Algorithms & Complexity: Lecture 8, Greedy Algorithms 2

Sam Barrett

June 2, 2021

1 Interval Scheduling Problem

Problem 1 *(The Interval Scheduling Problem)*

- *Given a set of n requests* $R = \{\texttt{Req}(1), \texttt{Req}(2), \texttt{Req}(3), \ldots, \texttt{Req}(n)\}$
- $\texttt{Req}(i)$ *has a start time given by* $\texttt{Start}(i)$ *and a finish time by* $\texttt{Finish}(i)$
- *There exists a machine which can handle one request at a time.*
- *Two requests conflict if they overlap*

Select a set $C \subseteq R$ *of requests s.t.* $|C|$ *is maximised and no two requests from* C *conflict.*

See Slides for lecture 8 for details on failed greedy algorithm construction. A correct greedy algorithm for tackling this problem can be seen below:

Algorithm 1: Greedy Interval Scheduling

Let $R = \{\texttt{Req}(1), \texttt{Req}(2), \ldots, \texttt{Req}(i), \ldots, \texttt{Req}(n)\}$ be the set of all requests
Let C denote the set of requests that we select
Instantiate $C = \emptyset$
while $R \neq \emptyset$ **do**
    Find the request $\texttt{Req}(i) \in R$ which has the **smallest finish time**
    Add $\texttt{Req}(i)$ to C
    Delete from R all requests that conflict with $\texttt{Req}(i)$
end

1.1 Correctness

- C does not contain any conflicting requests

Suppose there is another set `OPT` which selects more requests than C. Let C select $\texttt{Req}(i_1), \texttt{Req}(i_2), \ldots, \texttt{Req}(i_k)$ in that order.

Let `OPT` schedule $\texttt{Req}(j_1), \texttt{Req}(j_2), \ldots, \texttt{Req}(j_m)$ in that order.

Lemma 1 *For each $1 \leq l \leq k$, we have $\textit{\texttt{Finish}}(i_l) \leq \textit{\texttt{Finish}}(j_l)$*

Proof 1
1. ***Base Case:*** $l = 1$
2. ***Inductive Step:*** $\textit{\texttt{Start}}(j_l) \geq \textit{\texttt{Finish}}(j_{l-1}) \geq \textit{\texttt{Finish}}(i_{l-1})$
   *Therefore, $\textit{\texttt{Req}}(j_l)$ does **not** conflict with $\textit{\texttt{Req}}(i_{l-1})$. However, our algorithm chose i_l instead, meaning $\textit{\texttt{Finish}}(i_l) \leq \textit{\texttt{Finish}}(j_l)$*

Since $m > k$, `OPT` selects a request $\texttt{Req}(j_{k+1})$ we can say:

$$\texttt{Start}(j_{k+1}) \geq \texttt{Finish}(j_k) \geq \texttt{Finish}(i_k)$$

And thus derive a contradiction as our algorithm stops after selecting i_k

A harder variant of the interval scheduling problem must schedule **all** requests, minimising lateness. And is defined as follows:

Problem 2
- *We have n requests*
- *Each requests has a duration given by $\textit{\texttt{Time}}(i)$*
- *Additionally, each request, $\textit{\texttt{Req}}(i)$ now had a deadline, $\textit{\texttt{Deadline}}(i)$*
- *Choosing a start time $\textit{\texttt{Start}}(i)$ for each request not gives a finish time $\textit{\texttt{Finish}}(i) = \textit{\texttt{Start}}(i) + \textit{\texttt{Time}}(i)$*
- *A request $\textit{\texttt{Req}}(i)$ is **late** is $\textit{\texttt{Finish}}(i) > \textit{\texttt{Deadline}}(i)$*

$$\textit{\texttt{Lateness}}(i) = \begin{cases} \textit{\texttt{Finish}}(i) - \textit{\texttt{Deadline}}(i) & \textit{if } \textit{\texttt{Finish}}(i) > \textit{\texttt{Deadline}}(i) \\ 0 & \textit{otherwise} \end{cases}$$

Schedule all requests in a non-conflicting way, minimising the maximum lateness.

A simple greedy algorithm does not find an optimal scheduling.