Cost-Based Search as IP

Credits to Gavin for the problem :)

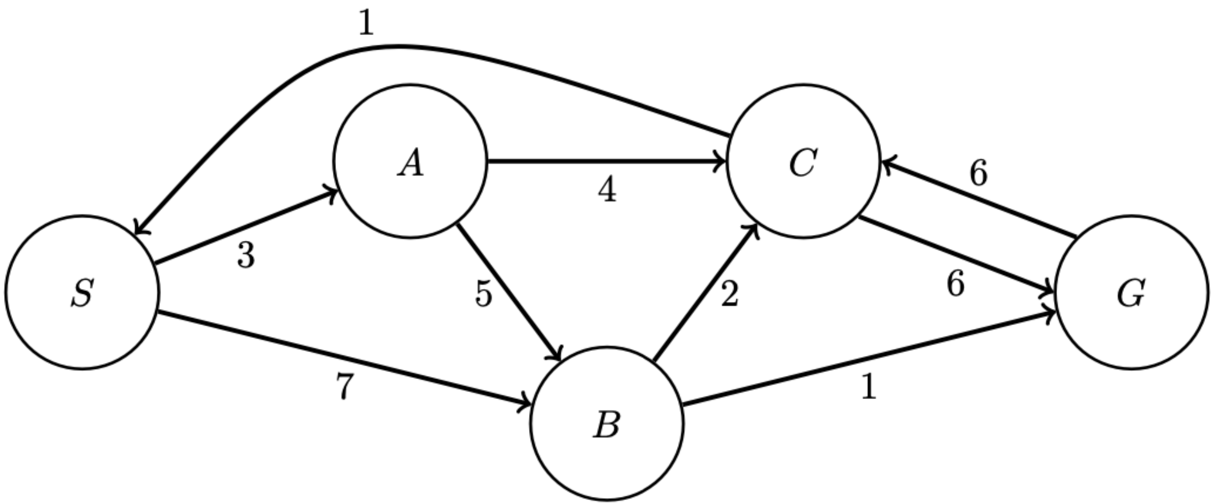
Announcements

- Midterm 1 on Thursday 9/28
    - One page cheat sheet double-sided, must be hand-written with paper and pen
    - Two practice midterms + solutions released
    - Covers Lectures 1-9 (everything before midterm is fair game)

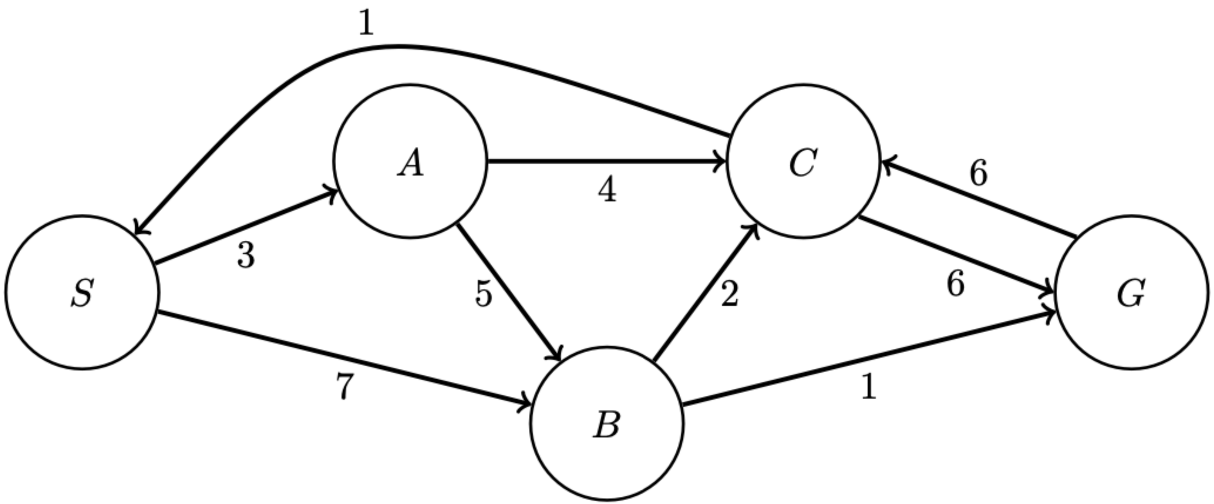
Motivation

- Many problems can be solved by search (e.g., backtracking, branch and bound, etc.) but we haven't seen anything on the other direction
- IP is a very expressive representation

