PRINT Name

- Grades depend on neatness and clarity.
- Write your answers with enough detail about your approach, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.
- Write your answers in the spaces provided. If needed, attach other pages.
- The grades would be out of 16. Four problems would be selected and everyone’s grade would be based only on those problems. You will also get 4 bonus points for trying to solve all problems.

1. [Prob 5.20, pg. 116] Let $x_1, x_2, ..., x_n$ be a set of integers, and let $S = \sum_{i=1}^{n} x_i$. Design an algorithm to partition the set into two subsets of equal sum, or determine that it is impossible to do so. The algorithm should run in time $O(nS)$.

   For a solution to exist, the sum of each subset must be $S/2$, and thus $S$ must be even. Also, a solution to the subset-sum problem for $S/2$ gives a solution to this problem. We can solve subset-sum using dynamic programming in $O(nS)$ time.
2. [Prob 7.2, pg. 248] Let $G = (V, E)$ be a connected, undirected graph, and let $T$ be a DFS tree of $G$ rooted at $v$.
   a. Let $H$ be an arbitrary induced subgraph of $G$. Show that the intersection of $H$ and $T$ is not necessarily a spanning tree of $H$.

   ![Graphs](image)

   b. Let $R$ be a subtree of $T$, and let $S$ be the subgraph of $G$ induced by the vertices in $R$. Prove that $R$ could be a DFS tree of $S$.

   Let $u$, be the vertex in $R$ that is closest to root $v$ in $T$, and take that as the root of $R$. Any edge in $S$ not in $R$, is an edge in $G$ not in $T$ and therefore a back-edge (connects a vertex to an ancestor in $T$). By construction, this is back-edge in $R$ as well. Therefore subtree $R$ rooted at $u$, is a DFS of $S$. 

3. Give an example that the Dijkstra algorithm is not working properly with negative edge weights. The graph should not contain a negative cycle (a negative cycle is a cycle which the sum of the weights of its edges is negative). Briefly explain why Dijkstra does not work in your example.

Starting from the leftmost vertex, the distance of the rightmost vertex in Dijkstra would be 2. However, the length of the shortest path to the rightmost vertex is actually 1.
4. Recall that given a graph $G$, the Bellman-Ford algorithm runs for $n - 1$ iterations, where $n$ is the number of vertices in $G$. Prove that $G$ contains a negative cycle, if and only if at least one edge gets relaxed if we run the algorithm for one extra iteration.

You can prove the problem by showing that this lemma holds (you can use induction):

**Lemma.** After $i$ repetitions of for cycle:

- If “$u$.distance” is not infinity, it is equal to the length of some path from $s$ to $u$;
- If there is a path from $s$ to $u$ with at most $i$ edges, then “$u$.distance” is at most the length of the shortest path from $s$ to $u$ with at most $i$ edges.

5. Given a string $s$ of length $n$ from a 4-letter alphabet (A,C,G,T), and integer $k$, design an algorithm to determine if there exists a substring of arbitrary length $m$, $k < m \leq n$, that occurs more than once in $s$ (i.e., is there a repeated substring of length greater than $k$). The algorithm should run in $O((n - k)^2)$. HINT: construct a directed graph where each vertex corresponds to those substrings of length $k$ occurring in $s$, with edges $(u,v)$ if the $k-1$ suffix of substring $u$ matches the $k-1$ prefix of $v$.

This type of problem is pervasive in DNA sequence analysis where sequence repeats are of biological interest.

