CS 255: Homework 3

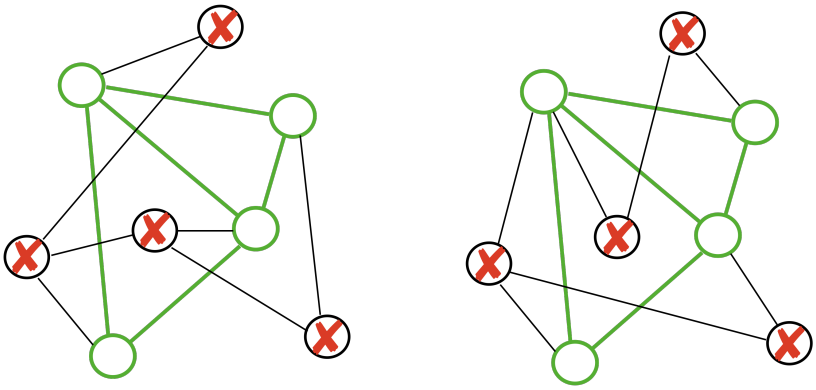
Due Friday November 10 at 11:59 pm

Instructions:

1. You may work in groups of up to 3 students.
2. Each group makes one submission on Canvas.
3. Include the name and SJSU ids of all students in the group.
4. You may discuss high level concepts with other groups, but do not copy other's work.

Problem 1

Suppose we are given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same vertices V. Consider the following problem: find the size of the smallest subset of vertices $S \subseteq V$ whose deletion leaves identical subgraphs $G_1 \setminus S = G_2 \setminus S$. For example, in the graphs below the smallest subset has size 4.



Give a polynomial time reduction for this problem from any of the following:

- MaxIndependentSet(G): find the size of the largest independent set $S \subseteq V$ (no pair of vertices in S are connected by an edge).
- MaxClique(G): find the size of the largest complete subgraph $S \subseteq V$ (every pair of vertices in S are connected).
- MinVertexCover(G): find the size of the smallest vertex cover $S \subseteq V$ (every edge in G is incident to at least one vertex in S).

A reduction is possible from all three problems, pick any *one* of them.

Solution: Let's call this problem MinDeletion. The intent was to reduce one of the three listed problems to the MinDeletion Problem.

MinVertexCover (MVC). Let $G = (V, E)$ be an undirected graph for an arbitrary instance of MVC. For the MinDeletion problem, we set $G_1 = G$ and $G_2 = (V, \emptyset)$. That is, G_2 contains all the vertices of G without any of the edges. Therefore, to make the graphs identical we need to choose a subset $S \subseteq V$ so that for every edge $e \in E$, at least one of its endpoints is in S. Then, S is a vertex cover. The smallest such S is a minimum vertex cover.

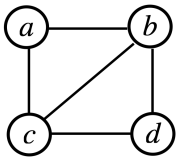
MaxIndependentSet. This case is nearly identical to MVC. Given a graph $G = (V, E)$, we construct G_1 and G_2 as above. Since the subset of vertices to remove $S \subset V$ contains *at least* one endpoint of every edge, then the remaining set of vertices contains *at most* one endpoint of every edge. That is, $V \setminus S$ is an independent set. Therefore, $V \setminus S$ is a maximum independent set. (You could say that the complement of a vertex cover is an independent set.)

MaxClique. In this case, we compare $G = (V, E)$ to a complete graph. That is, we set $G_1 = G$ and let $G_2 = (V, E')$ where $(u, v) \in E'$ for each pair $u \neq v \in V$. After removing the subset of vertices $S \subset V$, the remaining vertices $V \setminus S$ form a complete subgraph. That is, $V \setminus S$ is a clique. Since S is the minimum subset of vertices to remove to make a complete subgraph, $V \setminus S$ is a maximum clique.

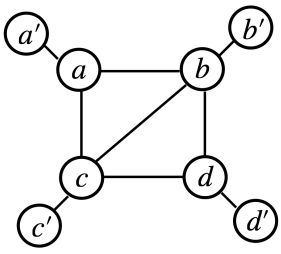
Problem 2

(a). A subset S of vertices in an undirected graph G is *half-independent* if each vertex in S is adjacent to at most *one* other vertex in S. Prove that finding the size of the largest half-independent set of vertices in a given undirected graph is NP-hard.

