Reductions

Madhavan Mukund

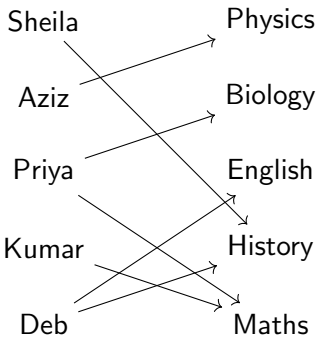
https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 11

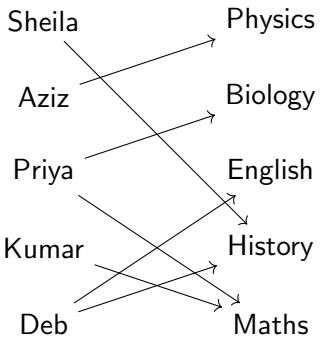
Bipartite matching

- Each instructor is willing to teach a set of courses



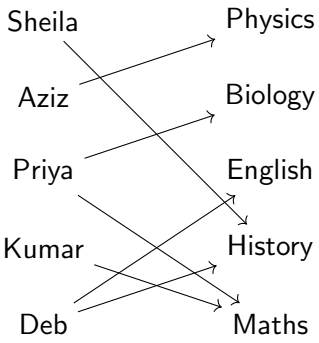
Bipartite matching

- Each instructor is willing to teach a set of courses
- Find an allocation so that
  - Each course is taught by a single instructor
  - Each instructor teaches only one course, which he/she is willing to teach



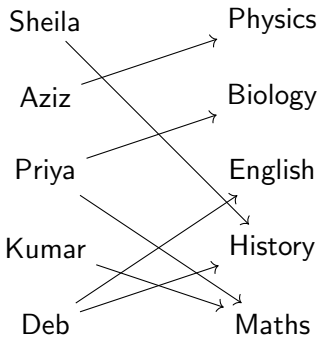
Bipartite matching

- V partitioned into V_0, V_1



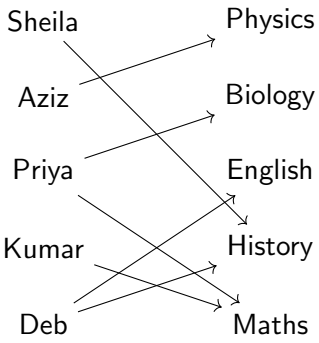
Bipartite matching

- V partitioned into V_0, V_1
- All edges from V_0 to V_1



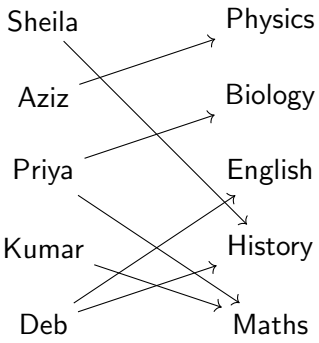
Bipartite matching

- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint



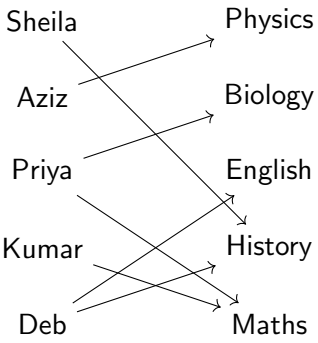
Bipartite matching

- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint
- Find largest matching



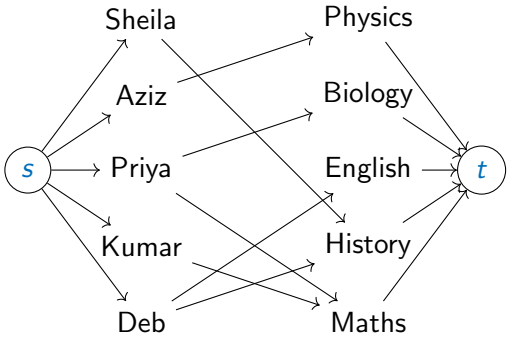
Bipartite matching

- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint
- Find largest matching
- If possible, a perfect matching, all nodes covered



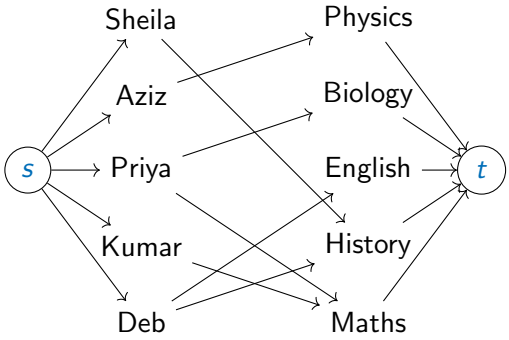
Bipartite matching

- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint
- Find largest matching
- If possible, a perfect matching, all nodes covered
- Add a source and a sink
  - All edge capacities 1



