CS787: Advanced Algorithms	**Scribe:** Tianyu Liu
Lecture 1: Intro; Vertex Cover; Independent Set	**Date:** Jan 22, 2019

1.1 VertexCover and IndependentSet

A *vertex cover* of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set. Formally, a vertex cover C of an undirected graph $G = (V, E)$ is a subset of V such that for any $\{u, v\} \in E$, $u \in C \vee v \in C$. The entire vertex set V is a trivial vertex cover of G of size $|V|$. A *minimum vertex cover* is a vertex cover of the smallest size. The optimization problem of MinVertexCover is to find the size of *a* minimum vertex cover in a graph G. The decision problem of VertexCover is to decide, given a parameter k, if there is a vertex cover of size k for a graph G.

Problem : VertexCover.
Instance : Graph G and a nonnegative integer k.
Question: Does G have a vertex cover of size k?

A brute force algorithm solving VertexCover enumerates every set $C' \subseteq V$ of size k and check if C' is a vertex cover. This algorithm takes $O(n^k \operatorname{poly}(n))$ time where $n = |V|$.

An *independent set* of a graph is a set of vertices such that no two vertices in the set are adjacent. Formally, An independent set S of an undirected graph $G = (V, E)$ is a subset of V such that for any $u, v \in S$, $\{u, v\} \notin E$. A *maximum independent set* is an independent set of the largest size. The optimization problem of MaxIndependentSet is to find the size of a maximum independent set in a graph G. The decision problem of IndependentSet is to decide, given a parameter k, if there is an independent set of size k for a graph G.

Problem : IndependentSet.
Instance : Graph G and a nonnegative integer k.
Question: Does G have an independent set of size k?

A brute force algorithm that solves IndependentSet enumerates every set $S' \subseteq V$ of size k and check if S' is an independent set. This algorithm takes $O(n^k \operatorname{poly}(n))$ time where $n = |V|$.

Fact 1.1.1 *A set of vertices is a vertex cover if and only if its complement is an independent set.*

Proof: Let C be a vertex cover of $G = (V, E)$ and suppose that $V \setminus C$ is not an independent set. Then there exists $u, v \in V \setminus C$ such that $\{u, v\} \in E$, contradicting that C is a vertex cover. The reverse direction goes similarly. ■

1.2 VertexCover on trees and bipartite graphs

Although VertexCover and IndependentSet are NP-complete problems, they admit polynomial-time algorithms if the input is from some graph classes such as trees and bipartite graphs.

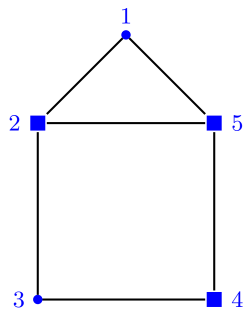

Figure 1.1.1: The house graph. The vertex set $\{v_2, v_4, v_5\}$ is a minimum vertex cover, and its complement $\{v_1, v_3\}$ is a maximum independent set.

1.2.1 Trees

Let T be a tree with root r. Observe that: if r is in a vertex cover, then all children of r can be either in or out of the vertex cover; if r is not in a vertex cover, then all children of r have to be in the vertex cover. The same observation holds for every subtree of T.

For a subtree T_u with root u, denote by $M_{\text{in}}(u)$ the size of the minimum vertex cover containing u, and by $M_{\text{out}}(u)$ the size of the minimum vertex cover not containing u. If u is a leaf, then $M_{\text{in}}(u) = 1$ and $M_{\text{out}}(u) = 0$; if u has children, then we have the following recursive formula:

$$\begin{aligned} M_{\text{in}}(u) &= 1 + \sum_{v \in \text{children}(u)} \min(M_{\text{in}}(v), M_{\text{out}}(v)), \\ M_{\text{out}}(u) &= \sum_{v \in \text{children}(u)} M_{\text{in}}(v). \end{aligned}$$

$M_{\text{in}}(r)$ and $M_{\text{out}}(r)$ can be computed in polynomial time using a standard dynamic programming algorithm. The solution is hence $\min(M_{\text{in}}(r), M_{\text{out}}(r))$.

1.2.2 Bipartite graphs

Since the maximum matching problem in bipartite graphs can be solved in polynomial time using network flow algorithms, *König's Theorem* implies that VertexCover is also tractable in bipartite graphs.

Theorem 1.2.1 (König's Theorem) *For any bipartite graph, the size of a minimum vertex cover is exactly equal to the size of a maximum matching.*

Recall the network flow algorithms that solves the maximum matching problem for a bipartite graph $G = (L, R, E)$. It creates a network $G' = (L \cup R \cup \{s, t\}, E')$ from $G = (L \cup R, E)$ such that:

- A "source" vertex s is introduced. For each vertex $l \in L$, add (s, l) to E' with capacity 1.
- A "sink" vertex t is introduced. For each vertex $r \in R$, add (r, t) to E' with capacity 1.

