COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

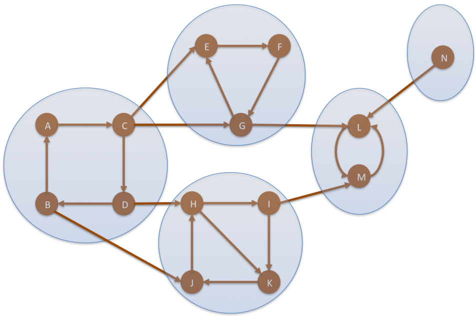
Graph Algorithms

Algorithm

- *time* $\leftarrow 0$

`GraphDFS-with-start-finish(`G`)`
- While there is an "unexplored" vertex u
- `DFS-time(`u`)`

`DFS-time(`u`)`
- Mark u as "explored" and set $start(u) \leftarrow ++\, time$
- While there is an "unexplored" neighbor v of u
  - `DFS-time(`v`)`
- $finish(u) \leftarrow ++\, time$



Algorithm

- $time \leftarrow 0$

`GraphDFS-with-start-finish(`G`)`

- While there is an "unexplored" vertex u
- `DFS-time(`u`)`

`DFS-time(`u`)`

- Mark u as "explored" and set $start(u) \leftarrow ++time$
- While there is an "unexplored" neighbor v of u
  - `DFS-time(`v`)`
- $finish(u) \leftarrow ++time$

- Let $V_1, V_2, ..., V_k$ be the k vertex sets of strongly connected components of G.
- Consider G^{scc} where the vertices are labeled $1, 2, ..., k$.
- Claim 1: If there is a directed edge from node i to node j in G^{scc}, then the highest finish time among vertices in V_i is bigger than the highest finish time among vertices in V_j, when `Graph-DFS-with-start-finish(`G`)` time executed.

Course Overview

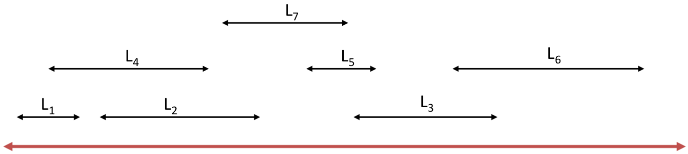
- Material that will be covered in the course:
  - Basic graph algorithms
  - Algorithm Design Techniques
    - Greedy Algorithms
    - Divide and Conquer
    - Dynamic Programming
    - Network Flows
  - Computational intractability

Greedy Algorithms

- *"A local (greedy) decision rule leads to a globally optimal solution."*
- There are two ways to show the above property:
  - Greedy stays ahead
  - Exchange argument

Problem

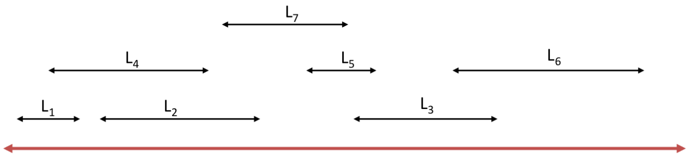
Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.



Problem

Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.

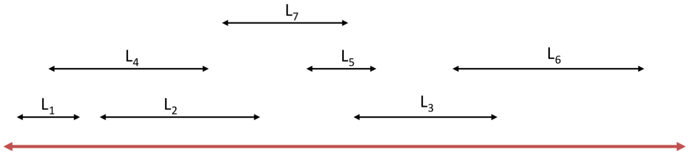
- Candidate greedy choices:
  - Earliest start time
  - Smallest duration
  - Least overlapping



Problem

Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.

- Candidate greedy choices:
  - Earliest start time
  - Smallest duration
  - Least overlapping
  - Earliest finish time



Problem

Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.

Algorithm

`GreedySchedule`

- Initialize R to contain all intervals
- While R is not empty
  - Choose an interval $(S(i), F(i))$ from R that has the smallest value of $F(i)$
  - Delete all intervals in R that overlaps with $(S(i), F(i))$

Problem

Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.

Algorithm

```
GreedySchedule
```
- Initialize R to contain all intervals
- While R is not empty
  - Choose an interval $(S(i), F(i))$ from R that has the smallest value of $F(i)$
  - Delete all intervals in R that overlaps with $(S(i), F(i))$

- Question: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Can we show that $A = O$?

- Question: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Can we show that $A = O$?
- Question Can we show that $|O| = |A|$?

- Question: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Can we show that $A = O$?
- Question Can we show that $|O| = |A|$?
- Yes we can! We will use "greedy stays ahead" method to show this.

Proof sketch

Let $a_1, a_2, ..., a_k$ be the sequence of requests that `GreedySchedule` picks and $o_1, o_2, ..., o_l$ be the requests in O sorted in non-decreasing order by finishing time.

- Claim 1: $F(a_1) \leq F(o_1)$.

- Question: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Can we show that $A = O$?
- Question Can we show that $|O| = |A|$?
- Yes we can! We will use "greedy stays ahead" method to show this.

Proof sketch

Let $a_1, a_2, ..., a_k$ be the sequence of requests that `GreedySchedule` picks and $o_1, o_2, ..., o_l$ be the requests in O sorted in non-decreasing order by finishing time.

- Claim 1: $F(a_1) \leq F(o_1)$.
- Claim 2: If $F(a_1) \leq F(o_1)$, $F(a_2) \leq F(o_2)$, ..., $F(a_{i-1}) \leq F(o_{i-1})$, then $F(a_i) \leq F(o_i)$.

- Question: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Can we show that $A = O$?
- Question Can we show that $|O| = |A|$?
- Yes we can! We will use "greedy stays ahead" method to show this.

Proof sketch

- Let $a_1, a_2, ..., a_k$ be the sequence of requests that `GreedySchedule` picks and $o_1, o_2, ..., o_l$ be the requests in O sorted in non-decreasing order by finishing time.
- We will show by induction that $\forall i, F(a_i) \leq F(o_i)$
  - Claim 1 (base case): $F(a_1) \leq F(o_1)$.
  - Claim 2 (inductive step): If $F(a_1) \leq F(o_1)$, $F(a_2) \leq F(o_2)$, ..., $F(a_{i-1}) \leq F(o_{i-1})$, then $F(a_i) \leq F(o_i)$.
- `GreedySchedule` could not have stopped after a_k.

Problem

Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.

Algorithm

`GreedySchedule`

- Initialize R to contain all intervals
- While R is not empty
  - Choose an interval $(S(i), F(i))$ from R that has the smallest value of $F(i)$
  - Delete all intervals in R that overlaps with $(S(i), F(i))$

- Running time?

Problem

Interval scheduling: Given a set of n intervals of the form $(S(i), F(i))$, find the largest subset of non-overlapping intervals.

Algorithm

`GreedySchedule`

- While R is not empty
- Choose an interval $(S(i), F(i))$ from R that has the smallest value of $F(i)$
- Delete all intervals in R that overlaps with $(S(i), F(i))$

- Running time? $O(n \log n)$

Problem

Job scheduling: You are given n jobs and you are supposed to schedule these jobs on a machine. Each job i consists of a duration $T(i)$ and a deadline $D(i)$. The *lateness* of a job w.r.t. a schedule is defined as $\max(0, F(i) - D(i))$, where $F(i)$ is the finishing time of job i as per the schedule. The goal is to minimise the maximum lateness.

End