Algorithms II - CS450

Jacopo Philip Moretti

Fall 2024

Introduction

Hello ! I'm jack `:3`. I am a student who likes to retype their notes in LaTeX. If this can be of any use to you, I'm happy ! Do shoot me an email if you find any mistakes or unclear passages. Elements of notation or other global considerations will be added here as the course goes along. Do refer back to this if something weird happens later !

In the meantime, enjoy the course and the ride `:3`.

Notation

- ▷ Let $S \subset A$ a subset over a given support set. Then, for $i \in A$, we denote the following: $S + i$. More generally, when two sets A, B are disjoint, their union will be represented as $A + B =: A \cup B$, as a shorthand.

Contents

Chapter 1

Greed is good, actually (sometimes (when?))

1 Warmup: Maximal Spanning Tree (MST)

Week 1
Lecture 1

Definition 1 *Given a connected graph $G = (V, E)$, and a weight function $w : E \to \mathbb{R}$, we say that a tree $T \subseteq E$ is a* maximal spanning tree *if it maximizes the sum of the weights of its edges.*

MSTs are ubiquitous in algorithms! To find them, we use *Kruskal's algorithm.*

Algorithm 1 Kruskal's Algorithm

```
   procedure GREEDY($(G, w)$):
1: Sort and label the edges by their weigths:

          $w_{e_1} \geqslant w_{e_2} \geqslant ... \geqslant w_{e_{|E|}}$

2: $S \leftarrow \emptyset$
3: for $i \leftarrow 1$ to $|E|$ do
4:     if $S + e_i$ is acyclic then
5:         $S \leftarrow S + e_i$
6:     end if
7: end for
8: return $S$
```

Where lines 3 to 7 make up the "greedy" part of the algorithm. This is the "natural" approach, but we can prove that it *actually* works !

Theorem 1 *Procedure* Greedy *finds a Maximal Spanning Tree for any graph $G = (V, E)$ and weight function w passed as input.*

Proof. Suppose it's not the case, and let $S = \{s_1, ..., s_n\}$ be its output. We suppose that there exists a tree $T = \{t_1, ..., t_n\}$ such that $w(T) > w(S)$. We also sort the edges in S and T in such a way that $w(t_i) > w(t_j)$ if and only if $i > j$, and similar for S.

We know that there exist indices where $w(t_p) > w(s_p)$, and we denote p the smallest of such indices. We let $A = \{t_1, ..., t_p\}$, and $B = \{s_1, ..., s_p\}$.

We point out a key property for these two sets:

Key property: $\exists e \in A \setminus B : B + e$ is acyclic.

This is because an acyclic graph with k edges always has $n - k$ connected components.

We know that

$$w(e) \geqslant w(t_p) \geqslant w(s_p)$$

Therefore, when e was considered by our algorithm, it "held" in state $B' \subseteq B$ (we know it was before because $w(e) > w(s_p)$). However, if $B + e$ is acyclic, the we know that $B' + e$ is acyclic as well, due to the downward closedness of acyclicity. Therefore, Greedy should have chosen it, which contradicts our assumption.

□

2 Matroids

2.1 Definitions

It's somewhat surprising that our naive, greedy approach to algorithm design gives us an optimal algorithm. Is it possible to generalize this result to rigorously define the algorithms where the greedy answer is optimal? To do this, we can define what a matroid is.

Definition 2 *A tuple $M = (E, \mathcal{I})$, with E a ground set, and $\mathcal{I}$ a family of subsets of E, is a* matroid *if and only if, for $X, Y \subseteq E$:*

(I_1) $(X \subseteq Y \wedge Y \in \mathcal{I}) \Rightarrow X \in \mathcal{I}$

(I_2) $(X \in \mathcal{I} \wedge Y \in \mathcal{I} \wedge |Y| > |X|) \Rightarrow \exists e \ \in X \setminus Y : X + e \in \mathcal{I}$

We call independent sets the elements of $\mathcal{I}$, and we say that a maximal independent set is an independent set that isn't the proper subset of any other independent set.[1]*. Maximal independent sets are also called bases.*

The reader can convince themselves that the structure $\tilde{M} = (E, \mathcal{I})$, where E is the set of edges of a graph G, and $\mathcal{I} = \{F \subseteq E : F \text{ is acyclic}\}$ is a matroid. We can now try to redefine Kruskal's algorithm on matroids !

