Advanced II September 8, 2025

Lecture 1: When and why does GREEDY work?

Notes by Ola Svensson[1]

1 Introduction

- Welcome
- Michael Kapralov (*michael.kapralov@epfl.ch*)
- CS-450 staff email: *CS-450-staff@epfl.ch*
- Join the moodle page!
- Show cool introductory slides

1.1 Grading

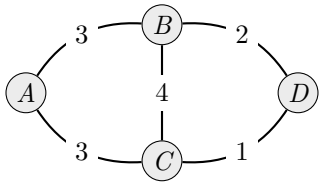
- 2 homeworks [30%] (please form groups with up to three students)
- Midterm exam [30%]
- Final exam [40%]

2 Warmup: Greedy Algorithm for Max Weight Spanning Trees

In this lecture, we will better understand perhaps the easiest of algorithms: always select the best available option in a greedy way. We first describe the algorithm when considering the spanning tree problem or rather the maximum weight forest problem. First recall the definition of the maximum weight [2] spanning tree problem:

Definition 1 *Given a connected undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$, find a spanning tree $T \subseteq E$ of maximum total weight $w(T) := \sum_{e \in T} w(e)$.*

Example 1 *The following undirected edge-weighted graph has two maximum spanning trees: $\{\{A, B\}, \{B, C\}, \{B, D\}\}$ and $\{\{A, C\}, \{B, C\}, \{B, D\}\}$.*



We are now going to describe the most basic greedy algorithm there is: repeatedly add the edge of largest weight that does not create a cycle. When considering the spanning tree problem this is known as Kruskal's algorithm (named after its inventor).

[1] **Disclaimer:** These notes were written as notes for the lecturer. They have not been peer-reviewed and may contain inconsistent notation, typos, and omit citations of relevant works.

[2] It is perhaps more common to consider the minimum spanning tree problem (which is equivalent by replacing each weight $w(e)$ by $-w(e)$). Here we consider the maximum weight spanning tree problem for the purpose of generalizing it to matroids.

GREEDY(G, w):

Input: A connected undirected graph $G = (V, E)$ and weights $(w_e)_{e \in E}$.
Output: A maximum weight spanning tree S.

1. Sort and relabel the edges so that $w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_{|E|}}$.
2. $S \leftarrow \emptyset$.
3. **for** $i = 1$ **to** $|E|$:
4. **if** $S + e_i$ is acyclic **then** $S \leftarrow S + e_i$.
5. **return** S.

Runtime analysis. Steps 3-4 can be implemented in almost linear time using the UNION-FIND (also called DISJOINT-SET) data structure that I hope you have seen during your Bachelor studies. The running time is thus dominated by the sorting of Step 1. Using e.g. Merge-Sort this step runs in time $\Theta(|E| \log |E|)$. The total running time is thus $\Theta(|E| \log |E|)$.

Why does it work? The correctness of the algorithm follows from the following lemma.

Lemma 2 GREEDY *returns a maximum-weight spanning tree.*

Proof Suppose not. Let $S = \{s_1, s_2, \ldots, s_{n-1}\}$ be the set (spanning tree) returned by the algorithm and suppose that a solution T has a higher weight, where $T = \{t_1, t_2, \ldots, t_{n-1}\}$ (indexed in decreasing weight). Let p be the first index such that $w(t_p) > w(s_p)$. Let $A = \{t_1, \ldots, t_p\}$ and $B = \{s_1, \ldots, s_{p-1}\}$.

Now we have the following key property of acyclic graphs.

Key property: As $|A| > |B|$, there exists $e \in A \setminus B$ such that $B + e$ is acyclic.

The key property follows from that an acyclic graph with k edges has $n - k$ components[3]. Thus the graph (V, A) has fewer components than (V, B) and so at least one edge $e \in A$ must connect two different components of (V, B). It follows that $e \notin B$ and that $B + e$ is acyclic.

