MAJOR EXAM (COL 351)

Name **Entry No.**

I do hereby undertake that as a student at IIT Delhi: (1) I will not give or receive aid in examinations, and (2) I will do my share and take an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honour Code.

Signature:

There are 6 questions. Each algorithm must be accompanied by justification about its correctness and efficiency. You can use the fact that the following problems are NP-complete: 3-Satisfiability, Clique, Vertex Cover, Independent Set, Subset Sum, Hamiltonian Cycle.

1. Consider the following decision problem: given a **connected** undirected graph G on n vertices, does G have a cycle containing at least $n-100$ vertices? Prove that this problem is NP-complete. (**6 marks**)

   Solution: First observe that the problem is in NP. Indeed, a verifier takes as input a graph G and a path P and checks if G is connected, P is a path in G and contains at least $n-100$ vertices. This can clearly be done in polynomial time.

   Now we reduce the Hamiltonian cycle problem to this problem. Let G be an instance of the Hamiltonian cycle problem. We can assume that G is connected, otherwise the answer to the Hamiltonian cycle problem is trivially no. We now construct a new graph G' as follows: we take G and add 100 new vertices to G. Note that the instance G' has to be connected. Let $v_1, \ldots, v_{100}$ be these 100 vertices. Let v_0 be any vertex in G. Then we add edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{99}, v_{100})$ to G'. Now, we claim that G has a Hamiltonian cycle iff G' has a cycle containing at least $n'-100$ vertices, where n' is the number of vertices in G'.

   Suppose G has a Hamiltonian cycle C. Then C is also a cycle in G' and has $n = n'-100$ vertices. Let us check the converse. Suppose G' has a cycle containing $n = n'-100$ vertices. We first claim that such a cycle cannot contain any of the new vertices. Indeed, if it contains a new vertex, it has to contain the entire path P from v_0 to v_{100}. But v_{100} cannot be part of a cycle as it has degree 1. Thus, this cycle can only contain vertices of G and hence, must be a Hamiltonian cycle in G.

2. You are given a directed graph G on n vertices and m edges with edge lengths c_e, which could be positive or negative real. You are also given two vertices s and t. Further, each edge of the graph is assigned a colour – it is coloured either Blue or Red. Assume you are given this information in the adjacency list representation of the graph. Give an $O(mn)$ time algorithm for finding the length of the shortest **walk** from s to t with the condition that the walk should not have two consecutive Blue edges (for example, it can look like RBRBRRRRB, but cannot look like BRRRRBBRR – here B denotes Blue and R denotes Red) and it should have **at most** n edges. Recall that a walk from s to t is a sequence of vertices $s = v_1, v_2, \ldots, v_k = t$, such that each (v_i, v_{i+1}) is an edge. Note that unlike a path, a walk can visit the same vertex again. The algorithm just outputs a number (in case there is no such walk, it outputs infinity). (**7 marks**)

Solution: We use the same idea as in the Bellman Ford algorithm. In Bellman Ford, we maintain tables $D_k[v]$ denoting the shortest walk length from v to t using at most k edges. Here we have 2 tables: $D_k^0[v], D_k^1[v]$, where $D_k^r[v]$ denotes the shortest valid walk length from v to t using at most k edges and that starts with exactly r Blue edges. A walk is valid if it does not contain more than one consecutive blue edges. Now, we can have a similar recurrence for $D_k^r[v]$: let $N(u)$ denote the out-neighbours of a vertex u. Let $N_B(u)$ be the out-neighbours w for which the edge (u, w) is Blue and $N_R(u)$ be the ones for which the out-going edge is Red. So, now the recurrence is as follows: Consider $D_k^0[v]$: it has to start with a red edge. Hence,

$$D_k^0[v] = \min(D_{k-1}^0[v], \min_{w \in N_R(v)} (c_{(v,w)} + \min(D_{k-1}^0[w], D_{k-1}^1[w]))).$$

Now we consider $D_k^1[v]$. It must start with a Blue edge, but then the following path can should not start with Blue edge. Therefore,

$$D_k^1[v] = \min(D_{k-1}^1[v], \min_{w \in N_B(v)} (c_{(v,w)} + D_{k-1}^0[w])).$$

Thus, the final algorithm is:

```
Initialize D_0^0[v] = 0 if v=t, infinity otherwise.
Initialize D_1^1[v]= infinity
For k=1 to n
     For v = 1 to n
         Update D_k^r[v],  as above.
```

