Reductions

Madhavan Mukund

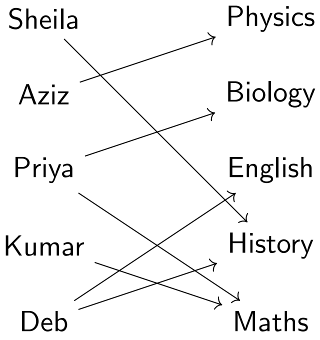
https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 11

Bipartite matching

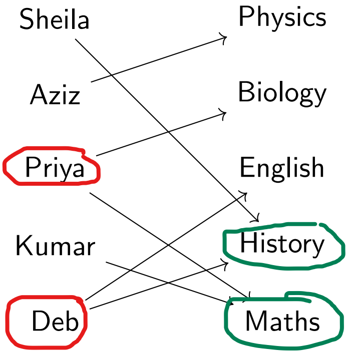
- Each instructor is willing to teach a set of courses



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 These instructors teach two courses
  - Each instructor teaches only one course, which he/she is willing to teach

These courses are taught by more than one teacher

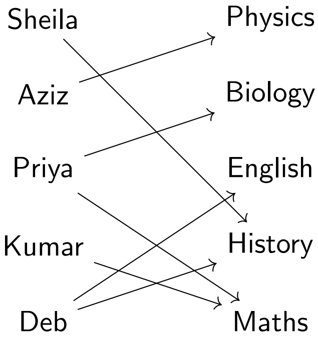


Bipartite matching

- V partitioned into V_0, V_1

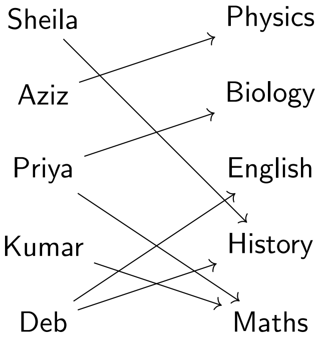
A bipartite matching or a complete bipartite graph is a graph whose vertices can be divided into two disjoint sets, U and V.

Such that every vertex in U is connected to exactly one vertex in V and vice versa. This is also known as a 1-to-1 mapping.



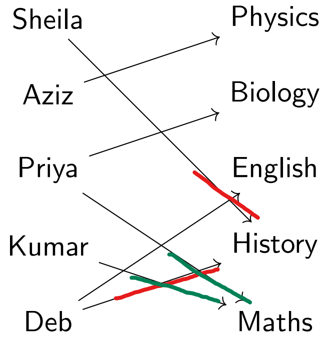
Bipartite matching

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



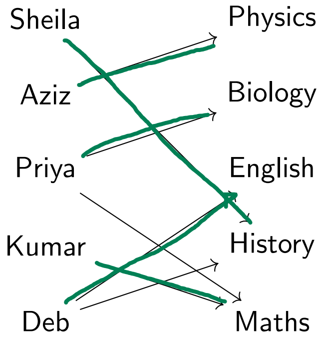
Bipartite matching

- 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

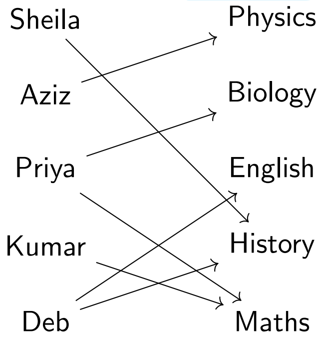
- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint
- Find largest matching



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

$V_0 = 5$ $V_1 = 5$



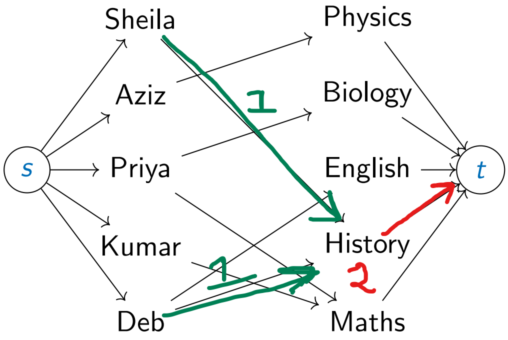
Perfect Matching: This is a special case of a bipartite matching, where the number of edges is equal to the number of vertices in both sets

Bipartite matching

s: source node
t: sink node

If both Sheila and Deb are matched with History then flow from History to sink is 2, but max capacity is 1

