Greedy Exchange Argument Explored

Aditya Bhimalingam

August 31, 2025

1 Introduction

Greedy algorithms are often very intuitive. The idea is that at each step one makes the choice that looks best at the moment. In some problems this local optimality extends to global optimality. In this note I want to explore one such problem and explain why greedy works in a way that is both rigorous and easy to follow. I will focus on interval selection as a concrete example and then consider more general principles. Throughout I will explain my reasoning in a way that shows why each step is necessary.

Theorem 1.1. *The earliest-finish-time greedy algorithm returns a maximum-cardinality set of pairwise compatible activities.*

Algorithm: sort activities by finish time; scan and pick when $s_i \geq$ last finish.
Time: $O(n \log n)$ (sorting) + $O(n)$ (scan).

2 Problem Setup

Let $A = \{1, 2, \ldots, n\}$ be a set of activities. Each activity i has a start time s_i and a finish time f_i with $s_i < f_i$. Two activities i and j are compatible if $[s_i, f_i]$ and $[s_j, f_j]$ do not overlap. The objective is to select a subset of A of maximum size such that all selected activities are compatible. At first glance one might try all subsets but this is impractical for large n. A greedy approach offers a systematic way to choose. To avoid boundary issues we treat intervals as half-open: $[s_i, f_i]$, so an activity ending at time t does not conflict with one starting at t.

3 Greedy Algorithm

The natural greedy choice is to pick the activity that finishes earliest among the activities still compatible with the ones already chosen. Let G denote the greedy set. Formally $G = \{g_1, g_2, \ldots, g_k\}$ with finish times $f_{g_1} \leq f_{g_2} \leq \cdots \leq f_{g_k}$.

The construction is as follows

1. Sort all activities by finish time
2. Initialize an empty set S
3. Consider activities in order. If the current activity does not overlap with anything in S add it to S

4. At the end S is G

It is clear that each choice leaves as much room as possible for future activities. The question is whether this greedy solution is truly optimal.

4 Exchange Lemma

To justify the greedy choice we use the exchange lemma. Suppose O is an optimal set. Then we can exchange activities from O with activities from G one by one without ever reducing the total number of activities until O becomes G. This shows that G is optimal.

Lemma 4.1 (Exchange Lemma)**.** *Let O be an optimal solution and G the greedy solution. Then O can be transformed into G by a sequence of exchanges such that at every step the solution remains optimal.*

Proof. Sort O and G by finish times. Consider g_1 and o_1. Since g_1 finishes no later than o_1, we can replace o_1 by g_1 without introducing conflicts. In this case one could clearly see that the number of activities remains the same.

Assume by induction that the first i activities of G have been exchanged into O. Let g_{i+1} be the next greedy activity and o_{i+1} the corresponding optimal activity. Since g_{i+1} finishes no later than o_{i+1} and is compatible with all previously chosen activities we can replace o_{i+1} with g_{i+1}. Repeating this we eventually have $O = G$. In this case one could clearly see that the set must be equivalent in terms of the number of elements to the original optimal set, therefore proving correctness. □

5 Worked Numerical Example

Consider five activities with intervals as follows

Activity	Start	Finish
1	1	4
2	3	5
3	0	6
4	5	7
5	8	9

Table 1: Activity start and finish times for the worked example

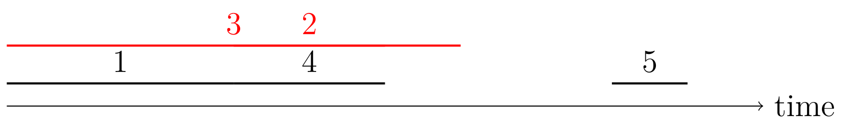

Figure 1: Black intervals are greedy choices; red intervals show other activities in a possible optimal set before exchange.

We sort by finish times giving order $1, 2, 3, 4, 5$. The greedy algorithm selects 1 first. Activity 2 overlaps with 1 so it is skipped. Activity 3 overlaps with 1 so it is skipped. Activity 4 is compatible and is chosen. Activity 5 is compatible and is chosen. Hence $G = \{1, 4, 5\}$. One can verify that any optimal set also contains three activities. The exchange lemma can be applied to show that G is equivalent to any optimal solution.

6 Weighted Intervals

Suppose each activity has a weight w_i. If we apply the same greedy algorithm by finish time it can fail. For example let $a_1 = [0, 2]$ with $w_1 = 10$, $a_2 = [0, 1]$ with $w_2 = 1$, $a_3 = [1, 2]$ with $w_3 = 1$. The greedy algorithm picks a_2 and a_3 giving total weight 2. But clearly a_1 alone gives weight 10. In this case one sees that naive greedy does not work and more careful reasoning such as dynamic programming is needed.

7 Matroid Generalization

The principle behind the exchange lemma can be generalized. Any set system that satisfies the matroid exchange property allows greedy algorithms to produce optimal solutions. Formally if I and J are independent sets with $|I| < |J|$ there exists e in $J \setminus I$ such that $I \cup \{e\}$ is independent. Activity selection is a simple example. One could try to find other combinatorial structures where this principle applies.

8 Conclusion

The greedy algorithm picks locally optimal activities. By the exchange lemma one can clearly see that this local optimality extends globally. In this case the greedy set must be equivalent in number of elements to any optimal set therefore proving correctness. Weighted intervals illustrate the limitations of naive greedy strategies. The matroid abstraction shows the general principle. Going through these examples one can see both why greedy works in some settings and why more care is needed in others. By following this reasoning step by step one can develop intuition for when greedy algorithms succeed and when they fail.

9 Remarks

The exchange argument resembles the matroid exchange property, but interval scheduling is not a matroid in general. In particular, when activities have weights, the greedy strategy fails and a dynamic programming approach is required.