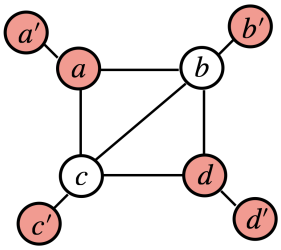
Solution: We give a reduction from IndependentSet. Given an input graph $G = (V, E)$ for an arbitrary instance of IndependentSet, we create a new graph $H = (V', E')$ by adding new vertices and edges to G. For each vertex $v \in V$, we create a new vertex v' and add an edge (v, v'). The new vertex does not connect to anything else. This step clearly takes polynomial-time. We call the vertices of G real, and the newly created ones copy vertices. Let C denote the set of copy vertices.



(a) Input graph G



(b) The graph H.



(c) A max half-independent set in H

It is easy verify that for any independent set I in G, the set $I \cup C$ is half-independent in H. The following claim shows the correctness of the reduction.

Claim: Let I^* be a maximum independent set of G, then $I^* \cup C$ is a maximum half-independent set of H.

Proof. Let S be a maximum half-independent set of H. First, we show that the set of real vertices of S form an independent set. For contradiction, suppose S contains two adjacent real vertices u and v (that is, $(u, v) \in E$). Notice that S does not contain either of the copy vertices u' or v', otherwise u and v are adjacent to two vertices in S. If we remove *one* of the vertices u or v, then we can add *both* copies u' and v' and obtain a strictly larger half-independent set. This contradicts that S has maximum size. Therefore, the set of real vertices of S form an independent set. Since a copy vertex v' is only adjacent to is corresponding real vertex v, then S must contain all copy vertices. Thus, $S = I \cup C$, for some independent set I of G. Then, the claim flows easily.

(b) A subset S of vertices in an undirected graph G is *sort-of-independent* if each vertex in S is adjacent to *at most 255* other vertices in S. Prove that finding the size of the largest sort-of-independent set of vertices in a given undirected graph is NP-hard.

Solution: (direct proof) We can generalize the construction and proof from part a. For each vertex $v \in V$, we create 255 new copy vertices $v_1, \ldots, v_{255}$ and edges (v, v_i) for each $i \in \{1 \ldots 255\}$. Again, let C denote the set of copy vertices. It is easy to verify that for any independent set I of G, $I \cup C$ is sort-of-independent in H.

Claim: Let I^* be a maximum independent set of G. Then, $I^* \cup C$ is a maximum sort-of-independent set in H.

Proof. Let S be a largest sort-of-independent set in H. For contradiction, suppose S contains a real vertex v and $j \in \{1 \ldots 255\}$ other real vertices that are adjacent to v. This means that j of v's copy vertices are not in S, and that for each adjacent real vertex $u \in S$ at least of it's copy vertices is not selected. If we remove v, then we can add all of these missing copy vertices yielding a net gain of $2j - 1 > 0$ additional vertices. This contradicts that S was the largest sort-of-independent set in H. The claim now follows from the same reasoning as part a.

Alternate Solution: (induction) For any $k \geq 0$, define a k-independent set as a subset of vertices $S \subseteq V$ so that any vertex $v \in S$ is adjacent to at most k other vertices in S. For example, a 0-independent set is the standard problem, a 1-independent set is half-independent (part a), and part (b) asks for a 255-independent set.

We show that finding the size of the largest k-independent set is NP-hard for all $k \geq 0$ by induction. We proved the base case, $k = 0$ (the standard IndependentSet problem), in lecture using a reduction from 3SAT. Now assume that for some $k \geq 0$, the k-IndependentSet problem is NP-hard. We show that the $(k + 1)$-IndependentSet problem is hard using a similar approach to (a). Let $G = (V, E)$ be the k-IndependentSet instance. We create the $(k + 1)$ instance (H) by adding a single copy vertex v' for each $v \in V$, and adding an edge (v, v'). Observe that if $S \subseteq V$ is k-independent in G, then $S \cup C$ is $(k + 1)$-independent in H.

Claim: Let I^* be a maximum k-independent set in G. Then, $I^* \cup C$ is a maximum $(k+1)$-independent set in H.

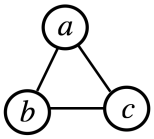
Proof. We use an exchange argument to show that we can swap certain real vertices with their copy counterparts. Let S be a largest $(k+1)$-independent set in H. Suppose that some vertex $v \in S$ is adjacent to $(k+1)$ real vertices in S. This means the copy vertex $v' \notin S$, otherwise v is adjacent to $k+2$ vertices in S. We can swap v and v' without decreasing the size of S. Repeating this replacement for all real vertices adjacent to $(k+1)$ real vertices in S proves the claim.

