CSC373— Algorithm Design, Analysis, and Complexity — Spring 2018

Solutions for Tutorial Exercise 10: Cliques and Intersection Graphs for Intervals

---

1. **Clique.** Given an undirected graph $G = (V, E)$ a clique (pronounced "cleek" in Canadian, heh?) is a subset of vertices $K \subseteq V$ such that, for every pair of distinct vertices $u, v \in K$, the edge (u, v) is in E. See Clique, Graph Theory, Wikipedia.

   A second concept that will be useful is the notion of a complement graph G_c. We say G_c is the complement (graph) of $G = (V, E)$ iff $G_c = (V, E_c)$, where $E_c = \{(u, v) \mid u, v \in V, u \neq v, \text{ and } (u, v) \notin E\}$ (see Complement Graph, Wikipedia). That is, the complement graph G_c is the graph over the same set of vertices, but it contains all (and only) the edges that are not in E.

   Finally, a clique $K \subset V$ of $G = (V, E)$ is said to be a **maximal clique** iff there is no superset W such that $K \subset W \subseteq V$ and W is a clique of G.

   Consider the decision problem:

   Clique: Given an undirected graph and an integer k, does there exist a clique K of G with $|K| \geq k$?

   Clearly, Clique is in NP. We wish to show that Clique is NP-complete. To do this, it is convenient to first note that the independent set problem IndepSet(G, k) is very closely related to the Clique problem for the complement graph, i.e., Clique(G_c, k). Use this strategy to show:

   $$\text{IndepSet} \equiv_p \text{Clique}. \tag{1}$$

   Solution for Q1: Suppose $G = (V, E)$ and its complement $G_c = (V, E_c)$. Suppose G has an independent set S of size $|S| = k$. By definition of independent set, for every $u, v \in S$, with $u \neq v$, it must be the case that G does not include the edge (u, v). That is, $e = (u, v) \notin E$. Therefore $e = (u, v) \in E_c$, i.e., e is an edge in the complement graph G_c. But this holds for every $u, v \in S$, so S is a clique in G_c.

   Similarly, it follows that if S is a clique in G_c then S is an independent set in G.

   Therefore, we have shown IndepSet(G, k) is true iff Clique(G_c, k) is true. Since the complement graph can be constructed in poly-time from G, and vice versa, the desired result follows.

2. Consider the interval scheduling problem we started this course with (which is also revisited in Assignment 3, Question 1). That is, suppose we are given a set of jobs $J(k)$ which start at time s_k and end at time f_k, where $s_k < f_k$ are non-negative integers for $k = 1, 2, \ldots, K$. We assume all the finish times are distinct. We wish to schedule a maximum-size subset of these jobs such that no two scheduled jobs overlap in time (i.e., the open time intervals (s_j, f_j) and (s_k, f_k) do not intersect for any pair of jobs $J(j)$ and $J(k)$ that are scheduled).

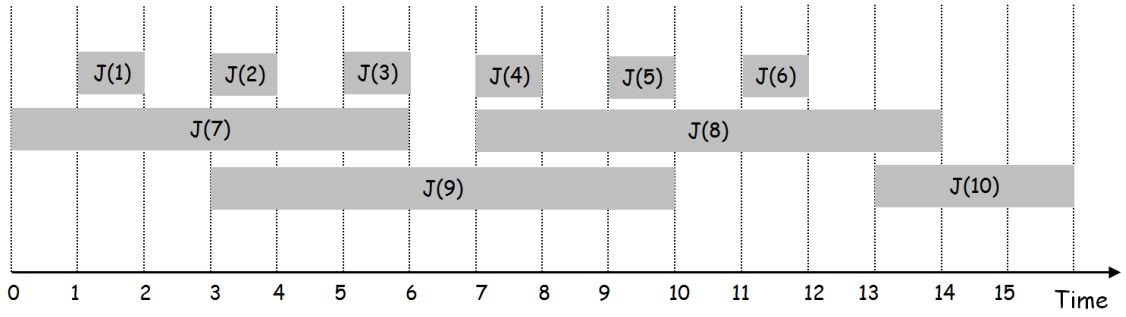
   Consider the intersection graph for this problem, say $G = (V, E)$, where the vertices, say v_n, are in one to one correspondence with the jobs, $J(n)$, and there is an edge $(v_n, v_m) \in E$ iff $n \neq m$ and jobs $J(n)$ and $J(m)$ intersect (i.e., $(s_n, f_n) \cap (s_m, f_m) \neq \emptyset$). In A3Q1 we show:

