17.1 Introduction

We talked a lot the last lecture about greedy algorithms. While both Prim and Kruskal are greedy, we talked about how Kruskal is "super-greedy" – rather than doing anything unnecessary like choosing a starting point, it just keeps choosing the minimum weight edge that it can add without causing a cycle. A natural thing to do would be to abstract this out, and consider the question of "when does the greedy algorithm result in an optimal solution". That's the question we're going to try to answer today.

Let's first figure out the right abstraction. We want something which abstracts MSTs, but also works for other problems. So let's use abstract sets. In particular, we'll have a universe of elements U and a collection $\mathcal{I}$ of subsets of U. These are usually called *independent sets*, for reasons which will become apparent later. For MSTs, think of U as the set of edges, and $\mathcal{I}$ as the collection of all forests, i.e. all acyclic subgraphs. We are also given a weight function $w : U \to \mathbb{R}^+$. Our goal is to find a subset $I \in \mathcal{I}$ with *maximum* weight, i.e., to find $I \in \mathcal{I}$ which maximizes $\sum_{e \in I} w(e)$. In other words, we are trying to find a maximum weight independent set.

This seems backwards for MSTs, since there we were trying to compute a *minimum* weight spanning tree. But it's not hard to see that if we can do one then we can do the other (at least for MSTs – the proof for more general settings is obvious based on stuff we'll do later). For example, let $\bar{w}$ be some value which is larger than $w(e)$ for all e. Then we can replace each $w(e)$ with $w'(e) = \bar{w} - w(e)$. Since every spanning tree has exactly $n-1$ edges, a spanning tree T of total original weight $w(T)$ will have new weight $(n-1)\bar{w} - w(T)$. So minimizing the original weight is equivalent to maximizing the new weight. So from now on we'll talk about finding the independent set of maximum weight, and keep in mind that the MST problem is a special case.

17.2 Matroids

Let's think about what properties forests have which we can abstract out to our independent sets. There are two obvious properties:

1. $\emptyset \in \mathcal{I}$, and
2. If $F \in \mathcal{I}$ and $F' \subseteq F$, then $F' \in \mathcal{I}$.

Any set system with these two properties is called a *hereditary system*. The empty set is a forest, and if we start with a forest and take a subgraph, we still have a forest. Those are obvious. But there's another really nice property of forests which is not so obvious:

3. If $F_1 \in \mathcal{I}$ and $F_2 \in \mathcal{I}$ with $|F_2| > |F_1|$, then there is some element $e \in F_2 \setminus F_1$ such that $F_1 \cup \{e\} \in \mathcal{I}$.

This is sometimes called the *augmentation property*.

Lemma 17.2.1 *If $\mathcal{I}$ is the collection of forests of a graph G, then $\mathcal{I}$ has the augmentation property.*

Proof: Consider two forests F_1 and F_2 with $|F_2| > |F_1|$. We claim that there must exist an edge $\{a, b\} \in F_2$ where a and b are in different trees of F_1. To see this, suppose that it is false. Then every connected component of F_2 would be contained in a connected components of F_1. So F_2 has at least as many connected components as F_1. Since F_2 and F_1 are both forests, the number of connected components in F_2 is $n - |F_2|$ and the number of connected components of F_1 is $n - |F_1|$. Thus $n - |F_2| \geq n - |F_1|$, and thus $|F_2| \leq |F_1|$. This contradicts our assumption that $|F_2| > |F_1|$, and hence such an $\{a, b\}$ must exist. Since $\{a, b\}$ goes between different trees of F_1 we can add it to F_1 and we still have a forest. ■

It is not obvious why we chose to single out this property, but it turns out to be crucial. It is so crucial that set systems with these three properties have their own name: *matroids*.

