Lecture notes for Feb 13, 2023
2-SAT, greedy algorithms and matroids

Chun-Hung Liu

February 13, 2023

Recall that we proved the following theorem last time.

Theorem 1 *Given the input digraph D, in linear time, we can find all strongly connected components of D and order them into $C_1, C_2, ..., C_k$ such that for every $1 \leq i \leq k$, there exists no edge of D from $\bigcup_{j=1}^{i} V(C_j)$ to $V(D) - \bigcup_{j=1}^{i} V(C_j)$.*

1 2-SAT

Knowing how to find strongly connected components also helps us to solve 2-SAT.

Corollary 2 *2-SAT can be solved in linear time.*

Proof. Let f be an instance of 2-SAT. Let $x_1, x_2, ..., x_n$ be the variables, and let $c_1, c_2, ..., c_m$ be the clauses. Let D be the digraph with $V(D) = \{x_i, \neg x_i : i \in [n]\}$ and $E(D) = \{(u, v) : \text{some clause in } f \text{ is logically equivalent to } \neg u \vee v\}$. Note that $|V(D)| = 2n$, $|E(D)| = 2m$, and constructing D takes time $O(n + m)$.

Use the previous algorithm to find all strongly connected components and an ordering of them $B_1, B_2, ..., B_k$ such that for every $1 \leq i \leq k$, there exists no edge of D from $\bigcup_{j=1}^{i} V(B_j)$ to $V(D) - \bigcup_{j=1}^{i} V(B_j)$. This step takes time $O(|V(D)| + |E(D)|) = O(n + m)$.

Claim 1: If there exist $i \in [n]$ and $j \in [k]$ such that $\{x_i, \neg x_i\} \subseteq V(B_j)$, then f is a negative instance.

Proof of Claim: Suppose to the contrary that f is satisfiable. Let g be a truth assignment. By symmetry, we may assume that $g(x_i)$ is True. Equivalently, $g(\neg x_i)$ is False. Since x_i and $\neg x_i$ are contained in B_j, there exists a directed path P in D from x_i to $\neg x_i$. Denote P by x_i-y_1-y_2-...-y_t-$\neg x_i$ for some nonnegative integer t. Let $y_0 = x_i$ and $y_{t+1} = \neg x_i$. By the definition of $E(D)$, for every $0 \leq j \leq t$, some clause of f is logically equivalent to $\neg y_j \vee y_{j+1}$. Since $y_0 = x_i$ is assigned to be True and f is satisfiable, y_1 is assigned to be True. By induction, we know y_j is assigned to be True for all $0 \leq j \leq t+1$. In particular, $\neg x_i$ is assigned to True, a contradiction. □

Note that we can test whether there exist $i \in [n]$ and $j \in [k]$ such that $\{x_i, \neg x_i\} \subseteq V(B_j)$ in linear time. If it happens, then we conclude that f is a negative instance, which is correct by Claim 1. So now we may assume that for every $i \in [n]$, x_i and $\neg x_i$ are contained in different members of $\{B_j : j \in [k]\}$.

For every $i \in [n]$, let α_i be the member of $\{x_i, \neg x_i\}$ such that $\alpha_i \in V(B_j)$ and the element in $\{x_i, \neg x_i\} - \{\alpha_i\}$ is contained in $V(B_{j'})$ for some $j < j'$. For every $i \in [n]$, let $g(x_i)$ be True if $\alpha_i = x_i$, and let $g(x_i)$ be False if $\alpha_i = \neg x_i$. (Equivalently, $g(\alpha_i)$ is True and $g(\neg\alpha_i)$ is False.) Note that g can be constructed in linear time.

Claim 2: g is a truth assignment of f.

Proof of Claim: Suppose to the contrary that g is not a truth assignment of f. So there exists a clause $c = \beta_i \vee \beta_j$ such that $\beta_i \in \{x_i, \neg x_i\}$, $\beta_j \in \{x_j, \neg x_j\}$, and $g(\beta_i)$ and $g(\beta_j)$ are False. By the definition of g, $\beta_i = \neg\alpha_i$ and $\beta_j = \neg\alpha_j$. By the definition of α_i, we know that $\beta_i = \neg\alpha_i$ is contained in B_{b_i} and α_i is contained in B_{a_i} for some $b_i > a_i$. Similarly, $\beta_j = \neg\alpha_j$ is contained in B_{b_j} and α_j is contained in B_{a_j} for some $b_j > a_j$. Since $c = \beta_i \vee \beta_j$, there exist an edge from $\neg\beta_i = \alpha_i$ to β_j and an edge from $\neg\beta_j = \alpha_j$ to β_i. By the ordering for $B_1, B_2, ..., B_k$, we know $a_i \geq b_j$ and $a_j \geq b_i$. But $b_j > a_j$, so $a_i \geq b_j > a_j \geq b_i$, contracting $b_i > a_i$. □

By Claim 2, we correctly find a truth assignment of f. ■

2 Greedy algorithms

A greedy algorithm is an algorithm that repeatedly chooses the "most favorite" element during the process. For example, BFS is a greedy algorithm because we construct the tree by repeatedly adding new edges, and at each time we add an edge, this edge is incident with the "oldest" vertices in the

tree, which is how we define "favorite" in this case. Similarly, DFS is a greedy algorithm. We will see more examples of greedy algorithms.

2.1 Minimum weighted spanning tree

A *weighted graph* is a pair (G, w), where G is a graph and $w : E(G) \to \mathbb{R}$ is a function. For each edge e of G, we call $w(e)$ the *weight* of e. Note that in some context, the weights are on the vertices, and the weights can be a non-real number. But we focus on real edge-weighted graphs here, unless otherwise specified.

