Greedy Algorithms: Activity Selection

Algorithms for Optimizing Activity Selection

Note: In this algorithm we assume that all activities in the input array are stored in ascending order of the activity's end time. If this is not the case, the algorithms can be modified in one of two ways:

- **sort** all activities in ascending order of the activity's end time, then call the algorithm.
- **call, on each step,** a function that finds, among the remaining activities the one that ends at the earliest time.

Iterative Algorithm

```
ALGORITHM ActivitySelection(N, A[1..N][1..2])
begin
  S[0..N];     //initialize output array
  currentEndTime ← −1;     //initialize current end time counter
  i ← 1;
  while i ≤ N do
        //check to see if current activity can be scheduled
    if A[i][1] ≥ currentEndTime then
      S[i] ← 1;     //select it
      S[0] ← S[0] + 1;     //update scheduled activity count
      currentEndTime← A[i][2];     //update current end time marker
    endif
    i ← i + 1;
  endwhile
 return S[0..N];
end
```

Recursive Algorithm

Note: The ActivitySelectionRecursive() algorithm adds a pair of input parameters. One, K, is the current index in the array $A[][]$ of activities. The other, StartTime is the earliest possible start time for the activities at the current moment. The initial call to this algorithm will set $K = 1$ and StartTime$= 0$.

Note: Again, this algorithm works only for arrays A that are sorted in ascending order by their second component.

```
ALGORITHM ActivitySelectionRecursive(N, A[K..N][1..2],K, StartTime)
      // K is the "current" index in the array of activities.
      //StartTime is the earliest allowed start time
begin
  if A[K][1]≥StartTime then     //determine if A[K] can be scheduled
    decision ← 1;     //A[K] will be selected
    StartTime← A[K][2];     //establish new start time
  else
    decision ← 0;     //A[K] will NOT be selected
  endif
  if K = N then     //base case:  reached the end of activity list
    S[0..N];
    S[0] =decision;     //set number of selected activities
    S[N] =decision;     //set activity selection flag
    return S[0..N];
  else     //recursion step
    S[0..N];
    S[0..N] ←ActivitySelectionRecursive(N,A[K+1,N][1..2],K+1,StartTime);
    S[K] = decision;     //set activity selection flag
    S[0] = S[0]+decision;     // update number of selected activities
    return S[0..N];
end
```