Activity scheduling

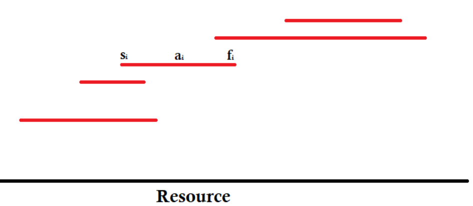
October 10, 2017

Overview

1. Activity scheduling

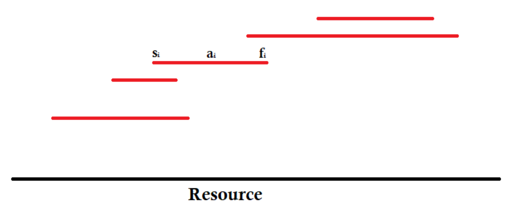
Activity scheduling

- The problem:
  We are given a set of n activities $a_1, \ldots, a_n$ that uses a common resource.
  Each activity a_i has a starting time s_i and a finishing time f_i. We denote $a_i = (s_i, f_i)$.
  The resource can only be used by one activity at a time.
  We wish to select a set of maximal size of activities that can be served by the resource. (The maximality is in terms of number of activities.)



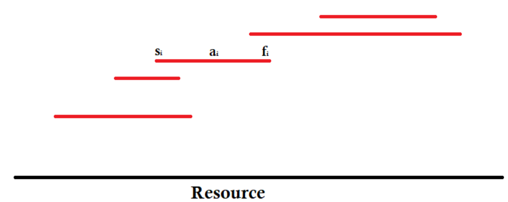
Activity scheduling

- Input: A set S of n activities $S = \{a_1, ..., a_n\}$.
  Output: A maximal sequence $B = (b_1, ..., b_k)$ of disjoint intervals.



Activity scheduling - Greedy approach

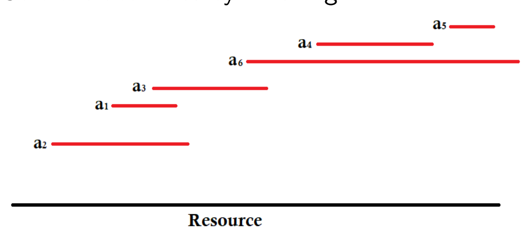
- At each step the algorithm takes the activity that finishes first among the possible activities.



- Sort the activities by finishing time.

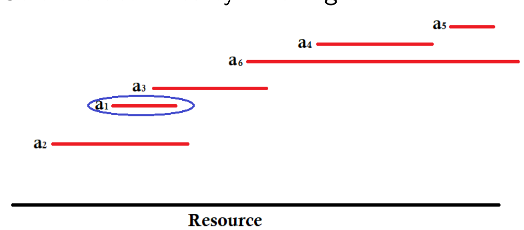
Activity scheduling - Greedy approach

- At each step the algorithm takes the activity that finishes first among the possible activities.
- Sort the activities by finishing time.



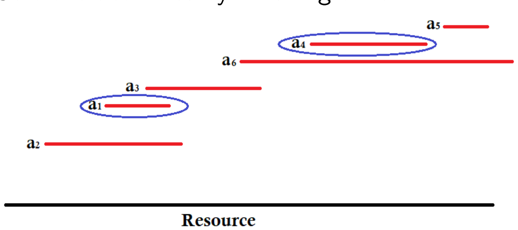
Activity scheduling - Greedy approach

- At each step the algorithm takes the activity that finishes first among the possible activities.
- Sort the activities by finishing time.



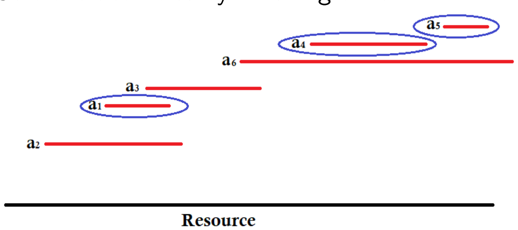
Activity scheduling - Greedy approach

- At each step the algorithm takes the activity that finishes first among the possible activities.
- Sort the activities by finishing time.



Activity scheduling - Greedy approach

- At each step the algorithm takes the activity that finishes first among the possible activities.
- Sort the activities by finishing time.



Activity scheduling - The algorithm

- Activity schedule(S)
  1. Sort S to $(a_1, a_2, ..., a_n)$ according to the finishing times such that $f_1 \leq f_2 \leq ... \leq f_n$.
  2. Initialize B to $\{a_1\}$ and $k = 1$.
  3. For $m = 2$ to n
  4.   if $f_k \leq s_m$
  5. $B = B \cup \{a_m\}$
  6. $k = m$
  7. return B
- The sorting step takes $O(n \log n)$ time. The rest of the steps takes $O(n)$ time.
  Overall we get $O(n \log n)$.

Activity scheduling - The algorithm

- Activity schedule(S)
  1. Sort S to $(a_1, a_2, ..., a_n)$ according to the finishing times such that $f_1 \leq f_2 \leq ... \leq f_n$.
  2. Initialize B to $\{a_1\}$ and $k = 1$.
  3. For $m = 2$ to n
  4. if $f_k \leq s_m$
  5. $B = B \cup \{a_m\}$
  6. $k = m$
  7. return B
- The sorting step takes $O(n \log n)$ time. The rest of the steps takes $O(n)$ time.
  Overall we get $O(n \log n)$.

Activity scheduling - The algorithm

- Activity schedule(S)
  1. Sort S to $(a_1, a_2, ..., a_n)$ according to the finishing times such that $f_1 \leq f_2 \leq ... \leq f_n$.
  2. Initialize B to $\{a_1\}$ and $k = 1$.
  3. For $m = 2$ to n
  4.   if $f_k \leq s_m$
  5. $B = B \cup \{a_m\}$
  6. $k = m$
  7. return B
- The sorting step takes $O(n \log n)$ time. The rest of the steps takes $O(n)$ time.
  Overall we get $O(n \log n)$.

Activity scheduling - Correctness of the algorithm

- Lemma: Let $(b_1, b_2, ..., b_k)$ be the sequence of intervals that the algorithm returns.
  For each i such that $0 \leq i \leq k$, the intervals $(b_1, ..., b_i)$ are disjoint intervals that are the prefix of an optimal solution for the problem.

Activity scheduling - Correctness of the algorithm

- We prove by induction on i.
  For $i = 0$ there is nothing to show.
  We assume that $(b_1, ..., b_i)$ $(i < k)$ is a prefix of an optimal solution and we show that $(b_1, ..., b_i, b_{i+1})$ is also a prefix of an optimal solution.
  Let $(b_1, ..., b_i, c_{i+1}, ..., c_k)$ be an optimal solution. It follows that there exists an interval that starts after b_i finishes. Therefore the algorithm must have some b_{i+1} in the output sequence.
  By the definition of the algorithm, the interval b_{i+1} will have the smallest finishing time possible. In particular:

  $$\text{finishing time of } b_{i+1} \leq \text{ finishing time of } c_{i+1}$$

  It follows that $(b_1, ..., b_i, b_{i+1}, c_{i+2}, ..., c_k)$ is an optimal solution. $\blacksquare$