1. Activity Selection Problem

One problem, which has a very nice (correct) greedy algorithm, is the Activity Selection Problem. In this problem, we have a number of activities. Your goal is to choose a subset of the activies to participate in. Each activity has a start time and end time, and you can’t participate in multiple activities at once. Thus, given *n* activities *a*1*,a*2*,...,an* where *ai* has start time *si*, finish time *fi*, we want to find a maximum set of non-conflicting activities.

The activity selection problem has many applications, most notably in scheduling jobs to run on a single machine.

# Optimal Substructure

Let’s start by considering a subset of the activities. In particular, we’ll be interested in considering the set of activities *Si,j* the start after activity *ai* finishes and end before activity *aj* starts. That is, *Si,j* = {*ak*|*fi* ≤ *sk,fk* ≤ *sj*}. We can participate in these activities between *ai* and *aj*. Let *Ai,j* be a maximum subset of non-conflicting activities from the subset *Si,j*. Suppose some *ak* ∈ *Ai,j*, then we can break down the optimal subsolution *Ai,j* as follows

|*Ai,j*| = 1 + |*Ai,k*| + |*Ak,j*|

where *Ai,k* is the best set for *Si,k* (before *ak*), and *Ak,j* is the best set for after *ak*. Another way of interpreting this expression is to say “once we place *ak* in our optimal set, we can only consider optimal solutions to subproblems that do not conflict with *ak*.”

Thus, we can immediately come up with a recurrence that allows us to come up with a dynamic programming algorithm to solve the problem.



This problem requires us to fill in a table of size *n*2, so the dynamic programming algorithm will run in Ω(*n*2) time. The actual runtime is *O*(*n*3) since filling in a single entry might take *O*(*n*) time.

But we can do better! We will show that we only need to consider the *ak* with the smallest finishing time, which immediately allows us to search for the optimal activity selection in linear time.

**Claim 1.** *For each Si,j, there is an optimal solution Ai,j containing ak* ∈ *Si,j of minimum finishing time fk.*

Note that if the claim is true, when *fk* is minimum, then *Ai,k* is empty, as no activities can finish before *ak*; thus, our optimal solution only depends on one other subproblem *Ak,j* (giving us a linear time algorithm). Here, we prove the claim.

*Proof.* Let *ak* be the activity of minimum finishing time in *Si,j*. Let *Ai,j* be some maximum set of nonconflicting activities. Consider *A*0*i,j* = *Ai,j* \ {*al*} ∪ {*ak*} where *al* is the activity of minimum finishing time in *Ai,j*. It’s clear that

|*A*0*i,j*| = |*Ai,j*|. We need to show that *A*0*i,j* does not have conflicting activities. We know *al* ∈ *Ai,j* ⊂ *Si,j*. This implies *fl*

≥ *fk*, since *ak* has the min finishing time in *Si,j*.

All *at* ∈ *Ai,j* \ {*al*} don’t conflict with *al*, which means that *st* ≥ *fl*, which means that *st* ≥ *fk*, so this means that no activity in *Ai,j* \ {*al*} can conflict with *ak*. Thus, is an optimal solution.

Due to the above claim, the expression for *Ai,j* from before simplifies to the following expression in terms of

*ak* ⊆ *Si,j*, the activity with minimum finishing time *fk*.

|*Ai,j*| = 1 + |*Ak,j*|

*Ai,j* = *Ak,j* ∪ {*ak*}

Algorithm Greedy-AS assumes that the activities are presorted in nondecreasing order of their finishing time, so that if *i < j*, *fi* ≤ *fj*.

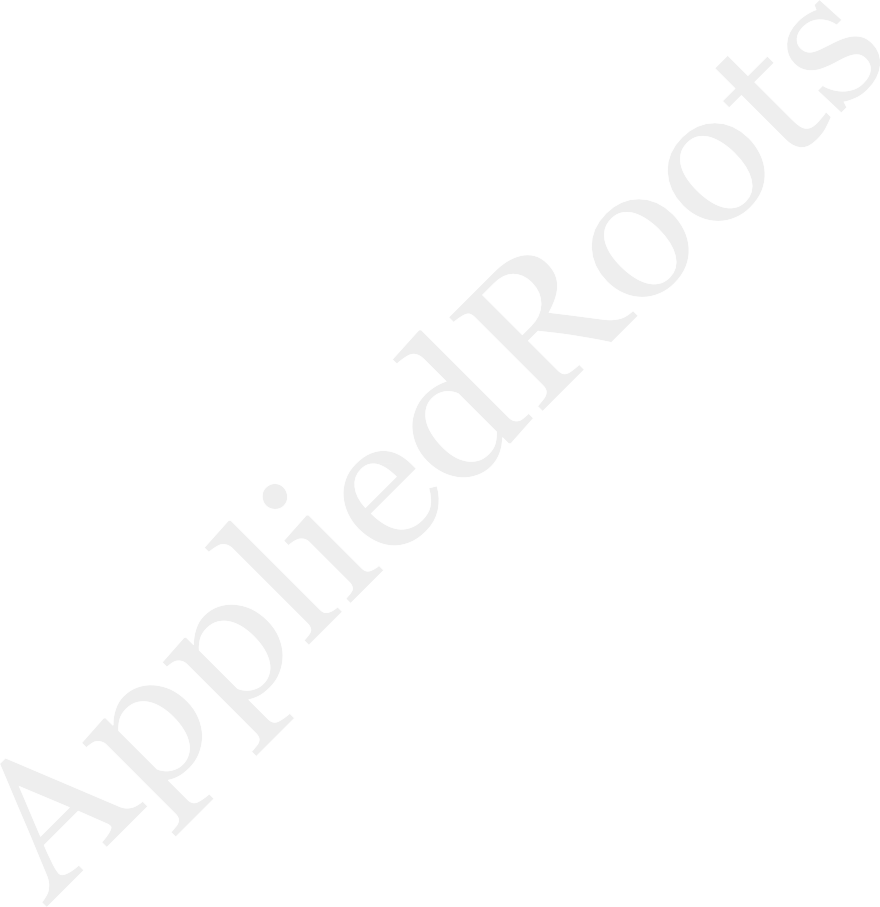
**Algorithm 1:** Greedy-AS(*a*)

*A* ← {*a*1} // activity of min *fi k* ← 1

**for** *m* =2 → *n* **do**

**if** *sm* ≥ *f k* **then**

//*am* startsafterlastacitivityin *A A* ← *A* ∪{ *am* }

*k* ← *m*

# return A

By the above claim, this algorithm will produce a legal, optimal solution via a greedy selection of activities. The algorithm does a single pass over the activities, and thus only requires *O*(*n*) time – a dramatic improvement from the trivial dynamic programming solution. If the algorithm also needed to sort the activities by *fi*, then its runtime would be *O*(*n*log*n*) which is still better than the original dynamic programming solution.