Formulating Search as IP

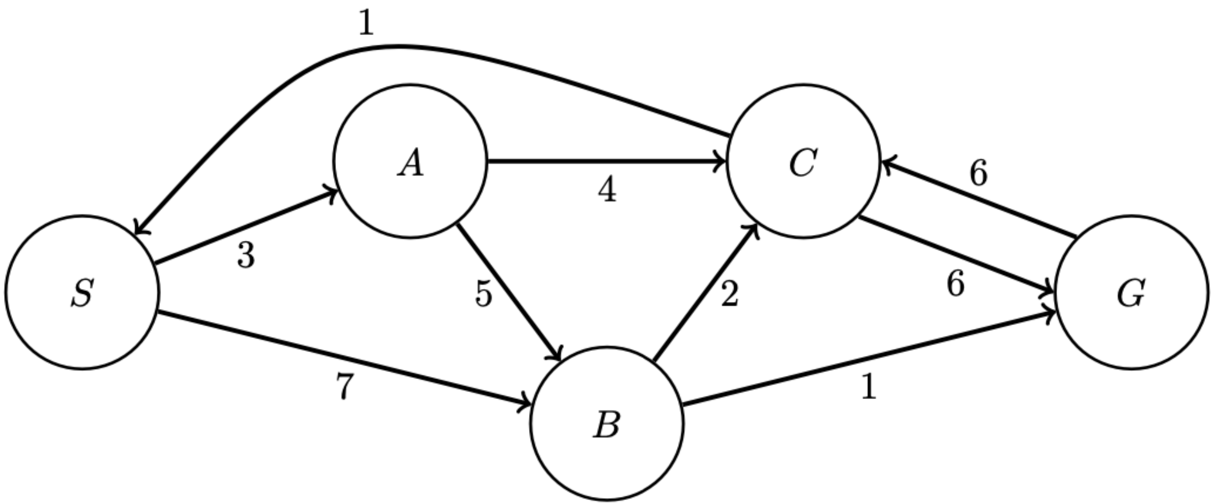


Formulating Search as IP



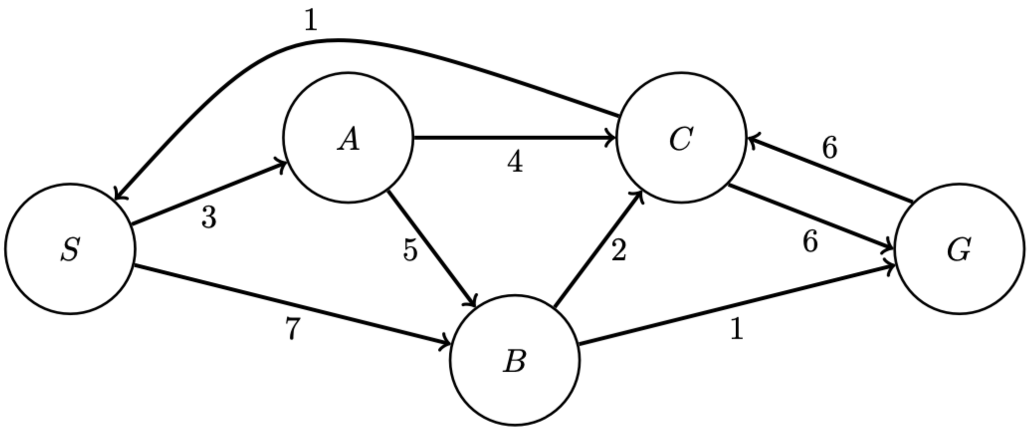
Variables:

