0.1 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 gredy algorithm and showing how it can be modified in a variety of ways, including 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, 𝒮, ω, T)
2:     order the elements of E according to their weight
3:     E = {e_1, ..., e_m} with ω(e_1) ≥ ω(e_2) ≥ ... ≥ ω(e_m)
4:     T ← ∅
5:     for k = 1 to m do
6:         if T ∪ {e_k} ∈ 𝒮 then
7:             append e_k to T
```

We can see in the above pseudocode that this is a sequential iterative algorithm. Our data is kept in a list sorted by weight,going from heaviest to lightest and then sequentially selecting the heaviest entry in our list at each iteration. By selecting that element and then adding 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 varibale 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.*

First we will see how this 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. Which is illustrated by the below pseudocode.

Algorithm 2 Greedy algorithm

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

```
1: procedure GREEDY(𝓘, ω)
2:     Set x_0 = ∅ and j = 0
3:     if ∃e ∈ E \ x_j such that x ∪ {e} ∈ 𝓘 then
4:         Choose such an element e_{j+1} of maximum weight,
5:         let x_{j+1} = x_j ∪ {e_{j+1}} and
6:         GREEDY(𝓘, ω)
7:     else
8:         Let x_j = B_G
9:         return x_j
10:    j++
```

The procedure described above is as follows: Select the emptyset as your first element denoted x_0 as the empty set is always independent, we want to build the set x. Now as before order your elements from heaviest to lightest.
Then select the heaviest weighted element possible such that $x \cup \{e\} \in \mathscr{I}$ where e is the selected element. Otherwise we want to return x as x is then equal to the solution of the greedy algorithm, B_G (the base generated by the greedy algorithm). The element e should then be removed from further selection, and the process should be called recursively until there is no possible element that can be added to the set x without inducing a circuit.

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. 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 $\mathscr{B}$. Assuming that our algorithm is correct.