COL202: Discrete Mathematical Structures

Ragesh Jaiswal, CSE, IIT Delhi

Induction and Recursion

Definition (Principle of mathematical induction)

To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

- Basis step: We verify that $P(1)$ is true.
- Inductive step: We show that the conditional statement $P(k) \rightarrow P(k+1)$ is true for all positive integers k.

- In the inductive step, we assume that for arbitrary positive integer $P(k)$ is true and then show that $P(k+1)$ must also be true. The assumption that $P(k)$ is true is called the *inductive hypothesis*.
- Induction may be expressed as the following rule of inference:

$$(P(1) \wedge \forall k(P(k) \rightarrow P(k+1))) \rightarrow \forall n\, P(n)$$

- Show that the sum of the first n odd integers is n^2.

- Show that the sum of the first n odd integers is n^2.
- Show that for all n, $n < 2^n$.

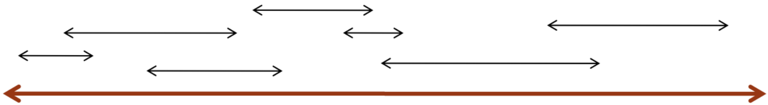
- Show that the sum of the first n odd integers is n^2.
- Show that for all n, $n < 2^n$.
- Show that $2^n < n!$ for every integer n with $n \geq 4$.

- Show that the sum of the first n odd integers is n^2.
- Show that for all n, $n < 2^n$.
- Show that $2^n < n!$ for every integer n with $n \geq 4$.
- Prove the following generalization of De Morgan's laws:

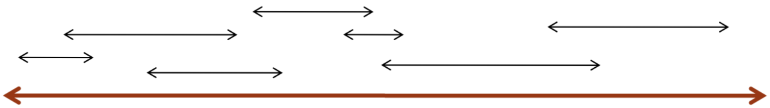
$$\overline{\cap_{j=1}^{n} A_j} = \cup_{j=1}^{n} \overline{A_j}$$

whenever $A_1, A_2, ..., A_n$ are subsets of a universal set U and $n \geq 2$.

- You have a lecture room and you get *n* requests for scheduling lectures. Each request has a start and an end time. Design an algorithm that maximizes the number of lectures held in the room.



- You have a lecture room and you get n requests for scheduling lectures. Each request has a start and an end time. Design an algorithm that maximizes the number of lectures held in the room.



- Consider the algorithm that schedules based on end time of lectures. We will show that this algorithm gives the optimal solution.

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? $O(n \log n)$

- Claim: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Then $|O| = |A|$.

- Claim: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Then $|O| = |A|$.

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: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Then $|O| = |A|$.

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 $F(a_i) \leq F(o_i)$.

- Claim: Let O denote some optimal subset and A be the subset given by `GreedySchedule`. Then $|O| = |A|$.

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 (basis step): $F(a_1) \leq F(o_1)$.
  - 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))
```

- Another way to prove that the above greedy algorithm returns an optimal solution is by a slightly different induction argument.
- We consider the propositional function:
  $P(n)$: For any input instance, if the greedy algorithm returns a solution with n lectures, then all optimal solutions also have n lectures.
- The detailed discussion with respect to this Induction argument may be found in the textbook.

- We will only consider simple graphs for this discussion which are graphs that do not have self loops or multi-edges (i.e., multiple edges between a pair of vertices).

Definition (Strongly connected graph)

An undirected graph is called strongly connected iff for every pair of vertices in the graph there is a path between these vertices.

Definition (Tree)

An undirected graph is called a tree iff the graph is strongly connected and does not have any cycles.

Definition (Cycle)

A sequence of vertices $v_1, v_2, ..., v_k$ in an undirected graph is called a cycle iff $k > 3$, $v_1 = v_k$, $v_1, v_2, ..., v_{k-1}$ are distinct, and for every $1 \leq i \leq k-1$, there is an edge between v_i and v_{i+1}.

- Show that: Every tree with n vertices has exactly $(n-1)$ edges.

End