Having proved the key property, we conclude as follows. Since $w(e) \geq w(t_p) > w(s_p)$, e should have been selected when it was considered.

To be more precise and detailed, when e was considered, the greedy algorithm checked whether e could be added to the current set at the time, say B'. But since $B' \subseteq B$, adding e to B' would have resulted in an acyclic graph (**a subset of acyclic edges is also acyclic**) since its addition to B results in an acyclic graph.

This gives a contradiction and completes the proof. ■

3 Matroids: The exact set of problems for which basic greedy algorithm works

Note that in the correctness of the GREEDY algorithm for max-weight spanning trees (Lemma 2) we used two properties of acyclic graphs: (i) a subset of acyclic edges is acyclic and (ii) for two acyclic edge sets A, B on the same vertex-set with $|A| > |B|$ there is $e \in A \setminus B$ such that $B + e$ is acyclic.

Matroids are defined to satisfy these two properties and we will see that they define the set of problems for which the basic greedy algorithm works. It also generalizes the notion of linear independence in matrices. There are many equivalent definitions. We use the one that focus on its independent sets.

Definition 3 *A matroid $M = (E, \mathcal{I})$ is defined on a finite ground set E and a family $\mathcal{I}$ of subsets of E that are called independent sets satisfying two axioms:*

[3]It is a good exercise to prove this using induction!

(I_1) *if* $X \subseteq Y$ *and* $Y \in \mathcal{I}$ *then* $X \in \mathcal{I}$.

(I_2) *if* $X \in \mathcal{I}$ *and* $Y \in \mathcal{I}$ *and* $|Y| > |X|$ *then* $\exists e \in Y \setminus X : X \cup \{e\} \in \mathcal{I}$.

Remarks:

- Letting E be the edges of a graph and $\mathcal{I} = \{F \subseteq E : \ F \text{ is acyclic}\}$ defines a so-called *graphic matroid*. Note that it satisfies (I_1) since the subset of an acyclic set of edges is acyclic and it satisfies (I_2) due to the key property used in the proof of Lemma 2. We see more examples of matroids in Section 3.1.
- The family $\mathcal{I}$ may be exponential in the size of the ground set E (as is the case for example for graphic matroids of complete graphs). One therefore often assumes that $\mathcal{I}$ is given implicitly by having a membership oracle: an algorithm that given $I \subseteq E$ efficiently answers whether $I \in \mathcal{I}$.
- The second axiom implies that every *maximal* independent set is of maximum cardinality. In other words, all maximal independent sets have the same cardinality. A maximal cardinality set is called a *base* of the matroid.

With the more abstract notation of matroids, the basic greedy algorithm becomes

GREEDY(M, w):
Input: A matroid $M = (E, \mathcal{I})$ and weights $(w_e)_{e \in E}$.
Output: A maximum weight base S.

1. Sort and relabel the elements so that $w_1 \geq w_2 \geq \cdots \geq w_{|E|}$.
2. $S \leftarrow \emptyset$.
3. **for** $i = 1$ **to** $|E|$:
4. **if** $S + i \in \mathcal{I}$ **then** $S \leftarrow S + i$.
5. **return** S.

The concept of matroids was defined so as to enable the correctness analysis of the basic greedy algorithm. Perhaps more surprisingly, it is if and only if.

Theorem 4 (Rado'57/Gale'68/Edmonds'71) *For any ground set* $E = \{1, 2, \ldots, n\}$*, and a family of subsets* $\mathcal{I}$*,* GREEDY *finds a maximum weight base*[4] *for any set of weights* $w : E \to \mathbb{R}$ *if and only if* $M = (E, \mathcal{I})$ *is a matroid.*

The if direction ($\Leftarrow$ part) follows from Lemma 2. For the only if direction we have the following claim.

Claim 5 *($\Rightarrow$ part) Suppose* $(E, \mathcal{I})$ *is not a matroid. There exists an assignment of weights* $w : E \to \mathbb{R}$ *so that* GREEDY *does not return a maximum weight base.*

Proof If $(E, \mathcal{I})$ is not a matroid, then it violates at least one of the two axioms.

