We’ve done

- Introduction to the greedy method
  - Activity selection problem
  - How to prove that a greedy algorithm works
  - Fractional Knapsack
  - Huffman coding

Now

- Matroid Theory
  - Matroids and weighted matroids
  - Generic matroid algorithms
  - Minimum spanning trees

Next

- A task scheduling problem

- Dijkstra’s algorithm
Matroids

A matroid $M$ is a pair $M = (S, \mathcal{I})$ satisfying:

- $S$ is a finite non-empty set
- $\mathcal{I}$ is a collection of subsets of $S$.
  (Elements in $\mathcal{I}$ are called independent subsets of $S$.)
- **Hereditary**: $B \in \mathcal{I}$ and $A \subseteq B$ imply $A \in \mathcal{I}$.
- **Exchange property**: if $A \in \mathcal{I}$ and $B \in \mathcal{I}$ and $|A| < |B|$, then $\exists x \in B - A$ so that $A \cup \{x\} \in \mathcal{I}$.

Example of a matroid

- $M_1 = (S_1, \mathcal{I}_1)$ where $S_1 = \{1, 2, 3\}$ and
  $$\mathcal{I}_1 = \{\{1, 2\}, \{2, 3\}, \{1\}, \{2\}, \{3\}, \emptyset\}$$

Example of a non-matroid

- $M_2 = (S, \mathcal{I}_2)$ where $S_2 = \{1, 2, 3, 4, 5\}$ and
  $$\mathcal{I}_2 = \{\{1, 2, 3\}, \{3, 4, 5\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{3, 5\}, \{4, 5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \emptyset\}$$

Why isn’t $M_2$ a matroid?
Graphs

- $G = (V, E)$, $V$ the set of vertices, $E$ the set of edges.
- $G$ is *simple* means there’s no multiple edge and no loop.
- $G' = (V', E')$ is a *subgraph* of $G$ if $V' \subseteq V$, and $E' \subseteq E$.
- A graph with no cycle is called a *forest*.
- A subgraph $G' = (V', E')$ of $G$ is *spanning* if $V' = V$.
- Other notions: *path*, *distance*.
- **Connected graphs**: there is a path between every pair of vertices.
- **Connected components**: maximal connected subgraphs.
Our First Interesting Matroid

Graphic Matroid

- $G = (V, E)$ a non-empty, undirected simple graph
- $M_G$: the graphic matroid associated with $G$
  - $M_G = (S_G, \mathcal{I}_G)$
  - $S_G = E$
  - $\mathcal{I} = \{A \mid A \subseteq E \& (V, A) \text{ is a forest}\}$

In other words, the independent sets are sets of edges of spanning forests of $G$. 
$M_G$ is a matroid

Lemma 1. A tree on $n$ vertices has precisely $n - 1$ edges, $n \geq 1$.

Lemma 2. A spanning forest of $G = (V, E)$ with $c$ components has precisely $|V| - c$ edges.

Theorem 3. If $G$ is a non-empty simple graph, then $M_G$ is a matroid.

Proof. Here are the steps:

- $S_G$ is not empty and finite
- $\mathcal{I}_G$ is not empty (why?)
- Hereditary is easy to check
- Exchange property if $A$ and $B$ are independent, i.e. $(V, A)$ and $(V, B)$ are spanning forests of $G$, then $(V, B)$ has less connected components than $(V, A)$.

Thus, there is an edge $e$ in $B$ connecting two components of $A$.

Consequently, $A \cup \{e\}$ is independent.
More terminologies and properties

Given a matroid $M = (S, \mathcal{I})$

- $x \in S$, $x \notin A$ is an extension of $A \in \mathcal{I}$ if $A \cup \{x\} \in \mathcal{I}$.

- $A \in \mathcal{I}$ is maximal if $A$ has no extension.

**Theorem 4.** *Given a matroid $M = (S, \mathcal{I})$. All maximal independent subsets of $S$ have the same size.*

Question: let $S$ be a set of activities, $\mathcal{I}$ be the collection of sets of compatible activities. Is $(S, \mathcal{I})$ a matroid?
Weighted Matroids

$M = (S, I)$ is weighted if there is a weight function

$$w : S \longrightarrow \mathbb{R}^+$$

(i.e. $w(x) > 0, \forall x \in S$).

For each subset $A \subseteq S$, define

$$w(A) = \sum_{x \in A} w(x)$$

The Basic Matroid Problem:

Find a maximal independent set with minimum weight

Example: minimum spanning tree (MST)

- Given a connected edge-weighted graph $G$, find a minimum spanning tree of $G$

- MST is one of the most fundamental problems in Computer Science.
Greedy Algorithm for Basic Matroid Problem

- Input: $M = (S, \mathcal{I})$, and $w : S \rightarrow \mathbb{R}^+$
- Output: a maximal independent set $A$ with $w(A)$ minimized
- Idea: greedy method

What’s the greedy choice?
Greedy Algorithm for Basic Matroid Problem (cont.)