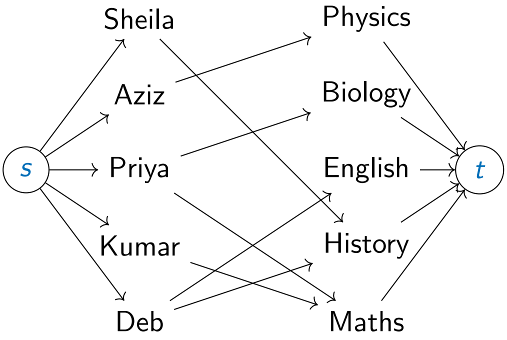
- V partitioned into V_0, V_1
- All edges from V_0 to V_1
- Matching: subset of edges so that no two of them share an endpoint
- Find largest matching
- If possible, a perfect matching, all nodes covered
- Add a source and a sink
  - All edge capacities 1



1. Now we will think about this matching as a flow.
2. As capacity of each edge is 1, no teacher can be assigned 2 subjects as edge cannot have flow of 2

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!

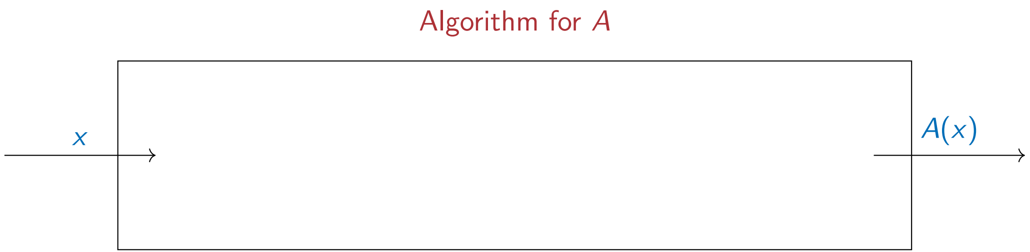


So checking if we have a bipartite matching corresponds to finding maximum flow from s to t.

Reductions

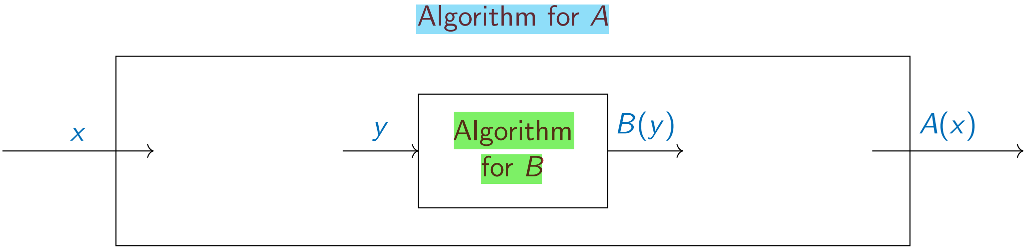
- We want to solve problem A

1. Reductions are algorithms that transform one problem into another, usually easier, problem.
2. In this case we transform the problem of Bipartite Matching to Network Flow



Reductions

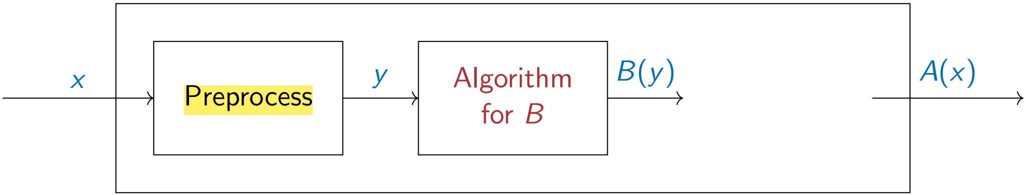
- We want to solve problem A (Bipartite)
- We know how to solve problem B (Network Flow)



Reductions

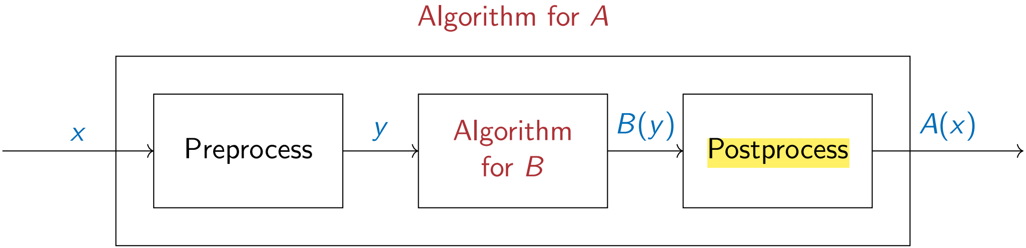
- We want to solve problem A
- We know how to solve problem B
- Convert input for A into input for B

