CMPSCI 611 Advanced Algorithms **Fall 2000**

Lecture 4: September 21

Lecturer: Micah Adler *Scribes: Clayton T. Morrison and Xianglong Huang*

4.1 Kruskal's Minimum Spanning Tree (MST) Algorithm

4.1.1 Algorithm

Kruskal's MST algorithm is an example of a **greedy algorithm**.

Input: connected, undirected graph $G = (V, E)$,
and each edge in E has an associated weight (cost).
Output: a spanning tree of G with a minimum total weight.

(Recall that a *spanning tree* of G is a "free tree" (connected, acyclic, undirected graph) that connects all of the vertices of G.) Intuitively, the algorithm works as follows:

1 Set $F = \emptyset$ (where F is the set of edges defining our candidate spanning tree)
2 Sort the edges in E by increasing weight
3 For each edge in E, consider whether adding that edge to F introduces a cycle
If it does, ignore the edge
If it does not, add it to F
4 Return F

Put another way, at each step, the algorithm maintains a forest (F) and increments that forest by 1 edge at a time without introducing cycles, until all of the vertices are connected.

4.1.2 Running Time

The steps that govern running time are: the initial *sorting*, *checking* whether the edge being added introduces a cycle, and *updating* F after adding a new edge:

1. Sorting is $O(|E| \cdot \log |E|)$
2. Checking whether adding a new edge to F introduces a cycle depends on how we implement our bookkeeping of which vertices belong to which component trees of F. For now, we assume the following: maintain an array with an entry for each vertex; array values designate to which component tree of F the respective vertex belongs (a kind of disjoint-set data structure). Checking is thus reduced to $O(1)$, and there will be $|E|$ such checks.
3. Adding a new edge to F involves updating the vertex membership array: $O(|V|)$, and there will be $|V|$ many, for a total time $O(|V^2|)$

Total running time: $O(|V^2| + |E| \cdot \log |E|)$.
(A more efficient vertex membership implementation brings this down to $O(|E| \cdot \log |E|)$.)

4.1.3 Proof of Correctness

Claim 4.1 *Given any forest F, let S be the set of all possible spanning trees (including minimal spanning tree(s)) that include the forest F. There is a minimum cost $T \in S$ which includes e, the minimum weight edge that is: (1) not in the forest F, and (2) does not cause a cycle in F.*

Proof: Let $T' \in S$ be a minimum cost spanning tree such that edge $e \notin T'$. In this case, we assume that if we added e to T', we would get exactly one cycle. Figure 4.1 pictures this situation:

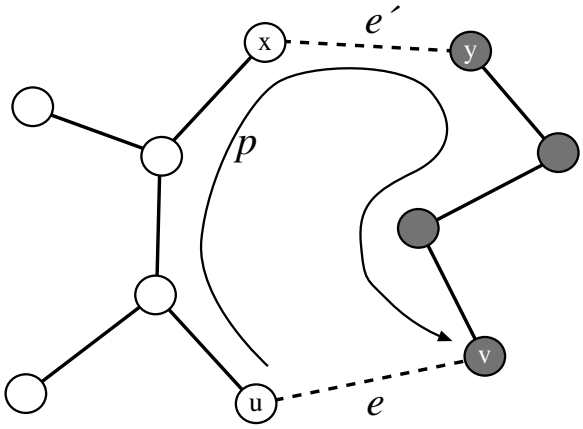

Figure 4.1: Depiction of forest F in S

In this case, we can consider T' as including the edge e', so that if we added edge e to T' we would get the cycle denoted in the figure by path p.

Claim 4.2 *there must be some edge $e' \neq e$ on cycle p such that $e' \notin F$ and $w(e') \geq w(e)$ (where $w(e)$ denotes the weight of e). We could replace edge e' in T' with e, producing spanning tree $T' \cup \{e\} - \{e'\}$, which has a total weight $\leq$ that of T'.*

■

Proof: Finally the proof that Kruskal's algorithm always yields a minimum spanning tree. Inductively:

Base Case: $F = \emptyset$

Inductive Step: Assume F is a subgraph of an MST. Then claim 4.1 tells us that $F + e$ (e being the minimal cost edge remaining) is also a subgraph of an MST.