Formulating Search as IP



Variables: binary variable for each edge in the graph, representing whether the edge is in the final path or not (0 means edge is not in the final path, 1 means edge is in the final path)

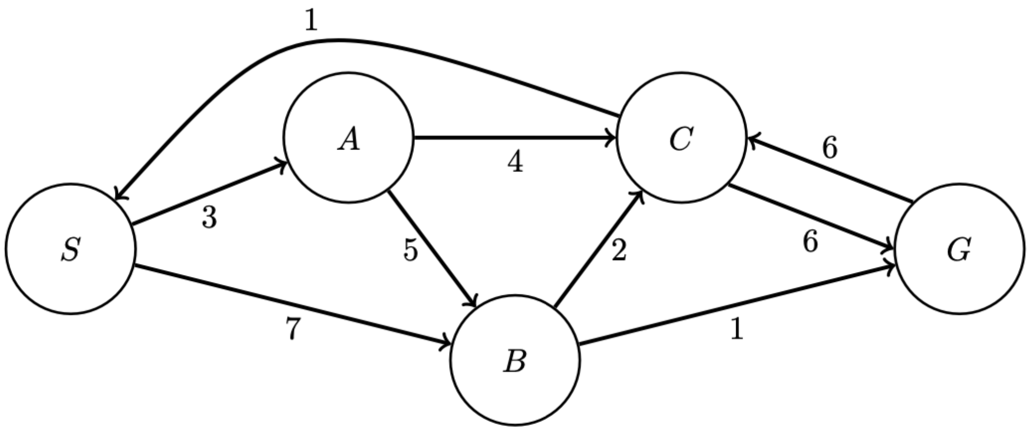
Formulating Search as IP



Variables: binary variable for each edge in the graph, representing whether the edge is in the final path or not (0 means edge is not in the final path, 1 means edge is in the final path)

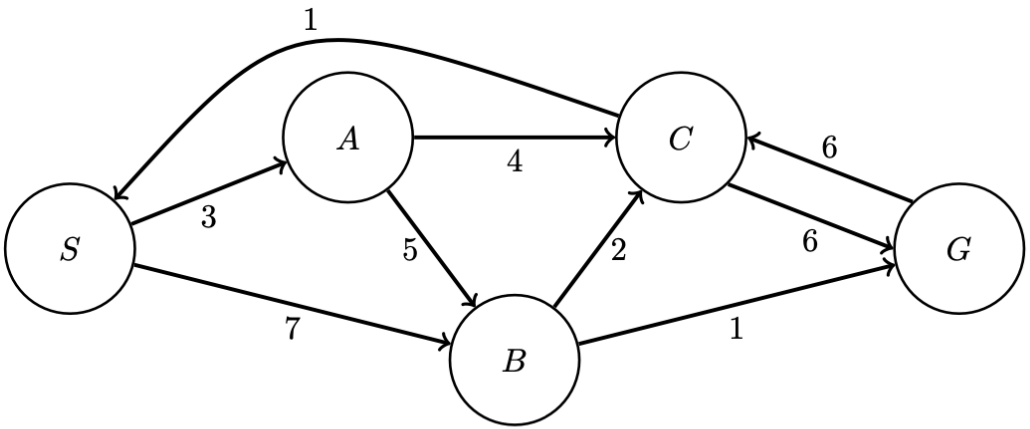
Ex: $x_{X \to Y}$ is a binary variable representing whether the edge $X \to Y$ is in the final path