[1] One intuition for this is seeing the MISs as the roots (or leafs) of the tree structure generated by axiom (I_1), meaning they're maximal with respect to inclusion

Algorithm 2 Kruskal's Algorithm on Matroids

```
   procedure MatroidGreedy($(G, w)$):
1: Sort and label the *ground set elements* by their weigths:
       $w_{e_1} \geqslant w_{e_2} \geqslant \dots \geqslant w_{e_{|E|}}$
2: $S \leftarrow \emptyset$
3: for $i \leftarrow 1$ to $|E|$ do
4:     if $S + e_i \in \mathcal{I}$ then
5:         $S \leftarrow S + e_i$
6:     end if
7: end for
8: return $S$
```

The idea remains the exact same ! We just replace the condition on acyclicity with a condition on the belonging to $\mathcal{I}$. Does this still work?

Theorem 2 *For any ground set E, and $\mathcal{I}$ a family of subsets of E,* MatroidGreedy *finds a maximal weight base of M for every $w : E \to \mathbb{R}$ if and only if $M = (E, \mathcal{I})$ is a matroid.*

Proof. Since we have a bi-implication, we will proceed by proving both directions separately:

($\Leftarrow$) Similar as the previous proof, but swapping out the acyclicity with independence.

($\Rightarrow$) For this direction, a supportive claim will be of use to us.

Claim 1 *Suppose $(E, \mathcal{I})$ is not a matrioid. Then, there exists a weight function $w : E \to \mathbb{R}$ such that* MatroidGreedy *does not return the maximum weight base.*

Suppose M is not a matroid. Two cases present:

- Suppose M violates axiom (I_1) (downward closedness). Then there exist two subsets $S, T \subseteq E$ such that $S \subset T, T \in \mathcal{I} \wedge S \notin \mathcal{I}$. We can define weight a weight function $\tilde{w}$ such that:

$$\tilde{w}(e) = \begin{cases} 2, & e \in S \\ 1, & e \in T \setminus S \\ 0, & e \notin T \end{cases}$$

  Under these conditions, the algorithm will first consider all elements of S, since they have a higher weight. Among them, it will take a subset S', which will necessarily be a proper subset since $S \notin \mathcal{I}$. After S', the algorithm will at most pick $T \setminus S$. Therefore, the weight has bounds:

$$C(\text{MatroidGreedy}) \leqslant |T \setminus S| + 2|S'| < |T \setminus S| + 2|S| < w(T)$$

  Which means our algorithm didn't return the maximum weight base.

- Suppose now axiom (I_1) holds, but axiom (I_2) (extension) is violated. This means that there exists two independent sets $S, T \in \mathcal{I}$ such that $|S| < |T|$ and for any element $i \in T$, $S + i$ is not an independent set. We claim that function $\tilde{w}$:

$$\tilde{w}(e) = \begin{cases} 1 + \frac{1}{2|S|}, & e \in S \\ 1, & e \in T \setminus S \\ 0, & e \notin T \end{cases}$$

is such that MatroidGreedy doesn't find the maximal base. Because downwardness, Greedy will select all of S, which yields a base of weight $w(S) = |S|\left(1 + \frac{1}{2|S|}\right) = |S| + \frac{1}{2} < w(T)$ since $|T| > |S|$.

Week 1
Lecture 2

2.2 Matroid intersections

We mostly consider the problem of the form: given a matroid $M = (E, \mathcal{I})$ and a weight function $w : E \to \mathbb{R}$, find

$$\max_{F \in \mathcal{I}} w(F)$$

More often than not, this kind of problem can't be solved directly, but we can find some new techniques to help us tackle the issue efficiently. But first:

Examples of matroids. Recall that a matroid is a pair $M = (E, \mathcal{I})$ where E is the *ground set* and $\mathcal{I}$ is a family of subsets of E such that the two axioms hold:

(I_1) $X \subseteq Y \wedge Y \in \mathcal{I} \Rightarrow X \in \mathcal{I}$

(I_2) $X, Y \in \mathcal{I} \wedge |Y| > |X| \Rightarrow \exists\, e \in Y \setminus X : X + e \in \mathcal{I}$

Let us consider a few examples of constructions that have matroid structure.

- *graphic matroid*: given $G = (V, E)$ a graph, we know that

$$M = (E, \mathcal{I}) \text{ where } \mathcal{I} = \{F \subseteq E | F \text{ is acyclic}\}$$

- *k-uniform matroid*: on an arbitrary ground set E, and with $k \in \mathbb{N}$, we define $M = (E, \mathcal{I})$ where

$$\mathcal{I} = \{F \subseteq E || F| \leqslant k\}$$

- *partition matroid*: $M = (E, \mathcal{I})$, with $E = \bigcup_i E_i$ with E_i a disjoint family of sets, and $\mathcal{I}$ such that

$$\mathcal{I} = \{F \subseteq E || F \cap E_i| \leqslant k_i \forall i\}$$

- *linear matroid*: let A a matrix, and define E the index set of its columns. For $X \subseteq E$, denote A_X the matrix consisting of the columns indexed by X. Then, $M = (E, \mathcal{I})$ is a matroid, with

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

For an example of uses of a linear matroid, refer back to Annex A.