Algorithm for A



Reductions

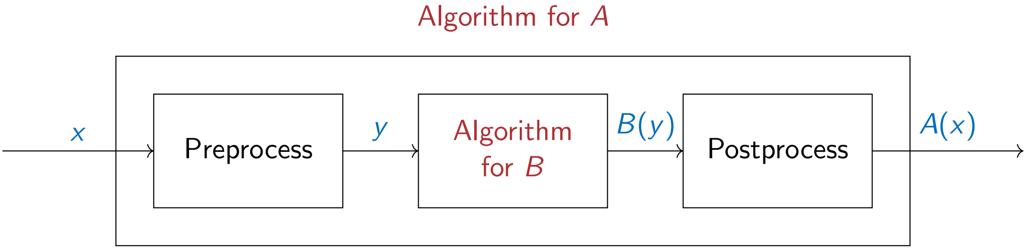
- We want to solve problem A
- We know how to solve problem B
- Convert input for A into input for B
- Interpret output of B as output of A



Reductions

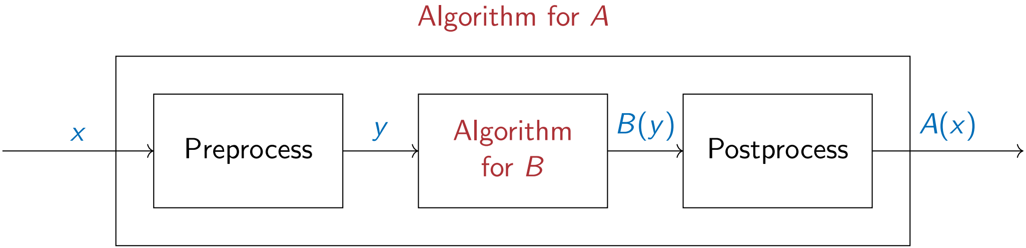
- A reduces to B

In other words the problem of solving A reduces to solving B



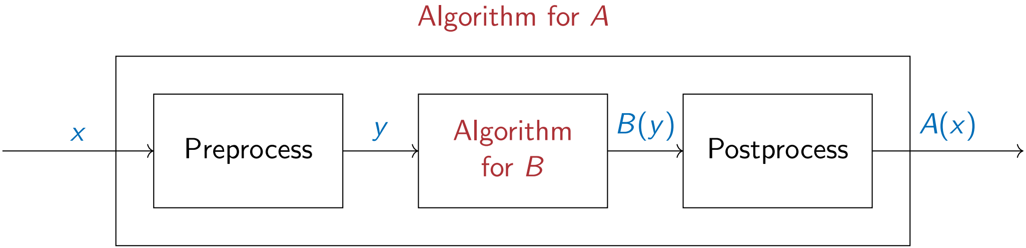
Reductions

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



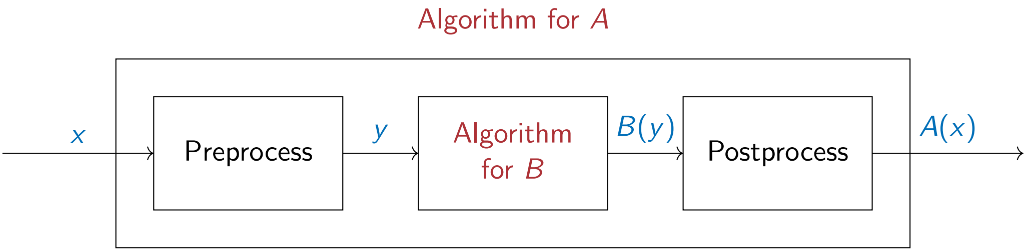
Reductions

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



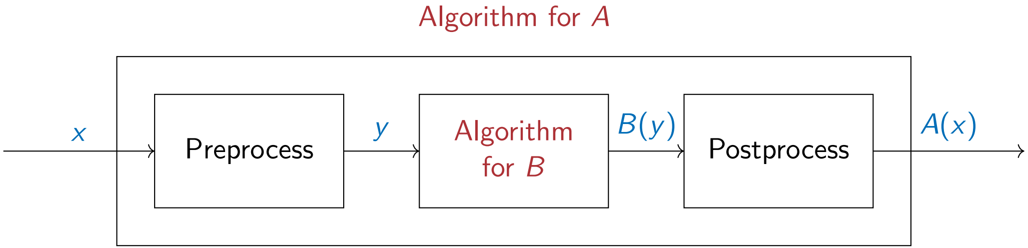
Reductions

- 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



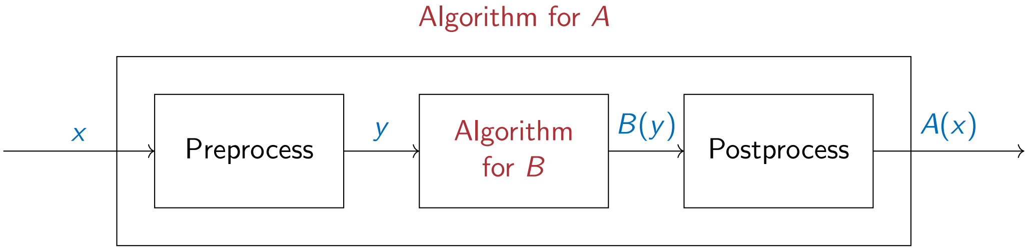
Reductions

1. It was a LP because it had an objective function of maximizing the flow coming out of edges of the source node.

2. We had constraints on capacities of each edge

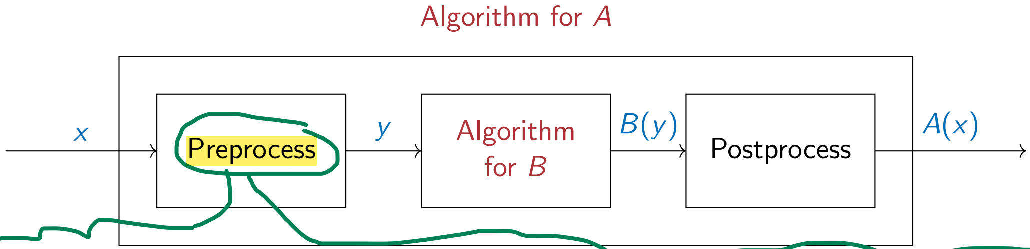
3. We had conservation constraint per node