Definition 17.2.2 *Let U be a universe of elements and $\mathcal{I}$ a collection of subsets of U. Then $(U, \mathcal{I})$ is a* matroid *if the following three properties hold:*

1. *$\emptyset \in \mathcal{I}$,*
2. *If $F \in \mathcal{I}$ and $F' \subseteq F$, then $F' \in \mathcal{I}$, and*
3. *If $F_1 \in \mathcal{I}$ and $F_2 \in \mathcal{I}$ with $|F_2| > |F_1|$, then there is some element $e \in F_2 \setminus F_1$ such that $F_1 \cup \{e\} \in \mathcal{I}$.*

There is a whole area of math called matroid theory, and many deep and interesting results are known. Today we're just going to touch the surface, and talk about their relationship to greedy algorithms. As a good warmup, though, note that the augmentation property implies that if $F_1 \in \mathcal{I}$ and $F_2 \in \mathcal{I}$ are both maximal (i.e., no element can be added to either of them while maintaining independence) then $|F_1| = |F_2|$. This value, the size of the maximal independent sets, is known as the *rank* of a matroid.

17.2.1 Examples of Matroids

We've seen that the forests of a graph form a matroid. But one of the powerful things about matroids is that they generalize a huge number of other interesting set systems. The most famous (and arguably most important) is the collection of linearly independent vectors in a vector space. Let's instantiate this with the vector space $\mathbb{R}^d$, although we're not going to use anything about this space in particular (it's just a vector space that everyone is familiar with).

Suppose that we are given a finite set U of vectors in $\mathbb{R}^d$. I'm assuming that you all know basic linear algebra, but as a refresher we say that a set $F \subseteq U$ of these vectors is *linearly independent* if no vector in F can be written as a linear combination of the others (there are many other equivalent definitions as well). Let $\mathcal{I} = \{F \subseteq U : F \text{ linearly independent}\}$.

Clearly the first two properties of a matroid are satisfied. But it's easy to see that the augmentation property is also satisfied! If $|F_1| < |F_2|$, and they are both linearly independent, then the dimension of the space spanned by F_1 is strictly less than the dimension of the space spanned by F_2, so there

is some vector from F_2 which is not in the subspace spanned by F_1. So we can add this vector to F_1.

There are many, many other examples of matroids, but forests in graphs and linearly independent sets in vector spaces are the two most famous. Indeed, matroids were originally developed as a tool for abstracting the meaning of "independence" in linear algebra, and it was soon realized that matroids also generalize a significant amount of graph theory.

17.3 Representation

So now our algorithmic goal is clear: given a matroid, find an independent set of maximum weight. Before we can really talk about algorithms, though, we need to talk about representation. If $\mathcal{I}$ is extremely small then perhaps all independent sets are just given to us as part of the input. Then we can calculate the weight of each one, and return the best. Easy! But in many cases, such as MSTs, it is not reasonable to assume that $\mathcal{I}$ is given to us explicitly. For example, for MSTs we are given a graph, not a list of all spanning trees. There can be an exponential number of spanning trees, so even writing them all down would take exponential time.

So let's think about what was going on in Kruskal's algorithm. In Kruskal we were given a graph, but we didn't actually do a whole lot with this graph. All we needed was a way to test whether adding an edge would cause a cycle. In other words, given a current forest (independent set) and a new edge (element) e, would adding e to our current solution keep it a forest (independent)?

This is called an *independence oracle*, and is the usual assumption when working with matroids. Slightly more formally, an independence oracle is an algorithm which, when given a set $S \subseteq U$, tells us whether $S \in \mathcal{I}$. Note, for example, that an independence oracle is easy to implement for forests in graphs (check S for cycles) and for linearly independent sets of vectors (using e.g. Gaussian elimination).

17.4 Greedy Algorithm

Now we can finally start talking about algorithms. With an independence oracle in hand, an obvious algorithm for finding a maximum weight independent set is the greedy algorithm:

1. Initialize $F = \emptyset$.
2. Sort U by weight from largest to smallest
3. Consider the elements in sorted order. For each element e, add it to F if $F \cup \{e\}$ is independent.

When applied to forests, this is exactly Kruskal's algorithm! But in fact it generalizes: the greedy algorithm is optimal on *any* matroid.