- ▷ *truncated matroid*: Let $M = (E, \mathcal{I})$ be a matroid. Then from it we can define a truncated matroid $M_k = (E, \mathcal{I}_k)$, with $k \in \mathbb{N}$, such that

$$\mathcal{I} = \{X \in \mathcal{I} | |X| \leqslant k\}$$

Verifying the axioms for these example matroids is a very good exercise that we leave to the reader.

To define matroid intersections, we define a problem they're of use in:

Definition 3 *Given a graph $G = (V, E)$, find a matching of maximal size. Recall that $M \subseteq E$ is a matching if and only if every vertex is touched only once:*

$$\forall\ v \in V, |\{e \in M : v \in e\} \leqslant 1|$$

We can try to define a matroid to find the matchings efficiently using Kruskal's Greedy. We tentatively define $\tilde{M} = (E, \tilde{\mathcal{I}})$, where

$$\tilde{\mathcal{I}} = \{M \subseteq E : M \text{ is a matching}\}$$

and we realize pretty fast that this doesn't work, as we can easily construct counterexamples to (I_2).

It's safe to assume that we can't build this kind of metroid, but the answer might be more exotic...

Definition 4 *Given two matroids $M_1 = (E, \mathcal{I}_1), M_2 = (E, \mathcal{I}_2)$ their* intersection *is defined as :*

$$M_1 \cap M_2 := (E, \mathcal{I}_1 \cap \mathcal{I}_2)$$

Theorem 3 (Edmonds, Lawler, 70s) *There exists an efficient algorithm to find the max weight independent set between two matroids.*

No proof ! But we can convince ourselves of this by looking at a few problems.

Matroid intersections. Here are a few problems that are solved by matroid intersections:

- ▷ *bipartite matching*: Consider a graph $G = (V, E)$ that admits the partition $A + B = V$ such that every edge in E is between a node in A and another in B. We can define this problem in terms of the intersection $M_A \cap M_B$, where

$$\mathcal{I}_A = \{F \subseteq E : |F \cap \delta(a)| \leqslant 1 \forall\ a \in A\}$$

and similarly for B. In this expression, $\delta : V \to \mathcal{P}(E)$ is the function returning all the proximal edges in E to the input vertex.

- ▷ *colorful spanning trees*: Let $G = (V, E)$ be a graph, and define over its edges the function `color` $: E \to \mathbb{C}$ where $\mathbb{C}$ is the set of possible colors. We say that a tree T is *colorful* if every edge in T is a different color. This problem can be studied as the intersection of the graph matroid of G, $M_1 = (E, I_1)$, and of the partition matroid of the color classes, defined as $M_2 = (\bigcup E_i, I_2)$

where $E_i = \{\text{all edges of color i}\}$ and

$$I_2 = \{F \subseteq E : |F \cap E_i| \leqslant 1 \ \forall i\}$$

▷ *arborescences*: Consider a digraph $D = (V, E)$ and a root $r \in V$. Finding arborescences (aka spanning trees directed away from r) is equivalent to finding maximal bases in the intersection matroid between

- M_1, the graphical matroid of G, which is D but made undirected by making all edges bidirectional.
- and M_2, the partition matroid such that

$$\mathcal{I}_i = \{F \subseteq E : |F \cap \delta^-(v)| \leqslant 1 \ \forall v \in V\}$$

where $\delta^-(v)$ returns the set of incoming edges for all vertices that aren't r, and the empty set for r.

2.3 Case study on matroid intersection algorithms

We consider the algorithm for maximum cardinality bipartite matching. Recall that a *path* is a sequence of vertices $(v_0, v_1, ..., v_n)$ where $(v_i, v_{i+1}) \in E \ \forall i$.

Definition 5 *An* alternating path *with respect to a matching M is a path that alternates between M and $E \setminus M$.*

An augmenting path *with respect to a matching M is an alternating path that starts and ends on an unmatched vertex.*

There exists a known algorithm to find augmenting paths:

Algorithm 3 Augmenting Path Algorithm

```
   procedure AUGMENT($G$):
1: $M \leftarrow \emptyset$
2: while $\exists$ augmenting path $P$ with respect to $M$ do
3:     $M \leftarrow M\Delta P = (M \setminus P) + (P \setminus M)$
4: end while
5: return $M$
```

This motivates:

Theorem 4 *A matching M is a max cardinality matching if and only if there doesn't exist an augmenting path with respect to M.*

Annex A : Linear Matroid Application