- Max flow reduces to LP



Reductions

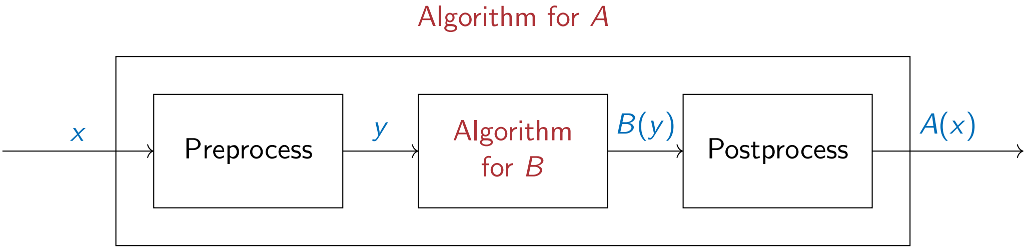
- Bipartite matching reduces to max flow
- Max flow reduces to LP
- Number of variables, constraints is linear in the size of the graph



Going from max flow we are adding one variable per edge (edge capacity) and one constraint per node (conservation constraint) then number of variables and constraints are linear in size.

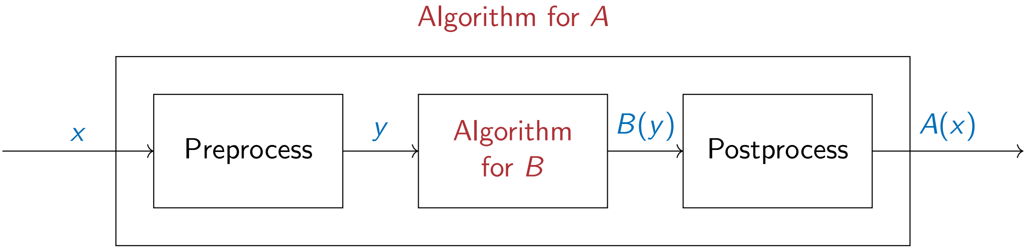
Reductions

- Reverse interpretation is also useful



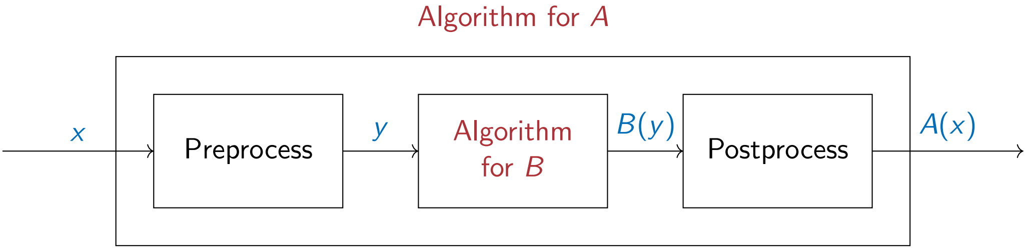
- Reverse interpretation is also useful
- If A is known to be intractable and A reduces to B, then B must also be intractable

In other words if A is known to be hard to solve and we can reduce A to B, then B must also be hard to solve



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

There are efficient commercial and non-commercial libraries available to which solve these problems efficiently

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