   $$\text{IntervalSched}(\{J(n)\}_{n=1}^{N}) \leq_p \text{searchIndepSet}(G).$$

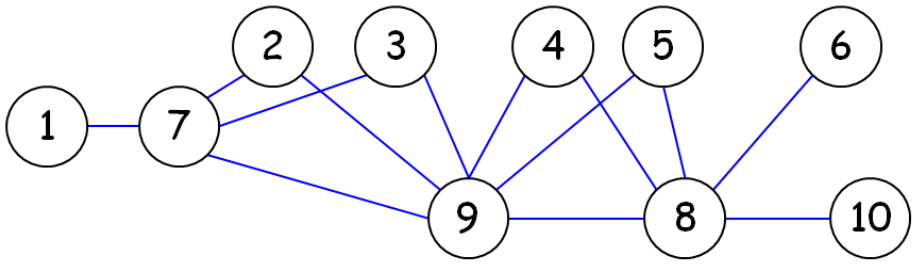
   Specifically, we show that finding a solution to the interval scheduling problem is equivalent to finding a maximum-sized independent set for G. Something must be very special about these graphs G to allow for the poly-time greedy solution. We reconsider this issue, this time with the concept of cliques in hand.

   (a) Choose any example of jobs $\{J(k)\}_{k=1}^{K}$ for the interval scheduling problem. Draw the corresponding intersection graph $G = (V, E)$. Find the set of all maximal cliques of G. (To find one maximal clique, you can pick any vertex v and set $T = \{v\}$. Pick any other vertex u such that there are edges from u to all the vertices in T. Add such a u to T, and repeat. When there is no such u, T is a maximal clique. To get the set of all maximal cliques, you need to do this using all possible ways of picking vertices to add to T.)

Solution Q2a. For example, consider the following jobs:

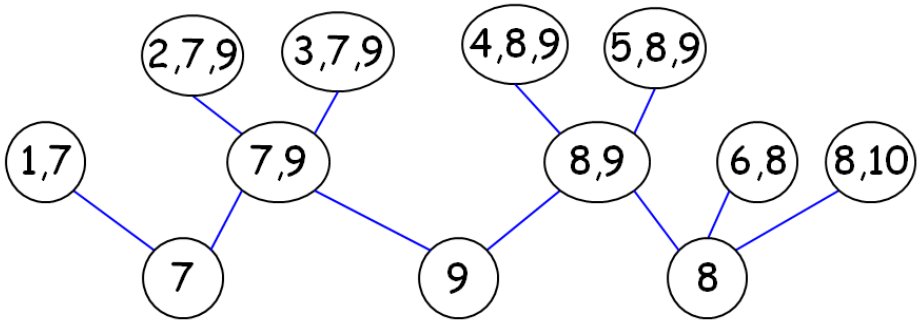


Intersection graph:



The sets of numbers in each of the vertices in the graph below indicate the original job numbers that are in each maximal clique.

(b) **Updated since handout.** Let $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ be the set of the maximal cliques found in part (a). Let $\mathcal{K}$ be the union of the maximal cliques $\mathcal{M}$ and all (non-empty) intersections $M_i \cap M_j$ for $M_i \neq M_j$. These will also be cliques, although not maximal.
Draw the clique graph $H = (\mathcal{K}, E_K)$ where each vertex of H is a clique in $\mathcal{K}$. Set the edges to be $(K_i, Kj) \in E_K$ iff K_i is a maximal subset of K_j in $\mathcal{K} \backslash K_j$ (or vice versa, since the edges are undirected). Then we have the following claim:
Claim 1: The graph H is a forest.
Is this claim true for your example?
Solution Q2b. The clique graph for the above graph is a tree:



Here the maximal cliques M_j are the leaves, and their intersections form the intermediate vertices. Note the vertex labelled (9) is a subclique many of the leaves, but it is only a maximal subset of the vertices labelled $(7, 9)$ and $(8, 9)$. Therefore, by the above definition, there are only the two edges $((9), (7, 9))$, and $((9), (8, 9))$ with the vertex (9) as an endpoint.