Problem 3

Consider the following heuristic for a vertex cover of a connected graph G. Choose any vertex $v \in V$ as the root. Compute any depth-first spanning tree of G. Return the set of non-leaf nodes from the tree.

(a). Prove that this heuristic returns a vertex cover of G.

Solution: This follows from the properties of DFS. Let T be the spanning tree, and let $L(T)$ denote the leaf nodes, and $I(T) = V \setminus L(T)$ denote the internal nodes of T. Clearly, every edge (u, v) where $u \in I(T)$ or $v \in I(T)$ is covered. The only potential issue is an edge between two leaf nodes $u, v \in L(T)$. For contradiction, suppose such an edge exists. One of these vertices, say u, is explored first in DFS. When we check the neighbors of u, all of them (including v) have already been explored since u is a leaf. However, this contradicts that u was visited first in the DFS. Therefore, selecting non-leaf nodes $I(T)$ gives a vertex cover. [Note: Observe that non-leaf nodes of a BFS spanning tree are not necessarily a vertex cover. For example, consider the figure below. If we start BFS from a, then in the next level we find b and c which means they are both leaves. Therefore, a is the only non-leaf node which is clearly not a vertex cover.]



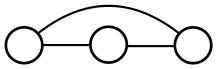
(b). Prove that this heuristic is a 2-approximation to the minimum vertex cover.

Solution: The proof hinges on the following key observation.

Claim 1: Let T be any spanning tree of G. Then, the size of the minimum vertex cover of T is less than or equal to the size of minimum vertex cover of G.

Proof. T has the same vertices as G, but only contains a subset of the edges. Therefore, any vertex cover of G is a vertex cover of T. However, since T contains only a subset of the

edges of G, there can be vertex covers of T that do not cover every edge of G. Therefore, feasible vertex covers for T are a superset of vertex covers for G. It follows that the minimum vertex cover for T can not be larger than the minimum vertex cover for G.



(a) The entire graph G

(b) A vertex cover of a spanning tree of G shown in red.

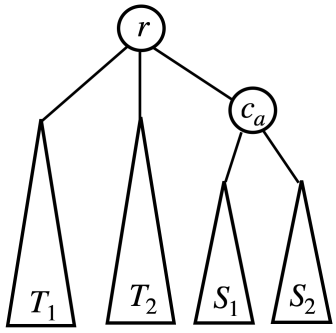
Let $MVC(H)$ be the size of the minimum vertex cover of a graph H. We want to show that the non-leaf nodes of the DFS spanning tree T give a 2-approximation to the minimum vertex cover of T. Using Claim 1, this shows that $ALG \leq 2 \cdot MVC(T) \leq 2 \cdot MVC(G)$. The proof that $ALG \leq 2 \cdot MVC(T)$ borrows ideas for the greedy approximation discussed in Lecture 17, and ultimately follows from the claim below.

Claim 2: Let T be a rooted spanning tree with n vertices and ℓ leaves. Then, T contains a matching of size at least $(n - \ell)/2$.

Before we prove Claim 2, let's see how this gives us the 2-approximation to minimum vertex cover of the spanning tree T, ie., $ALG \leq 2 \cdot MVC(T)$. Let M be a maximum matching in T. For every edge $(u, v) \in M$, a vertex cover must include u or v (this is same argument as the greedy approximation discussed in lecture). By Claim 2, the size of the minimum vertex cover of T is at least $|M| \geq (n-\ell)/2$. Our algorithm uses all $(n-\ell)$ non-leaf vertices, so it is a 2-approximation.

Proof (Claim 2): Again, let $I(T)$ denote the internal (non-leaf) nodes of a tree T. The proof is by induction on the depth d of T. In the base case, $d = 0$, the tree contains only the root which is also a leaf so the statement is true. Now suppose that any rooted tree of depth $\leq d$ contains a matching of size at least $I(T)/2$. Let r be the root of a tree T of depth $d + 1$. The children of r are the roots of the subtrees $\{T_1 \ldots T_a\}$. Let c_a be the child of r that is the root of the subtree T_a. The children of c_a are the roots of the subtrees $\{S_1 \ldots S_b\}$ (refer to the figure below).