3. The edge orientation problem is defined as follows: given an undirected graph G and a number k, we would like to assign a direction to each edge in G such that the in-degree of every vertex is at most k. For example, suppose the graph is given by three vertices $\{a, b, c\}$ with undirected edges $(a, b), (b, c), (c, a)$. Now if we direct these edges as $(a \to b), (a \to c)$ and $(b \to c)$, then the in-degrees of a, b, c are $0, 1, 2$ respectively. Using the max-flow min-cut theorem on a suitable instance, show that such an orientation of edges exists if and only if for every non-empty subset S of vertices, $|E(S)| \leq k|S|$. Here $E(S)$ denotes the set of edges e for which both the end-points of e lie in S. (**8 marks**)

   Solution: We first construct a directed bipartite graph H as follows: on the left side, we have one vertex for each edge e in G – call this vertex a_e, and on the right side, we have one vertex for each vertex v of G – call this vertex b_v. We have an edge (a_e, b_v) if v is an end-point of e – note that these are directed edges. Now we add a special vertices s, t, and add edges (s, a_e) for each e and (b_v, t) for each t. The (a_e, b_v) edges have infinite capacity, the edges (s, a_e) have unit capacity and (b_v, t) have capacity k. We claim that there is a valid orientation of edges in G iff the max-flow in H is equal to m – the number of edges in G.

   To see this, suppose there is a valid orientation of edges in G such that in-degree of each vertex is at most k. We send a flow in H as follows: if an edge $e = (u, v)$ is oriented $(u \to v)$, then we send 1 unit of flow from a_e to b_v, otherwise we send one unit of flow from a_e to b_u. We send 1 unit of flow on each of the edges (s, u_e). Now, the total incoming flow at a vertex b_v is the in-degree of this vertex – since this is at most k, we can send out the same amount of flow on the edge (b_v, t). Conversely, suppose there is an integral flow of value m in H. Then each of the edges (s, a_e) is saturated. Since the flow is integral, 1 unit of flow goes out from a_e on one of the edges out of it. We orient the edge e in G accordingly.

   Now suppose $|E(S)| \leq k|S|$ for each S. The max-flow min-cut theorem now says that H has flow at least m iff s-t min-cut is at least m. Let X be a minimum cut, where $s \in X, t \notin X$. Let X_E denote the vertices of the form a_e in X and X_V be the vertices of the form b_v in X. Since X has finite capacity, if $a_e \in X_E$ for some edge $e = (u, v)$, then $b_u, b_v \in E$ as well. Thus, if S denotes $\{v : b_v \in X_V\}$ and F denotes $\{e : a_e \in X_E\}$, then each edge in F has both the end-points in S.

   Now the capacity of this cut is (here n is the number of vertices)

   $$(m - |X_E| + k|X_V|).$$

   Therefore, the capacity is at least m if $|X_E| \leq k|X_V|$. Since $|X_E| = |F|$ and $|X_V| = S$, and F is a subset of $E(S)$, this condition is true. Hence, max-flow is equal to m.

   Conversely suppose there is a valid orientation. Then consider the graph $(S, E(S))$ with the edge orientations In this graph, the total number of edges, i.e., $|E(S)|$ is equal to the sum of the in-degrees of all the vertices, which is at most $k|S|$.

   Common Mistake: A common mistake in Q3 is that drawing the Bipartite Graph with vertices connected to source and edges connected to sink. If they use that graph, then it needs to be handled a bit differently.

4. We call a subset S of vertices in an undirected graph G to be triangle-free if there is no subset of three distinct vertices a, b, c in S such that all three edges $(a, b), (b, c), (c, a)$ are present in G. Prove that the following problem is NP-complete: given a graph G and a value k, does G have a triangle free subset of size at least k ? (**7 marks**)

   Solution: The problem is in NP: a polynomial time verifier takes a subset S of vertices and checks if S contains at least k vertices and there are no triangles in S.

   We now reduce the independent set problem to this problem. Let (G, k) be an instance of the independent set problem. We produce a new graph G' as follows: we add n new vertices $v_1, \ldots, v_n$. For each v_i, we join v_i to each vertex of G. Now we claim that G has an independent set of size at least k iff G' has a triangle free subset of size at least $n + k$ (**Note**: it is important to add many new vertices to G'. If we had added just one vertex to G', this reduction will be wrong).

   We prove both directions of this statement. Suppose G has an independent set S of size k. Then $S \cup \{v_1, \ldots, v_n\}$ is triangle free and has size $n + k$. Conversely, suppose G' has a triangle free subset S' of size $n+k$. Since the original graph G had only n vertices, S' must contain at least 1 new vertex, say v_i. Let S be the set of vertices in S' which belong to the original graph G. Since we add only n new vertices to G', $|S| \geq k$. We now claim that S is an independent set in G. Indeed, if there are two vertices $u, v \in S$ with an edge (u, v), then (u, v, v_i) will form a triangle, which is not possible. Thus G has an independent set of size k.