\textbf{MATROID-GREEDY}(S, I, w)

1: \( A \leftarrow \emptyset \)
2: Sort \( S \) in increasing order of weight
3: // now suppose \( S = [s_1, \ldots, s_n], w(s_1) \leq \cdots \leq w(s_n) \)
4: \textbf{for} \( i = 1 \) to \( n \) \textbf{do}
5: \textbf{if} \( A \cup \{s_i\} \in I \) \textbf{then}
6: \( A \leftarrow A \cup \{s_i\} \)
7: \textbf{end if}
8: \textbf{end for}

What’s the running time?
Correctness of MATROID-GREEDY

Theorem 5. MATROID-GREEDY gives a maximal independent set with minimum total weight.

Proof. MATROID-GREEDY gives a maximal independent set (why?)

• Let $B = \{b_1, \ldots, b_k\}$ be an optimal solution, i.e. $B$ is a maximal independent set with minimum total weight.

• Suppose $w(b_1) \leq w(b_2) \leq \cdots \leq w(b_k)$.

• Let $A = \{a_1, \ldots, a_k\}$ be the output in that order.

• Then

\[ w(a_i) \leq w(b_i), \forall i \in \{1, \ldots, k\}. \]
Minimum spanning tree

Given a connected graph \( G = (V, E) \)

A weight function \( w \) on edges of \( G \), \( w : E \rightarrow \mathbb{R}^+ \)

Find a minimum spanning tree \( T \) of \( G \).

The Matroid-Greedy algorithm turns into Kruskal’s Algorithm:

\[
\text{MST-Kruskal}(G, w)
\]

1. \( A \leftarrow \emptyset \) // the set of edges of \( T \)
2. Sort \( E \) in increasing order of weight
3. // suppose \( E = [e_1, \ldots, e_m], w(e_1) \leq \cdots \leq w(e_m) \)
4. for \( i = 1 \) to \( m \) do
5.    if \( A \cup \{e_i\} \) does not create a cycle then
6.       \( A \leftarrow A \cup \{e_i\} \)
7.    end if
8. end for

What’s the running time?
Kruskal Algorithm with Disjoint Set Data Structure

\textsc{MST-Kruskal}(G, w)

1: \(A \leftarrow \emptyset\) // the set of edges of \(T\)
2: Sort \(E\) in increasing order of weight
3: // suppose \(E = [e_1, \ldots, e_m], w(e_1) \leq \cdots \leq w(e_m)\)
4: \textbf{for} each vertex \(v \in V(G)\) do
5: \hspace{1em} \textsc{Make-Set}(v)
6: \textbf{end for}
7: \textbf{for} \(i = 1\) to \(m\) do
8: \hspace{1em} // Suppose \(e_i = (u, v)\)
9: \hspace{1em} \textbf{if} \textsc{Find-Set}(u) \neq \textsc{Find-Set}(v) \textbf{then}
10: \hspace{2em} // i.e. \(A \cup \{e_i\}\) does not create a cycle
11: \hspace{2em} \(A \leftarrow A \cup \{e_i\}\)
12: \hspace{1em} \textsc{Set-Union}(u, v)
13: \hspace{1em} \textbf{end if}
14: \textbf{end for}

It is known that \(O(m)\) set operations take at most \(O(m \lg m)\).

Totally, Kruskal’s Algorithm takes \(O(m \lg m)\).
A Generic MST Algorithm

First we need a few definitions:

- Given a graph \( G = (V, E) \), and \( w : E \rightarrow \mathbb{R}^+ \)

- Suppose \( A \subseteq E \) is a set of edges contained in some MST of \( G \), then a new edge \( (u, v) \notin A \) is safe for \( A \) if \( A \cup \{(u, v)\} \) is also contained in some MST of \( G \).

\[
\text{GENERIC-MST}(G, w)
\]

1: \( A \leftarrow \emptyset \)
2: \textbf{while} \( A \) is not yet a spanning tree \textbf{do}
3: \hspace{1em} find \( (u, v) \) safe for \( A \)
4: \hspace{1em} \( A \leftarrow A \cup \{(u, v)\} \)
5: \hspace{1em} end \textbf{while}

Need a way to find a safe edge for \( A \)
How to find a safe edge

- A cut $(S, V - S)$ of $G$ is a partition of $G$, i.e. $S \subseteq V$.
- $(u, v)$ crosses the cut $(S, V - S)$ if $u \in S$, $v \in V - S$, or vice versa.
- A cut $(S, V - S)$ respects a set $A$ of edges if no edge in $A$ crosses $(S, V - S)$.

**Theorem 6.** Let $A$ be a subset of edges of some minimum spanning tree $T$ of $G$.

Let $(S, V - S)$ be any cut respecting $A$.

Let $(u, v)$ be an edge of $G$ crossing $(S, V - S)$ with minimum weight among all crossing edges.

Then, $(u, v)$ is safe for $A$. 
Prim’s Algorithm

Kruskal’s algorithm was a special case of the generic-MST
Prim’s Algorithm is also a special case: start growing the spanning tree out.

Running time: $O(|E| \lg |V|)$. Pseudo-code: please read the textbook.
Concluding notes

• There is a vast literature on matroid theory

• Some study it as part of poset theory

• Others study it as part of combinatorial optimization

• Key works
  – Whitney (1935) defined matroids from linear algebraic structures [Ever wondered why the independent sets were called “independent sets”?
  – Edmonds (1967) [in a conference] realized that Kruskal’s algorithm can be casted in terms of matroids