0.1 Generic greedy algorithm

The greedy algorithm is an algorithmic paradigm. It does not always provide a solution in general, not least an optimal solution. As such the greedy algorithm is a description of a problem solving heuristic. It says that when trying to solve a problem we should choose the local optimum at each iteration in the hope of finding a global optimum. In this section, we will introduce the generic procedure that makes the greedy algorithm and showing how it can be modified in a variety of ways, including for compatibility with matroids.

Algorithm 1 Greedy algorithm[**?**]

```
Let $(E, \mathscr{S})$ be an independence system and $\omega : E \longrightarrow \mathbb{R}^+$
1: procedure GREEDY$(E, \mathscr{S}, \omega, T)$
2:     order the elements of $E$ according to their weight
3:     $E = \{e_1, ..., e_m\}$ with $\omega(e_1) \geq \omega(e_2) \geq ... \geq \omega(e_m)$
4:     $T \leftarrow \emptyset$
5:     for $k = 1$ to $m$ do
6:         if $T \cup \{e_k\} \in \mathscr{S}$ then
7:             append $e_k$ to $T$
```

We can see in the above pseudocode (taken from [**?**]) that this is a sequential iterative algorithm. Our data is kept in a list sorted by weight: heaviest to lightest. We then select the heaviest entry in our list at each iteration and append it to our initial empty variable T. By the end of the process we should have joined a certain number of the elements to our variable T. Where T at the time of termination is the greedy algorithm solution.
This is how the algorithm looks in the most abstract sense, later we will see it used as a way of solving the *minimal spanning tree* problem mentioned in *definition 3.11.*

0.2 Modified greedy for Matroids

Here, we will see how this algorithm corresponds to matroids. The greedy algorithm above, constructed a maximal weight element from our list called T. Depending on the structure of our system and the termination clause this element's cardinality can differ. It is our hope to show that when the greedy algorithm is applied to a matroid we get an optimal solution, meaning that we generate a base of the matroid. However, for now it is enough to show that the greedy algorithm always produces a solution in general. This is illustrated by the below pseudocode, which was adapted from Oxley's[**?**] description.

Algorithm 2 Greedy algorithm

The *greedy algorithm* for the pair $(\mathscr{I}, \omega)$ is as follows:

```
1: procedure GREEDY($E, \mathscr{I}, \omega$)
2:     Set $T \leftarrow \emptyset$ and $E = \{e_1, ..., e_m\}$
3:     while $\exists e \in E \setminus T$ such that $T \cup \{e\} \in \mathscr{I}$ do
4:         Choose such an element $e_{max}$ of maximum weight,
5:         let $T = T \cup \{e_{max}\}$
6:     return $T$
```

The procedure described above is as follows: Initialise the solution variable T as the emptyset (the empty set is always independent). We want to build the base of the matroid: B_G. At each iteration select the heaviest weighted element possible that is not already contained in T such that $T \cup \{e\} \in \mathscr{I}$ where e is the selected element, we want to repeat this procedure as many times as possible until there are no such elements left to choose. Then we return T, as T is now equal to the solution of the greedy algorithm, i.e B_G (the base generated by the greedy algorithm).

Remark. It is interesting to note that in this way we should find a maximal member B_{max} of $\mathscr{B}$ (the collection of bases of a matroid) and we will prove that this is certainly the case later in *theorem 5.2.* But if we negate this process in the following way we can use this exact procedure to find a minimal member B_{min} of $\mathscr{B}$. This works as follows:
Let $\omega : E \longrightarrow \mathbb{R}^-$ be the weight function, we want to find an independent set A whose weight is maximal, where

$$\omega(A) := \sum_{e \in A} |\omega(e)| \tag{1}$$

Due to the fact that our weights are now negative real numbers, finding the maximal element at each iteration corresponds to finding the value with the minimal absolute value. And so through completing the greedy algorithm process we should successfully find our minimal element B_{min} of $\mathscr{B}$.