Formulating Search as IP



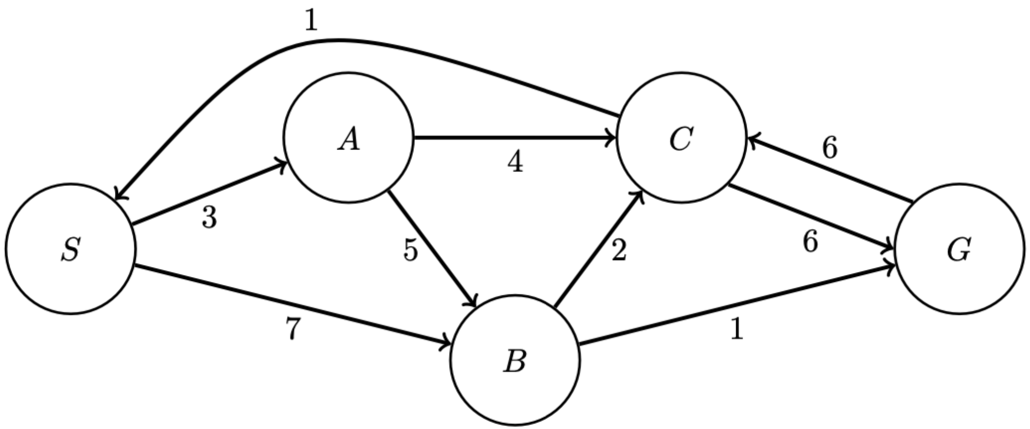
How to represent the path S→A→C→G?

Formulating Search as IP



How to represent the path S→A→C→G?
3 edges: {S→A, A→C, C→G}
$x_{S \to A}$ = indicator for whether S→A is in the path, etc (same for every path in our graph)

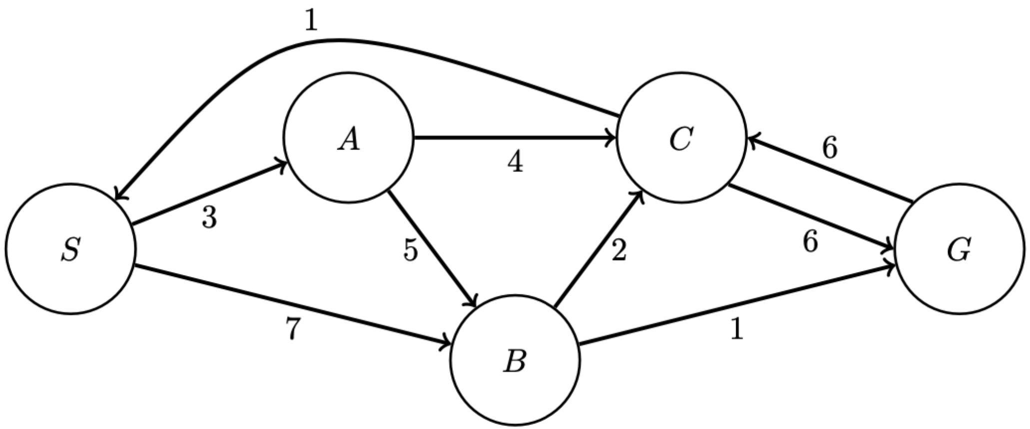
Formulating Search as IP



How to represent the path S→A→C→G?
3 edges: {S→A, A→C, C→G}
$x_{S\to A}$ = indicator for whether S→A is in the path, etc (same for every path in our graph)

$$\left(x_{S\to A}=1 \quad x_{S\to B}=0 \quad x_{A\to B}=0 \quad x_{A\to C}=1 \quad x_{B\to C}=0 \quad x_{B\to G}=0 \quad x_{C\to S}=0 \quad x_{C\to G}=1 \quad x_{G\to C}=0\right)$$

