one of $u$ or $v$ does not belong to $V'$, since every pair of vertices in $V'$ is connected by an edge of $E$. Equivalently, at least one of $u$ or $v$ is in $V - V'$, which means that edge $(u, v)$ is covered by $V - V'$. Since $(u, v)$ was chosen arbitrarily from $\overline{E}$, every edge of $\overline{E}$ is covered by a vertex in $V - V'$. Hence, the set $V - V'$, which has size $|V| - k$, forms a vertex cover for $\overline{G}$.

# Clique → Vertex Cover: Solution

Suppose that $G$ has a clique $V' \subseteq V$ with $|V'| = k$. We claim that $V - V'$ is a vertex cover in $\overline{G}$. Let $(u, v)$ be any edge in $\overline{E}$. Then, $(u, v) \notin E$, which implies that at least one of $u$ or $v$ does not belong to $V'$, since every pair of vertices in $V'$ is connected by an edge of $E$. Equivalently, at least one of $u$ or $v$ is in $V - V'$, which means that edge $(u, v)$ is covered by $V - V'$. Since $(u, v)$ was chosen arbitrarily from $\overline{E}$, every edge of $\overline{E}$ is covered by a vertex in $V - V'$. Hence, the set $V - V'$, which has size $|V| - k$, forms a vertex cover for $\overline{G}$.

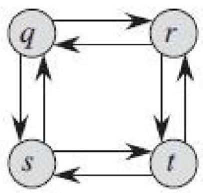
## Clique → Vertex Cover: Soln. - Converse

Conversely, suppose that $\overline{G}$ has a vertex cover $V' \subseteq V$, where $|V'| = |V| - k$. Then, for all $u, v \in V$, if $(u, v) \in \overline{E}$, then $u \in V'$ or $v \in V'$ or both. The contrapositive of this implication is that for all $u, v \in V$, if $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$. In other words, $V - V'$ is a clique, and it has size $|V| - |V'| = k$. ∎

## Dynamic Programming

7.

a. Consider the unweighted directed graph shown below. Does this graph has optimal substructure to find the longest simple path, say between q and r or q and t? Explain your answer. Can you use Dynamic Programming for this problem. Explain why or why not.




Ans.

No, this graph does not have optimal substructure. Consider path $q \to r \to t$, which is a longest simple path from $q$ to $t$. Is $q \to r$ a longest simple path from $q$ to $r$? No, for the path $q \to s \to t \to r$ is a simple path that is longer. Is $r \to t$ a longest simple path from $r$ to $t$? No again, for the path $r \to q \to s \to t$ is a simple path that is longer.

Since the graph does not have optimal substructure, we cannot use Dynamic Programming for this problem.