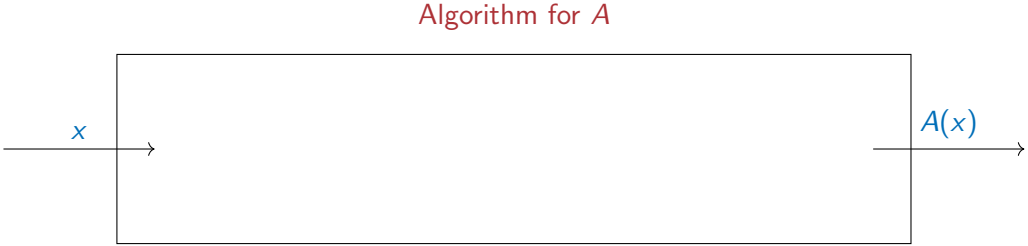
Bipartite matching

- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint
- Find largest matching
- If possible, a perfect matching, all nodes covered
- Add a source and a sink
  - All edge capacities 1
- Find a maximum flow from s to t!



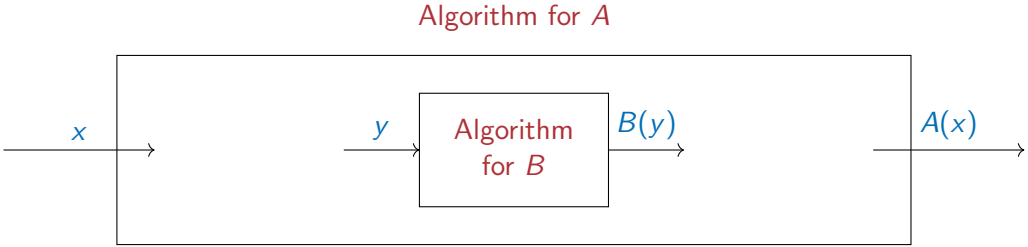
Reductions

- We want to solve problem A



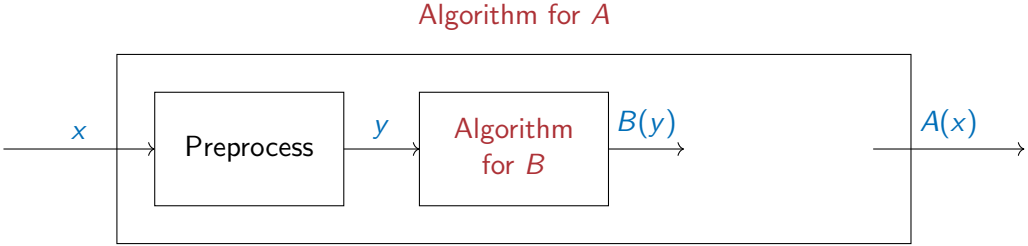
Reductions

- We want to solve problem A
- We know how to solve problem B



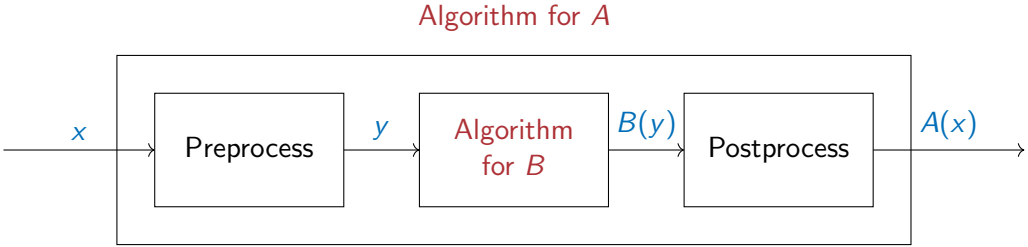
Reductions

- We want to solve problem A
- We know how to solve problem B
- Convert input for A into input for B



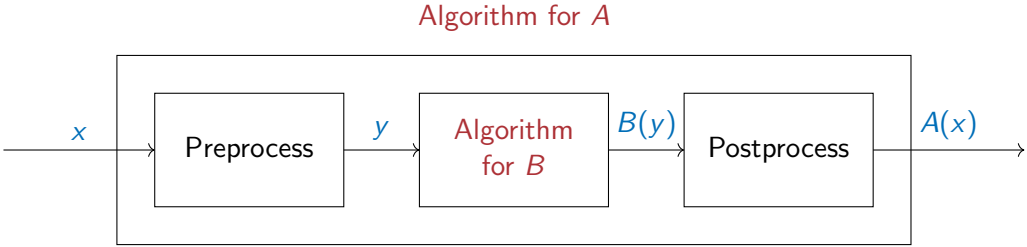
Reductions

- We want to solve problem A
- We know how to solve problem B
- Convert input for A into input for B
- Interpret output of B as output of A