For a weighted graph (G, w), the weight of a subgraph H, denoted by $w(H)$, is defined to be $\sum_{e \in E(H)} w(e)$. A natural question is to find a spanning tree with minimum weight.

==========

Kruskal's algorithm for finding minimum weighted spanning tree
Input: A weighted connected graph (G, w).
Output: A spanning tree T of G with $w(T)$ minimum.
Procedure:

Step 1: Sort the edges to obtain an ordering of the edges $e_1, e_2, ..., e_{|E(G)|}$ of G so that $w(e_1) \leq w(e_2) \leq ... \leq w(e_{|E(G)|})$. Set T to be the spanning subgraph of G with no edge.

Step 2: For $i = 1, 2, ..., |E(G)|$, if $T + e_i$ is a forest, then add e_i into T.

==========

Remark:

- Step 1 takes time $O(|E(G)| \log |E(G)| + |V(G)|)$. Step 2 can be implemented in a sophisticated way so that it runs in time $O(|E(G)| \cdot \alpha(|V(G)|))$, where α is the inverse Achermann function. So the entire algorithm runs in time $O(|E(G)| \log |E(G)| + |V(G)|)$.

- The proof for the correctness is almost identical to a more general algorithm for finding minimum weighted base of a matroid. We will only prove the matroid version.

2.2 Matroids

A *matroid* M is the pair $(E, \mathcal{I})$, where E is a set and $\mathcal{I}$ is a set of subsets of E satisfying the following conditions:

(M1) $\emptyset \in \mathcal{I}$.

(M2) Every subset of a member of $\mathcal{I}$ is a member of $\mathcal{I}$. (i.e. if $X \subseteq Y$ and $Y \in \mathcal{I}$, then $X \in \mathcal{I}$.)

(M3) If $A, B \in \mathcal{I}$ with $|A| < |B|$, then there exists $x \in B - A$ such that $A \cup \{x\} \in \mathcal{I}$.

The set E is called the *ground set* of M. Every member of $\mathcal{I}$ is called an *independent set* of M. Every maximal independent set is called a *base*. For every subset X of E, the *rank* of X is the maximum size of an independent subset of X. The *rank* of M is the rank of E. Every subset of E that is not a member of $\mathcal{I}$ is called a *dependent set* of M. Every minimal dependent set is called a *circuit*.

Examples:

- Let F be a field. Let A be a matrix over F (with finitely many rows and columns). Let E be the set of column vectors of A. Let $\mathcal{I} = \{S \subseteq E :$ the vectors in S are linearly independent$\}$. Then it is easy to verify (M1)-(M3) by using standard results in linear algebra, so $(E, \mathcal{I})$ is a matroid. Every matroid that is isomorphic to such a matroid is called a *representable matroid*. Note that each base of $(E, \mathcal{I})$ is a basis of the column space of A, and the rank of $(E, \mathcal{I})$ equals the rank of A.

- Let k be a nonnegative integer. Let E be a set with size at least k. Let $\mathcal{I} = \{S \subseteq E : |S| \leq k\}$. It is easy to see that $(E, \mathcal{I})$ is a matroid. Such a matroid is called a *uniform matroid*. Note that the rank of $(E, \mathcal{I})$ equals k, and every circuit has size $k + 1$.

- Let G be a graph. Let $E = E(G)$. Let $\mathcal{I} = \{S \subseteq E :$ the graph $(V(G), S)$ is a forest$\}$. Then $(E, \mathcal{I})$ is a matroid.

  (**Proof:** (M1) and (M2) obviously hold. So it suffices to prove (M3). Let $A, B \in \mathcal{I}$ with $|A| < |B|$. If B contains an edge e between different components of the graph $(V(G), A)$, then $A \cup \{e\} \in \mathcal{I}$ and

we are done. So we may assume that every edge in B has both ends in the same component of $(V(G), A)$. Hence every component of $(V(G), B)$ is a contained in a component of $(V(G), A)$. So the number of components c_B of $(V(G), B)$ is at least the number of components c_A of $(V(G), A)$. Since $(V(G), A)$ and $(V(G), B)$ are forests, $|A| = |V(G)| - c_A \geq |V(G)| - c_B = |B|$, a contradiction. $\square$)

Such a matroid is called the *cycle matroid of* G. Every matroid isomorphic to the cycle matroid of a graph is called a *graphic matroid*. Note that if G is connected, then bases of $(E, \mathcal{I})$ are exactly the edge-sets of spanning trees of G, and the rank equals $|V(G)| - 1$. And circuits of $(E, \mathcal{I})$ are exactly the edge-sets of cycles. Note that not every circuit has the same size.

- Let G be a connected graph. Let $E = E(G)$. Let $\mathcal{I} = \{S \subseteq E : G - S$ is connected$\}$. Then $(E, \mathcal{I})$ is a matroid. (Exercise.) Such a matroid is called a *bond matroid*. Note that every circuit is a minimal subset of edges such that its removal makes G disconnected. Such a minimal set is called a *bond* of G.

- Let D be a digraph. Let X, Y be distinct subsets of $V(D)$. Let $\mathcal{I} = \{S \subseteq X :$ there exist $|S|$ disjoint paths from Y to $X\}$. Then $(X, \mathcal{I})$ is a matroid. (It can be deduced from some proof of Menger's theorem. It will be clear after we discuss the Max-Flow-Min-Cut Theorem.) Such a matroid is called a *gammoid*.

- Let G be a bipartite graph with a bipartition $\{A, B\}$. Let $\mathcal{I} = \{S \subseteq A :$ there exists a matching in G saturating $S\}$. Then $(A, \mathcal{I})$ is a matroid because it is isomorphic to a gammoid. (It is not hard to prove that it is a matroid directly.) Such a matroid is called a *transversal matroid*.