Suppose that we match (r, c_j) and use maximum matchings in all other subtrees $\{T_1 \ldots T_{a-1}\}$



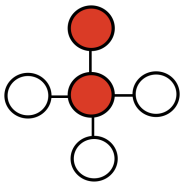
and $\{S_1 \dots S_b\}$. By the induction hypothesis, this gives a matching of size:

$$\begin{aligned}
&1 + \sum_{i=1}^{a-1} I(T_i)/2 + \sum_{j=1}^{k} I(S_j)/2 \\
&= 1 + \frac{1}{2}\big(\sum_{i=1}^{a-1} I(T_i) + \sum_{j=1}^{k} I(S_j)\big) \\
&= 1 + (I(T) - 2)/2 \\
&= I(T)/2,
\end{aligned}$$

where the second equality uses that the internal vertices of the subtrees $\{T_1 \dots T_{a-1}\}$ and $\{S_1 \dots S_b\}$ are the same as the internal vertices of T without r and c_a.

(c). Describe an infinite family of graphs for which this heuristic returns a vertex cover of size $2 \cdot OPT$.

Solution: If the DFS starts on the leaf of a star graph, then we get a 2-approximation. The algorithm uses two vertices: the root and the internal vertex of the star, while the optimal uses only the star's center. I also accepted odd length paths.



Problem 4

The Partition problem asks if a set of positive integers X can be partitioned into disjoint subsets A and B, so that the sum of the numbers in each subset is the same. This problem is NP-hard. Consider the following optimization version, partition the set of positive integers X into disjoint subsets A and B so that $\max\{\sum_{a \in A} a, \sum_{b \in B} b\}$ is as small as possible.

(a). Prove that the following algorithm is 3/2-approximation.

GreedyPartition$(X[1 \dots n])$**:**
  $a \leftarrow 0$, $b \leftarrow 0$
  for $i \leftarrow 1$ **to** n **do**
    if $a < b$ **then**
      $a \leftarrow a + X[i]$
    else
      $b \leftarrow b + X[i]$
  return $\max(a, b)$

[Hint: In the best case, the maximum would be $M = \frac{1}{2}\sum_{i=1}^{n} X[i]$. Use a similar analysis to online job scheduling (Lecture 13) and consider two cases: $\max_j X[j] \geq M$ and $\max_j X[j] < M$.]

Solution: Without loss of generality, we can assume that $a \geq b$. Let $X[j]$ be the last number added to a, and let a' and b' be the values of a and b just before this happens. Now, $X[j]$ is added to a since $a' \leq b'$. Therefore, the largest possible value for a' is:

$$a' \leq \frac{1}{2}\sum_{i \neq j} X[i] = \frac{1}{2}\big(\sum_{i=1}^{n} X[i] - X[j]\big) = M - X[j]/2,$$

where $M = \frac{1}{2}\sum_{i=1}^{n} X[i]$. Since $a = a' + X[j]$, this yields the upper bound:

$$a \leq M + X[j]/2.$$

Next we lower bound OPT. For any partition (A, B) of X, we have $\max(A, B) \geq (A + B)/2 = \frac{1}{2}\sum_i X[i] = M$. Therefore, $OPT \geq M$. This bound is loose when $X[j]$ is large. Specifically, if $X[j] > M$, then $\sum_{i \neq j} X[i] < M$ so that the optimal partition is $\{X[j],\ \cup_{i \neq j} X[i]\}$ and $OPT = X[j]$. In summary, we have:

$$OPT \geq \max(M, X[j]).$$

To compute the approximation ratio, we consider two cases. First, suppose $X[j] \leq M$. Then:

$$ALG \leq M + X[j]/2 \leq M + M/2 \leq \frac{3}{2}OPT.$$

Alternatively, if $X[j] > M$:

$$ALG \leq M + X[j]/2 \leq X[j] + X[j]/2 = \frac{3}{2}X[j] \leq \frac{3}{2}OPT.$$

(b). Prove that if the array X is sorted in non-decreasing order, then approximation ratio remains 3/2. That is, give an example of a sorted array X where the output of GreedyPartition is 50% larger than the cost of the optimal partition.

Solution: Suppose $X = \{1, 1, 2\}$. GreedyPartition creates the sets $\{1, 2\}$ and $\{1\}$. The optimal is $\{1, 1\}$ and $\{2\}$.