Construct the directed graph of substrings as described in the hint. A repeat of length more than $k$ exists if the path describing $s$ in this graph contains a cycle. There are at most $n-k$ vertices in this graph, and thus $O((n - k)^2)$ edges at worst. A depth-first or breadth-first traversal can be done to determine if the path contains a cycle.
6. A 2SAT instance is an instance of Satisfiability in which each clause contains two literals. Design a polynomial algorithm which decides whether a given 2SAT is satisfiable. (Hint: Make a graph based on the input)

You can find the answer in Wikipedia or in here: http://cgm.cs.mcgill.ca/~breed/308362B/2sat.ps
7. Consider the weighted graph below which is represented by its adjacency matrix. Run the Dijkstra algorithm starting from vertex 1. Write the vertices in the order which they are marked. Now run the Prim’s algorithm starting from vertex 1. Again write the vertices in the order which they are marked.

\[
\begin{bmatrix}
0 & 5 & 0 & 0 & 8 & 0 & 6 \\
5 & 0 & 0 & 7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 3 & \ \\
0 & 7 & 0 & 0 & 2 & 0 & 0 \\
8 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 4 \\
6 & 0 & 3 & 0 & 0 & 4 & 0
\end{bmatrix}
\]

**Prim:** 1, 2, 7, 3, 6, 4, 5
**Dijkstra:** 1, 2, 7, 5, 3, 4, 6
8. A Hamiltonian path in graph $G=(V,E)$ is a simple path that includes every vertex in $V$. Design an algorithm that runs in linear time, to determine if a Hamiltonian path exists in a given directed acyclic graph.

Perform a topological sort of the DAG, then check if successive vertices in the sort are connected in the graph. If so, the topological sort gives a Hamiltonian path. On the other hand, if there is a Hamiltonian path, then the path gives a topological sort of the DAG.
9. [Prob. 7.61, pg. 256]. Let $G=(V,E)$ be a connected, undirected graph, and let $T$ be a minimum cost spanning tree of $G$. Suppose the cost of one edge $e$ in $G$ is changed. Discuss the conditions in which $T$ is no longer a MCST of $G$. Design an efficient algorithm to either find a new MCST or to determine that $T$ is still an MCST. ($e$ may or may not belong to $T$).

Suppose $e=(u,v)$ is in $T$, then $T$ may no longer be an MCST if the cost of $e$ becomes larger than the cost of an edge not in $T$ that is in any path between $u$ and $v$. To find an MCST $e$ is removed such that now $T$ is disconnected into trees $T_1$ and $T_2$ (you can show that these are MCSTs for the corresponding induced subgraphs), then the minimum cost edge joining a vertex in $T_1$ to a vertex in $T_2$ is added to make the MCST.

Suppose $e=(u,v)$ is not in $T$, then $T$ may not be an MCST if the cost of $e$ becomes smaller than the largest cost in the path between $u$ and $v$ in $T$. In this case, this edge is removed and new edge is added to $T$ as before.
10. Consider an $N \times N$ (N even) board of alternating black and white squares (like a chess board). Prove that if one removes an arbitrary black square and an arbitrary white square, the rest of the board can be covered by dominoes (of size 2x1). (State the problem as a graph, and think how finding a Hamiltonian paths can be used prove this).

Construct a bipartite graph for the complete board where each square is a vertex and there is an edge between squares that share an edge. For a 4x4 board, the graph looks like this:

![Bipartite Graph Image]

Laying a domino on the board corresponds to traversing an edge, as does laying two dominos next to each other. A Hamiltonian path then gives a way of covering the entire board with dominoes. The bipartite graph obviously has a Hamiltonian path. It is easy to show that after removing one node from each set, there is still a Hamiltonian path.
11. [Prob. 11.32, pg. 373] Let $G=(V,E)$ be an undirected graph, such that each vertex is associated with some task. Two vertices are connected if the corresponding tasks cannot be performed at the same time (e.g., they need the same resource). This is the only limit on concurrency. Any set of tasks such that no two of them are connected can be performed in one step. Prove that the following problem is NP-complete: Given a graph $G=(V,E)$, and an integer parameter $k$, determine whether all corresponding tasks can be performed in at most $k$ steps.

Note that tasks can be performed in at most $k$ steps iff the graph does not contain a clique of size $k+1$ or larger (each task in the clique needs to be done one at a time and thus takes a total of $k+1$ steps). Therefore, the clique problem can be reduced to this problem in polynomial time, which makes this problem NP-complete.