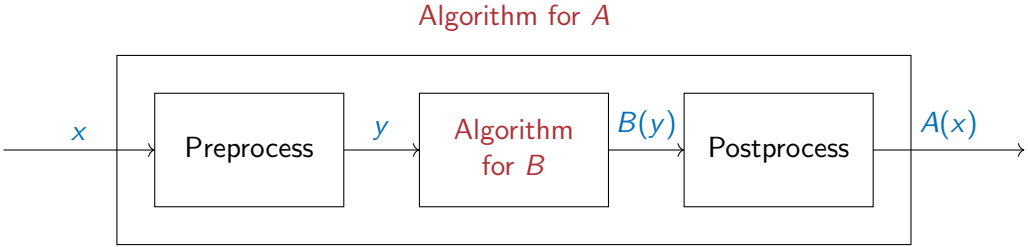
Reductions

- A reduces to B



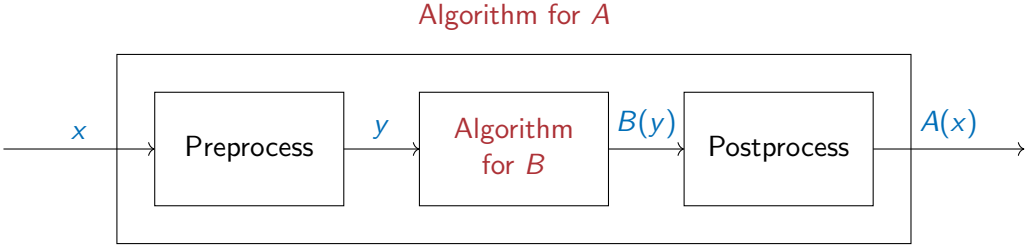
Reductions

- A reduces to B
- Can transfer efficient solution from B to A



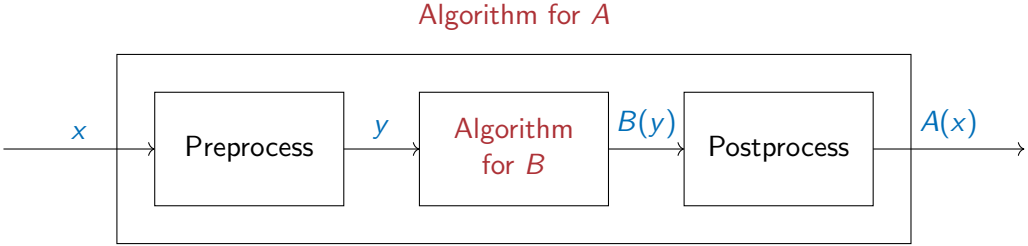
Reductions

- A reduces to B
- Can transfer efficient solution from B to A
- But preprocessing and postprocessing must also be efficient!



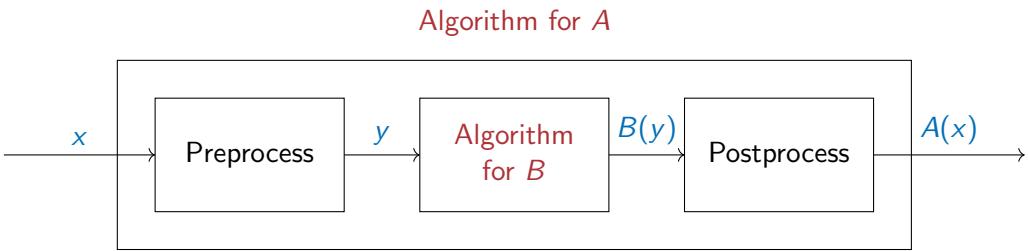
Reductions

- A reduces to B
- Can transfer efficient solution from B to A
- But preprocessing and postprocessing must also be efficient!
- Typically, both should be polynomial time