5. You are organizing a music event over a period of n days. On each day, there are k disjoint time-slots (e.g., 8-9am, 9-10 am, etc.) and exactly one singer can perform during each time-slot. But there are following constraints: (i) A singer can perform in at most 3 time-slots during a day, (ii) A singer can perform in at most 10 time-slots during the whole event, (iii) During any day, at most half the performances (i.e., at most $k/2$ performances) can be done by a male singer. Assume that you are given a list of singers along with their gender information. Give an efficient algorithm which outputs a schedule satisfying these conditions (or outputs that no such schedule is possible). If using max-flow, clearly specify the set of vertices, edges, capacities and the meaning of these vertices or edges. (**7 marks**)

   Solution: We have the following set of vertices:

   - For each day i and slot j on day i, we have a vertex $D_{i,j}$.
   - For each day i and gender g (i.e., g can be M or F), we have vertices $G_{i,g}$ and $G'_{i,g}$.
   - For each singer s and day i, we have a vertex $S_{s,i}$.
   - For each singer s, we have a vertex S_s.

   We also have special vertices s and t. Now we specify the edges and the capacities. We have edges of capacity 1 from s to $D_{i,j}$ for all i, j. We have an edge from $D_{i,j}$ to $G_{i,M}, G_{i,F}$, each of capacity 1 (actually capacity does not matter here). Flow on this edge means whether a Male or Female singer is singing on this slot. We have an edge from $G_{i,g}$ to $G'_{i,g}$. When $g = M$, we place capacity $\lfloor k/2 \rfloor$ on this edge to indicate that there is a limit on the set of male singers each day. For a male singer s and day i, we have an edge $(G_{i,M}, S_{i,s})$ – this edge has capacity 3 and the flow on this edge implies that the number of slots for which the singer s sings on this day. Similarly, we have edge $(G_{i,F}, S_{i,s})$ of capacity 3 for a female singer. Then we have edges $(S_{s,i}, S_s)$ for each day i. These edges do not have any capacity. Finally we have edges (S_s, t) of capacity 10. Flow on these edges indicate the total number of slots in which singer s has slots on which s has sung during the entire event.

   OR

   (most students have given the following solution:) We have the following set of vertices:

   - special vertices s and t.
   - For each day i, a vertex D_i and edges (s, D_i) of capacity k.
   - For each day i and gender g (i.e., g can be M or F), we have vertices $G_{i,g}$. There are edges from D_i to $G_{i,g}$ where $g = M, F$. The edge to $G_{i,M}$ has capacity $k/2$.
   - For each singer s and day i, we have a vertex $S_{s,i}$. If the gender of the singer is g, then we have edges $G_{i,g}$ to $S_{s,i}$. The capacity of these edges is 3.
   - For each singer s, we have a vertex S_s. We have edges from $S_{s,i}$ to S_s. Finally edges from S_s to t of capacity 10.

6. You are given a bipartite graph with V_L and V_R being the vertices on the left and the right sides respectively. We say that a subset $S \subseteq V_L$ (i.e., S is a subset of vertices on the left side) is matchable if there is a matching that matches all the vertices in S. Suppose S and T are two subsets of V_L that are matchable. Let $s = |S|, t = |T|$ and suppose $s < t$ (S may **not** be a subset of T). Show that there is a subset $X \subseteq T \setminus S$ such that $|X| = t - s$ and $X \cup S$ is matchable. Here $T \setminus S$ denotes the vertices in T which are not present in S (Hint: take union of two suitable matchings). (**5 marks**)

   Solution: Let M_1 be a matching that matches S and M_2 be a matching that matches T. Consider the union of the two matchings $M := M_1 \cup M_2$. As discussed in class, each connected component of M is either a path or a cycle. In both the cases, the edges of the cycle or the path alternate between the two matchings. Since a cycle has equal number of edges from both the matchings, and in a path the number of edges from the two matchings differ by at most 1, there must be at least $t - s$ alternating paths in M such that each of these paths has 1 more edge from M_2 than from M_1. Call these paths $P_1, \ldots, P_k$, where $k = t - s$. Let P_i^1 and P_i^2 be the edges in P_i from M_1 and M_2 respectively. Hence, $|P_i^2| = |P_i^1| + 1$. We construct a new matching from M_1 by replacing the edges in P_i^1 by P_i^2 for each path P_i. Call this new matching M_1'. M_1' now has t edges and matches all the vertices in S. In fact the set of vertices matched by M_1 is also a subset of $S \cup T$. Thus, the set of vertices matched by M_1 can be expressed as $S \cup X$, where $X \subseteq T \setminus S$ and $|X| = t - s$.