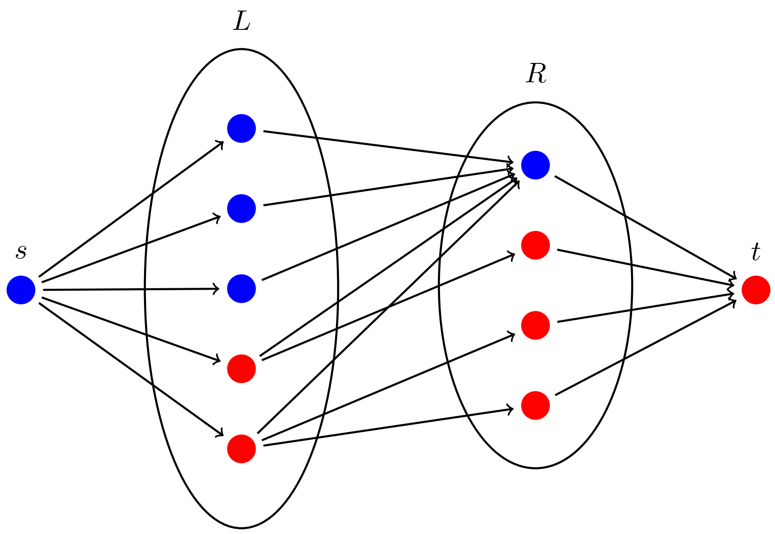

Figure 1.2.2: An example bipartite graph $G = (L \cup R, E)$. The blue and red vertices denote the vertex sets S and T respectively.

- For each edge $e = \{l, r\} \in E$ with $l \in L$ and $r \in R$, add (l, r) to E' with capacity ∞. [1]

The algorithm runs *Ford-Fulkerson algorithm* on G' and finds a maximum flow f as well as a minimum s-t cut $c = (S, T)$. Let $C = (S \cap R) \cup (T \cap L)$. We substantiate the following claims with proof sketches which imply Theorem 1.2.1.

Claim 1.2.2 *For any edge $(l, r) \in E'$ with $l \in L$ and $r \in R$, $(l, r) \notin S \times T$. In other words, the s-t cut c does not cut any edge from L to R.*

Proof: The value of the maximum flow f cannot exceed $\max(|L|, |R|)$, so is the size of the minimum cut c. For any edge $(l, r) \in E'$ where $l \in L$ and $r \in R$, the capacity on (l, r) is ∞. Therefore, $(l, r) \notin c$; otherwise the size of c becomes ∞. ∎

Claim 1.2.3 *$|C|$ is equal to the size of a maximum matching in G.*

Proof: According to Claim 1.2.2, no edge in $(S \cap L) \times (T \cap R)$ is in the cut c. Thus $|C|$ is the size of the minimum cut c. By the *max-flow min-cut theorem*, C is equal to the value of the maximum flow f which is also the size of a maximum matching in G. ∎

Claim 1.2.4 *C is a vertex cover of G.*

Proof: Suppose for contradiction that there is an edge $\{l, r\} \in E$ not covered by C where $l \in L, r \in R$. Then $l \in S$ because if $l \in T$, it would have been in C by definition; also $r \in T$ for similar reasons. This means (l, r) is in the cut $c = (S, T)$, contradicting Claim 1.2.2. ∎

By Claim 1.2.3 and Claim 1.2.4, we know that C is a vertex cover whose size is equal to the size of

[1]Note that this construction is different from the one given in lecture, which instead sets the capacity for each edge $e = \{l, r\}$ to be 1, not ∞. Setting the capacity to ∞ simplifies the proof of Claim 1.2.2.

a maximum matching in G. On the other hand, we know that the size of any vertex cover C' is no less than the size of any maximal matching M'. This is because in order to cover the edges in M', C' has to contain one of the two end points for each edge in M'. Thus Theorem 1.2.1 is proved.

1.3 A 2-approximation algorithm for MinVertexCover

Let P be a minimization problem and I an instance of P. Let A be an algorithm that computes feasible solutions given instances of P. Denote $A(I)$ as the result returned by A for instance I, and $\mathrm{OPT}(I)$ as the optimal solution for I. Then A is an α-approximation algorithm for P if for all instances I,

$$\frac{A(I)}{\mathrm{OPT}(I)} \leq \alpha$$

where $\alpha \geq 1$. Since P is a minimization problem and $A(I) \geq \mathrm{OPT}(I)$, a 1-approximation algorithm produces an optimal solution, and an α-approximation algorithm with a large α may return a solution that is much worse than the optimal. Therefore, the smaller α is, the better an approximation algorithm is.

Let us consider the minimization problem of MinVertexCover and a simple algorithm for it.

Problem : MinVertexCover.
Instance : Graph G.
Output: Smallest nonnegative integer k such that G has a vertex cover of size k.

Algorithm 1: A 2-approximation algorithm for MinVertexCover

Input: G
Find a maximal matching M in G
$S \leftarrow$ all end points of edges in M
return $|S|$

Algorithm 1 is 2-approximation algorithm for MinVertexCover because of the following two claims.

Claim 1.3.1 *S is a (not necessarily minimum) vertex cover.*

Proof: If any edge e is not covered by vertices in S, then both its two end points are not in M. Hence $M \cup \{e\}$ is also a matching, contradicting the fact that M is a maximal matching. ■

Claim 1.3.2 *Let k be the size of a minimum vertex cover in G. Then $|S| \leq 2k$.*

Proof: According to the algorithm, $|S| = 2|M|$. For any maximal matching M, a vertex cover has to contain at least one of the two end points for each edge to cover all edges in M, which yields $|M| \leq k$. ■