So, F is always a subgraph of a minimum spanning tree. Furthermore, once the algorithm has terminated, since we cannot add any edges to F without creating a cycle we know that F must actually be a minimum spanning tree. ■

4.2 Matroids

4.2.1 Subset System

Definition 4.3 *A* **subset system** $S = (E, \mathcal{I})$ *is a finite set* E *with a collection* $\mathcal{I}$ *of subsets of* E *such that*

$$\text{if } i \in \mathcal{I} \text{ and } i' \subseteq i \text{ then } i' \in \mathcal{I}.$$

The above property is commonly stated as "$\mathcal{I}$ is closed under inclusion."

An example of closure under inclusion:

$$\begin{aligned} E &= \{e_1, e_2\} \\ \mathcal{I} &= \{\{e_1, e_2\}, \{e_1\}, \{e_2\}, \{\}\} \end{aligned}$$

(Note: in this example, $\mathcal{I}$ happens to be the power-set of E, but in general this need not be the case; however, $\mathcal{I}$ must include the power-set of every element of $\mathcal{I}$.)

The **subset system** is used for optimization problems on $(E, \mathcal{I})$ of the following general form:

Input: $(E, \mathcal{I})$, and weight function $w : E \to \Re^+$
Output: elements of $\mathcal{I}$ of maximum total weight

We refer to the elements of $\mathcal{I}$ as "independent sets."

4.2.1.1 Example: Maximum Weight Forest Problem (MWF)

This is an example of a optimization problem suited for use of **subset systems**. The MWF problem is defined as follows:

Input: $G = (V, E), w : E \to \Re^+$
Output: forest of G with maximum weight

Translating this into an optimization problem, following the form above, we have E, the edges of the graph, and $\mathcal{I}$, the acyclic subgraphs (i.e., the forests) of the graph G. We know this is closed under inclusion, since removing an edge from a forest always returns a forest.

An even more naive algorithm than a greedy approach would be to check all of the sets, which would be exponential in time.

A greedy algorithm for the optimization problem on $(E, \mathcal{I})$ is as follows:

$i = \emptyset$
Sort the elements of E by non-increasing weight
For each $e \in E$
    if $i + e \in \mathcal{I}$ **then** $i = i + e$
return i

It turns out that for some subset systems, this is very fast and produces optimal output. But for others, it does not. For what subset system does a greedy algorithm works?

- MWF: Greedy finds optimal. See the following example in Figure 4.2:

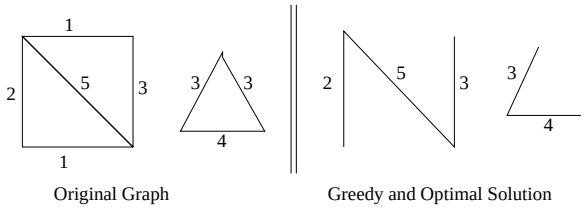

Figure 4.2: Example of MWF Algorithm

- MST: define weight function $w(e) = (w - \textit{cost of edge } e)$, where w is the maximum cost of any edge to solve MWF.

Definition 4.4 *: A matching M in a graph $G = (V, E)$ is a subset of edges $e \in E$ such that $\forall$ vertices $v \in V$, at most one edge in M is incident to v.*

Example, Figure 4.3:

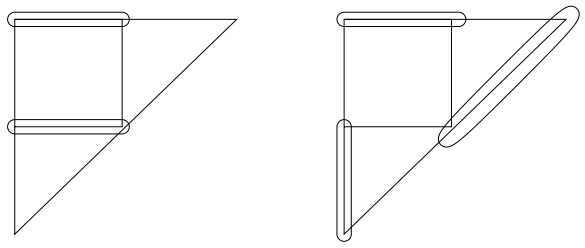

Figure 4.3: Example of two Matchings

4.2.1.2 Example: Maximum Matching Problem

Input: $G(V, E)$, $w : E \to \Re^+$
Output: a matching of maximum weight.