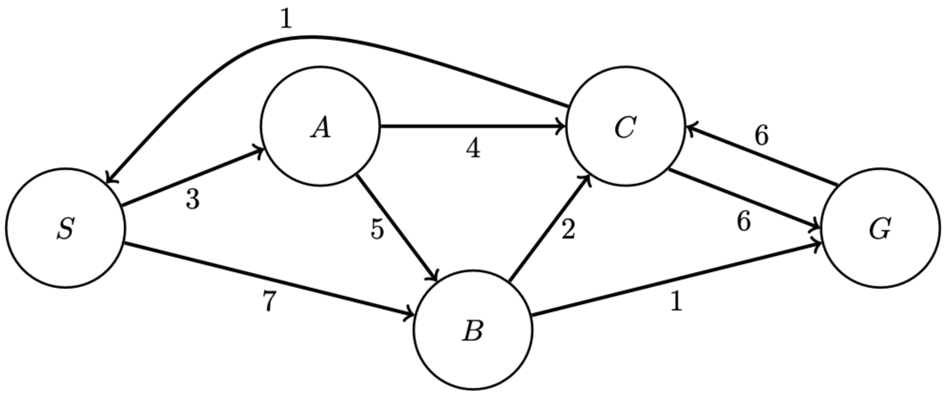
Formulating Search as IP



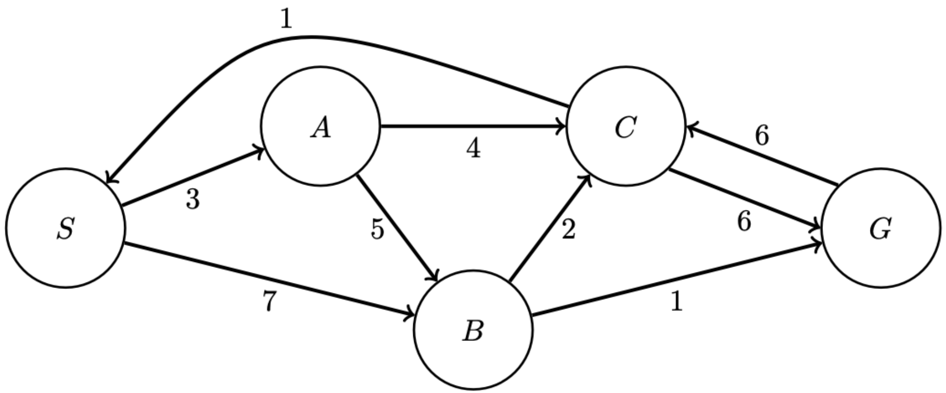
How to represent the path S→A→C→G?
3 edges: {S→A, A→C, C→G}
$x_{S\to A}$ = indicator for whether S→A is in the path, etc (same for every path in our graph)

$$\left(x_{S\to A}=1 \quad x_{S\to B}=0 \quad x_{A\to B}=0 \quad x_{A\to C}=1 \quad x_{B\to C}=0 \quad x_{B\to G}=0 \quad x_{C\to S}=0 \quad x_{C\to G}=1 \quad x_{G\to C}=0\right)$$

9-tuple: (1, 0, 0, 1, 0, 0, 0, 1, 0)

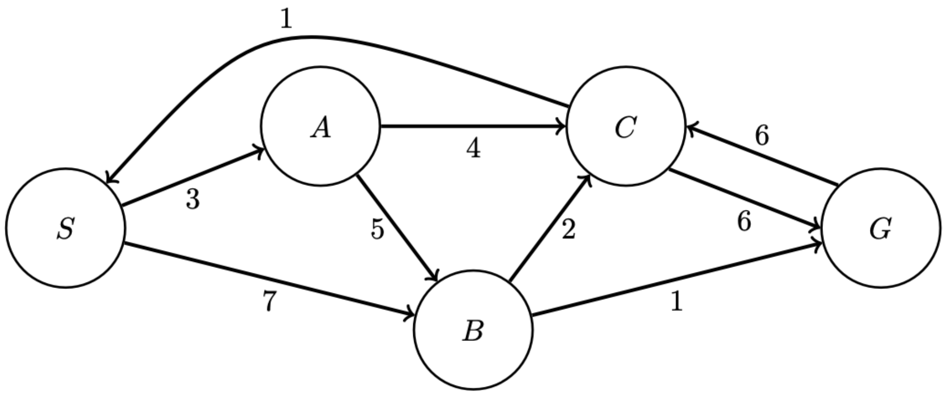


Order: $x_{S \to A}$, $x_{S \to B}$, $x_{A \to B}$, $x_{A \to C}$, $x_{B \to C}$, $x_{B \to G}$, $x_{C \to S}$, $x_{C \to G}$, $x_{G \to C}$