Theorem 17.4.1 *Let F be the independent set returned by the greedy algorithm. Then $w(F) \geq w(F')$ for all $F' \in \mathcal{I}$.*

Proof: Let $F = \{f_1, f_2, \ldots, f_r\}$ where $w(f_i) \geq w(f_{i+1})$ for all $i \in \{1, \ldots, r-1\}$. In other words this is the order we added the elements to F. Fix some other independent set $F' = \{e_1, \ldots, e_k\}$

where $w(e_i) \geq w(e_{i+1})$ for all $i \in \{1, \ldots, k-1\}$. Without loss of generality, we may assume that $k = r$ and r is the rank of the matroid (convince yourself of this! Remember that weights are nonnegative).

We will in fact prove the stronger statement that $w(f_i) \geq w(e_i)$ for all i. This clearly implies the theorem. So suppose that it is false, and let j be the smallest index at which it is violated, i.e. j is the first time where $w(f_j) < w(e_j)$. Let $F_1 = \{f_1, f_2, \ldots, f_{j-1}\}$, and let $F_2 = \{e_1, \ldots, e_j\}$. Then by the augmentation property, there is some element $e_z \in F_2 \setminus F_1$ so that $F_1 \cup \{e_z\}$ is independent. But $w(e_z) \geq w(e_j) > w(f_j)$. So the greedy algorithm would have chosen to add e_z instead of f_i! This is a contradiction, so proves the theorem. ■

So the greedy algorithm works on matroids, and for the same reason that Kruskal worked for MSTs. An astonishing fact is that the converse is also true: if the greedy algorithm works for all weight functions, then the independent sets are a matroid!

Theorem 17.4.2 *Let $(U, \mathcal{I})$ be an hereditary set system. If for every weighting $w : U \to \mathbb{R}_{\geq 0}$ the greedy algorithm returns a maximum weight independent set, then $(U, \mathcal{I})$ is a matroid.*

Proof: Suppose this is false. Then $(U, \mathcal{I})$ is not a matroid so there are some $F_1, F_2 \in \mathcal{I}$ with $|F_1| < |F_2|$ such that $F_1 \cup \{e\} \notin \mathcal{I}$ for all $e \in F_2 \setminus F_1$.

Clearly $|F_2 \setminus F_1| > |F_1 \setminus F_2|$ and $F_1 \setminus F_2$ is nonempty (or else F_1 would be a subset of F_2 so by the hereditary property we could add some element of F_2 to F_1). This means we can choose a positive number ϵ so that

$$0 < (1 + \epsilon)|F_1 \setminus F_2| < |F_2 \setminus F_1|.$$

and thus

$$\frac{1}{|F_1 \setminus F_2|} > \frac{1 + \epsilon}{|F_2 \setminus F_2|}. \tag{17.4.1}$$

Let's define a weight function using ϵ.

$$\begin{aligned}
w(e) &= 2 && \text{if } e \in F_1 \cap F_2 \\
w(e) &= \frac{1}{|F_1 \setminus F_2|} && \text{if } e \in F_1 \setminus F_2 \\
w(e) &= \frac{1 + \epsilon}{|F_2 \setminus F_1|} && \text{if } e \in F_2 \setminus F_1 \\
w(e) &= 0 && \text{otherwise}
\end{aligned}$$

What does the greedy algorithm do on this weight function? First it picks all elements of $F_1 \cap F_2$. Then it picks all elements of $F_1 \setminus F_2$ because of Equation (17.4.1). So at this point it has picked exactly the set F_1. Now since we assumed that no elements of $F_2 \setminus F_1$ could be added to F_1 while maintaining independence, the greedy algorithm never gets any more value. Thus the total weight obtained by the greedy algorithm is

$$2|F_1 \cap F_2| + |F_1 \setminus F_2| \frac{1}{|F_1 \setminus F_2|} = 2|F_1 \cap F_2| + 1.$$

On the other hand, the weight of F_2 is

$$2|F_1 \cap F_2| + |F_2 \setminus F_1| \frac{1+\epsilon}{|F_2 \setminus F_1|} = 2|F_1 \cap F_2| + 1 + \epsilon$$

This means that the greedy algorithm did not return the maximum weight independent set! But we assumed that the greedy algorithm worked for all weight functions, including this one. This is a contradiction. Thus no such F_1 and F_2 can exist, so $(U, \mathcal{I})$ is a matroid. ■