Figure 4.4 shows that for this problem, the greedy approach does not find the optimal solution:

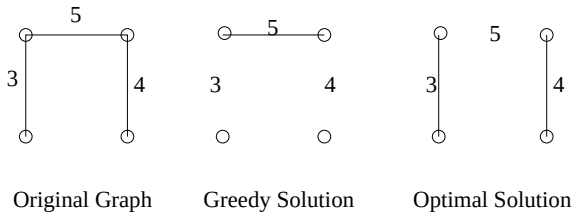

Figure 4.4: Comparison of Greedy and Optimal solutions for a Matching Problem

4.2.1.3 Example: Traveling Salesman Problem

Input: undirected graph $G = (V, E)$, $w : E \to \Re^+$
Output: a cycle of minimum weight that visits each vertex exactly once.

Another example for which the greedy algorithm does not find an optimal solution is pictured in Figure 4.5.

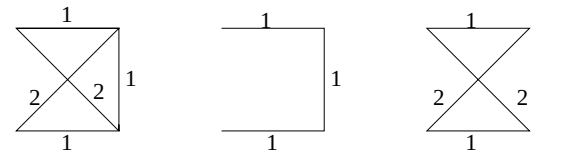

Figure 4.5: Comparison of Greedy and Optimal solutions for a Traveling Salesman Problem

So, what problems do greedy algorithms work for? $\rightarrow$ Matroids

4.2.2 Matroids

Definition 4.5 *A matroid is a subset system $M = (E, \mathcal{I})$ that satisfies the exchange property: if $i, i' \in \mathcal{I}$ such that $|i| < |i'|$, then there is some $e \in i' - i$, such that $i + e \in \mathcal{I}$.*

Example: The subset system for MWF is a matroid.

Proof: We need to show that if i and i' are forests, and $|i| < |i'|$ then i' has edge e such that $i + e$ is a forest. We will use, without proving, the knowledge that the number of connected components in a forest with K edges is $|V| - K$. Then

$|i| < |i'|$
$\Rightarrow$ i has more connected components than i'
$\Rightarrow$ i' has an edge e between connected components of i
$\Rightarrow$ $i + e$ is a forest. ■

Now we'll prove a fact about any subset system that has the matroid property, and use that fact to prove that any subset system with this property has an optimal greedy solution.

Theorem 4.6 *A subset system $(E, \mathcal{I})$ is a matroid if and only if $\forall A \subseteq E$, if i and i' are maximal independent subsets of A, then $|i| = |i'|$.*

Proof: ($\Longrightarrow$: i.e., if a matroid, then for all $A \subseteq E$, if i and i' are maximal independent subsets of A, then $|i| = |i'|$)

Let (E, I) be a matroid, and i, i' are maximal independent subsets of A.

If $|i| < |i'|$, then by the exchange property (which we have, since we're assuming we have a matroid) we can increase the size of i by $e \in i' - i$. And $i + e$ still belongs to A, which is a contradiction to the claim that i is a maximal independent subset.

Thus the only way i and i' can be maximal independent subsets of A and conform to the exchange property is if $|i| = |i'|$.

($\Longleftarrow$: i.e., if not a matroid, then not true for some A...)

If (E, I) is not a matroid, then $\exists i, i' \in I$, such that $|i| < |i'|$ but there is no $e \in i' - i$ that satisfies $i + e \in I$.

Let $A = i \cup i'$, then i is a maximal independent subset in A because there is no $e \in A$ that satisfies $i + e \in A$

There is some i'', such that $i' \subseteq i''$ and i'' is maximal independent subset in A.

Since $|i''| \geq |i'| > |i|$, we get i and i'' are both maximal independent subsets, and they satisfy $|i''| \neq |i|$. ■

Theorem 4.7 *The greedy algorithm solves the optimization problem for $(E, \mathcal{I})$ if and only if $(E, \mathcal{I})$ is a matroid.*