First suppose $\mathcal{I}$ is not a downward-closed family of sets, i.e., it violates (I_1). Therefore, there exist two sets $S \subset T$ such that $S \notin \mathcal{I}$ and $T \in \mathcal{I}$. Consider the following weights:

$$w_i = \begin{cases} 2 & i \in S \\ 1 & i \in T \setminus S \\ 0 & \text{otherwise.} \end{cases}$$

[4]With a base, we mean a maximal set with respect to inclusion.

By the weight assignment, the algorithm first considers the elements of S, then the elements of T, and then the rest of the elements. Suppose the algorithm selects a subset S_1 of S. Since $S \notin \mathcal{I}$, S_1 is a strict subset. Out of the remaining elements the algorithm can select at most $T \setminus S$. So the weight of the independent set that the algorithm returns is at most $2|S_1| + |T \setminus S|$ which is less than $w(T)$ so it is not a maximum weight independent set.

Second suppose that the extension axiom is violated (but downwardness is satisfied). In particular, let $S, T \in \mathcal{I}$ be two independent sets such that $|S| < |T|$, and for all $i \in T \setminus S$, $S + i \notin \mathcal{I}$. Now use the following weights

$$w_i = \begin{cases} 1 + \frac{1}{2|S|} & i \in S \\ 1 & i \in T \setminus S \\ 0 & \text{otherwise.} \end{cases}$$

Because of downwardness the algorithm would select all elements in S and return an independent set of value $|S| + 1/2$ where as the optimal set would have value at least $|T| > |S| + 1/2$. ■

3.1 Examples of matroids

We already saw the graphic matroids. Other basic matroids are as follows:

3.1.1 k-Uniform matroid

A matroid $M = (E, \mathcal{I})$ is k-Uniform if $\mathcal{I}$ satisfies:

$$\mathcal{I} = \{X \subseteq E : |X| \leq k\}.$$

3.1.2 Partition matroid

A matroid $M = (E, \mathcal{I})$ is a partition matroid if E is partitioned into *disjoint* sets $E_1, E_2, ..., E_\ell$ and

$$\mathcal{I} = \{X \subseteq E : |E_i \cap X| \leq k_i \text{ for } i = 1, 2, ..., \ell\}.$$

3.1.3 Linear matroid

A matroid $M = (E, \mathcal{I})$ is a linear matroid when it is defined from a matrix A. Let E be the index set of the columns and for $X \subseteq E$ let A_X be the matrix consisting of the columns indexed by X. Define $\mathcal{I}$ by

$$\mathcal{I} = \{X \subseteq E : rank(A_X) = |X|\}.$$

3.1.4 Truncated matroid

A truncated matroid $M_k = (E, \mathcal{I}_k)$ is defined from a matroid $M = (E, \mathcal{I})$ such that

$$\mathcal{I}_k = \{X \in \mathcal{I} : |X| \leq k\}.$$

It is quite easy to verify that the axioms still hold for M_k, as $X \in \mathcal{I}_k$ implies $X \in \mathcal{I}$ for all $X \subseteq E$.
(I_1) holds because $B \in \mathcal{I}_k$ means that $|B| \leq k$ and $A \subseteq B$ thus means $|A| \leq k$ as well. We know that M is a matroid so I_1 holds for M, which implies $A \in \mathcal{I}$. We conclude that $A \in \mathcal{I}_k$
The same reasoning can verify I_2: if $A, B \in \mathcal{I}_k$ and $|B| > |A|$, then $|B| \leq k$ and $|A| \leq k - 1$. We know that I_2 holds for M, so the inclusion of $\mathcal{I}_k$ in $\mathcal{I}$ tells us that $\exists e \in B \setminus A$ such that $A + e \in \mathcal{I}$. The fact that $|A + e| \leq k - 1 + 1 = k$ allows us to conclude.