(c) Suppose $G = (V, E)$ is a graph such that the clique graph of G, say $H = (\mathcal{K}, T)$, is a forest. Devise a polynomial-time greedy algorithm to find a maximum-sized independent set for G. Do you see why it is correct? (You do not need to prove it.)

Solution Q2c.

Suppose $H = (\mathcal{K}, F)$ is the clique graph of $G = (V, E)$. We assume H is a forest.
Each vertex $K_j \in \mathcal{K}$ is associated with $s_j \subseteq V$, the set of vertices in the clique K_j.
$S \leftarrow \emptyset$ # The currently selected vertices in V for the independent set of G.
$M_j \leftarrow \emptyset$ for each $K_j \in \mathcal{K}$. # $M_j \subseteq s_j$ are "masked" vertices for clique K_j, which cannot be added to S.
while $\mathcal{K}$ not empty:
    # Loop Invariant: S and any neighbours of S are contained in the set $C \equiv \bigcup_{\{j|K_j \in \mathcal{K}\}} M_j \subseteq V$ of
    # masked vertices (More needs to be said about S and a maximum-sized independent set.)
    Let K_j be a leaf of H.
    If K_j has a neighbour in H:
        Let K_n be the neighbour of K_j # Unique because K_j is a leaf.
        $P \leftarrow (s_j \setminus M_j) \setminus s_n$ # We're free to pick any of these vertices from clique K_j.
        if $P \neq \emptyset$:
            Pick one $v \in P$
            $S \leftarrow S \cup \{v\}$
            $M_n \leftarrow M_n \cup (s_j \cap s_n)$ # All vertices s_j in K_j should be masked in K_n.
        else:
            $M_n \leftarrow M_n \cup (M_j \cap s_n)$ # Vertices masked in K_j should be masked in K_n too.
    else:
        # The case in which K_j does not have a neighbour in H:
        $P \leftarrow (s_j \setminus M_j)$ # We're free to pick any of these unmasked vertices.
        if $P \neq \emptyset$:
            Pick one $v \in P$
            $S \leftarrow S \cup \{v\}$
    Update $H = (\mathcal{K}, F)$ by removing K_j from $\mathcal{K}$, along with any edge (K_j, K_n) from F.
end while
return S

At least it works for the above graph. For example, if we work from left to right in the maximal clique graph shown above, then we find $S = \{1, 2, 3, 4, 5, 6, 10\}$, which is an independent set of the correct size. Also, the result does not seem to depend on the order in which we select the leaves in this example.

Note this algorithm always generates an independent set S because it carefully keeps track of all vertices chosen for S, and also all their neighbours, in terms of the masked sets M_k. (See the loop invariant.) Therefore it never selects such a vertex to add to S.

Note that the set P can be empty during the execution of the above algorithm. For example, suppose $G = (V, E)$ is a path with four vertices with the cliques $\{1, 2\}$, $\{2, 3\}$, and $\{3, 4\}$. If we process the first and last of these cliques first (they are both leaves) then, when we get to the last clique, $K_j = \{2, 3\}$, we find $P = \emptyset$.

However, I am not yet convinced that this algorithm is correct. That is, for any input graph $G = (V, E)$ for which $H = (\mathcal{K}, F)$ is a forest, and for any order for which we select leaves, is the returned set S an independent set of **maximum-size**? In order to convince myself of that, I could try adding a stronger loop invariant to this code. In particular, consider a loop invariant that ties the choices made in forming S at each stage to some maximal-size independent set for the original graph. That is, we still need to consider an "is promising" style proof.

In addition, Claim 1 above may or may not be correct. It deserves some thought, although only after the final exams are over.