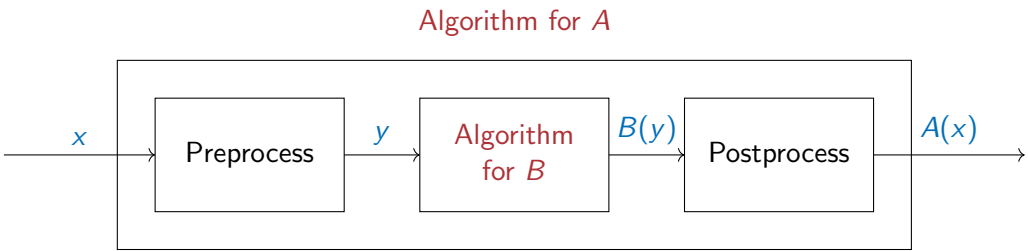
Reductions

- Bipartite matching reduces to max flow



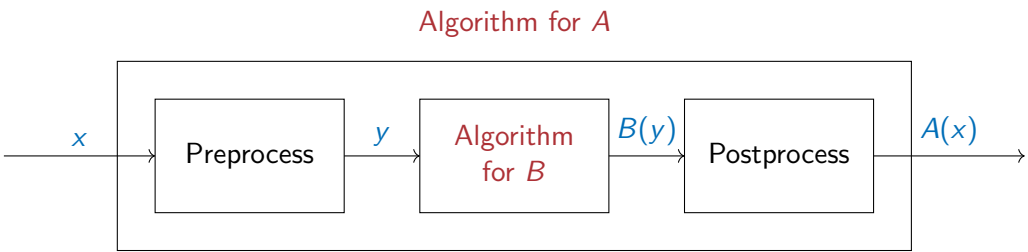
Reductions

- Bipartite matching reduces to max flow
- Max flow reduces to LP



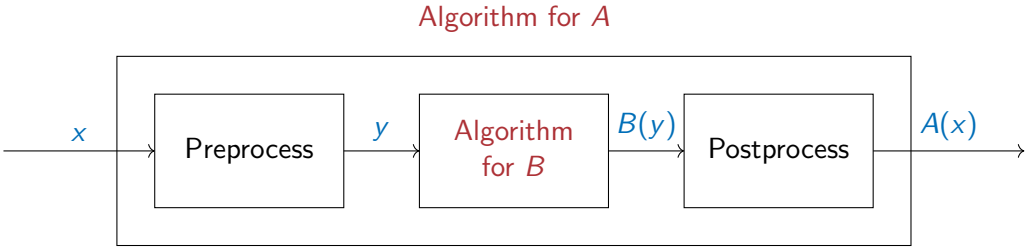
Reductions

- Bipartite matching reduces to max flow
- Max flow reduces to LP
- Number of variables, constraints is linear in the size of the graph



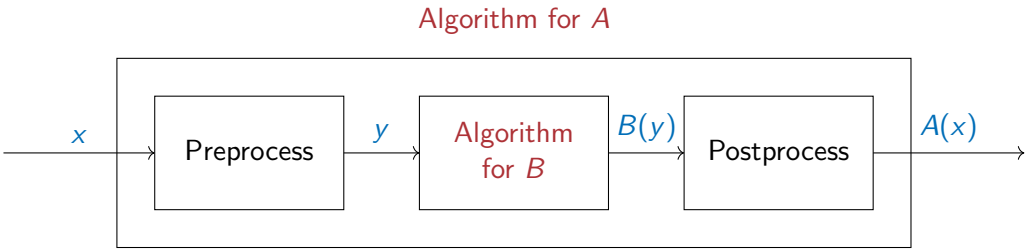
Reductions

- Reverse interpretation is also useful



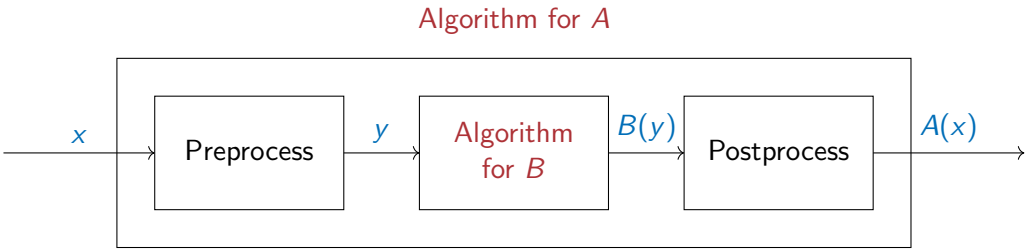
Reductions

- Reverse interpretation is also useful
- If A is known to be intractable and A reduces to B, then B must also be intractable



Reductions

- Reverse interpretation is also useful
- If A is known to be intractable and A reduces to B, then B must also be intractable
- Otherwise, efficient solution for B will yield efficient solution for A



Big hammers

- LP and network flows are powerful tools

Big hammers

- LP and network flows are powerful tools
- Many algorithmic problems can be reduced to them

Big hammers

- LP and network flows are powerful tools
- Many algorithmic problems can be reduced to them
- Efficient, off-the-shelf implementations are available

Big hammers

- LP and network flows are powerful tools
- Many algorithmic problems can be reduced to them
- Efficient, off-the-shelf implementations are available
- Useful to understand what can (and cannot) be modelled in terms of LP and flows