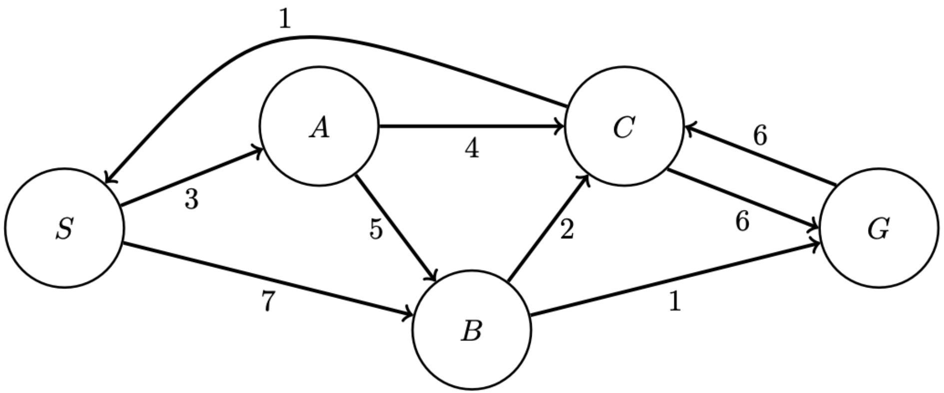
Order: $x_{S\to A}$, $x_{S\to B}$, $x_{A\to B}$, $x_{A\to C}$, $x_{B\to C}$, $x_{B\to G}$, $x_{C\to S}$, $x_{C\to G}$, $x_{G\to C}$

a) i) 9-tuple representation for S→A→B→C→G



Order: $x_{S\to A}$, $x_{S\to B}$, $x_{A\to B}$, $x_{A\to C}$, $x_{B\to C}$, $x_{B\to G}$, $x_{C\to S}$, $x_{C\to G}$, $x_{G\to C}$

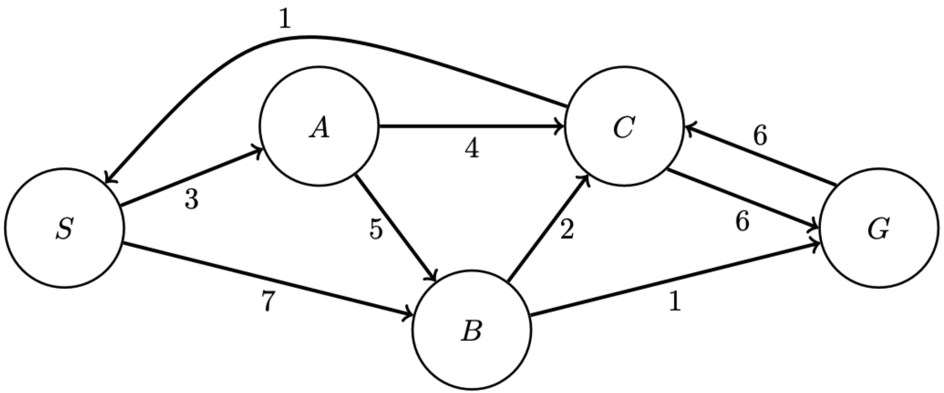
a) i) 9-tuple representation for S→A→B→C→G



Order: $x_{S\to A}$, $x_{S\to B}$, $x_{A\to B}$, $x_{A\to C}$, $x_{B\to C}$, $x_{B\to G}$, $x_{C\to S}$, $x_{C\to G}$, $x_{G\to C}$

a) i) 9-tuple representation for S→A→B→C→G

(1, 0, 1, 0, 1, 0, 0, 1, 0)

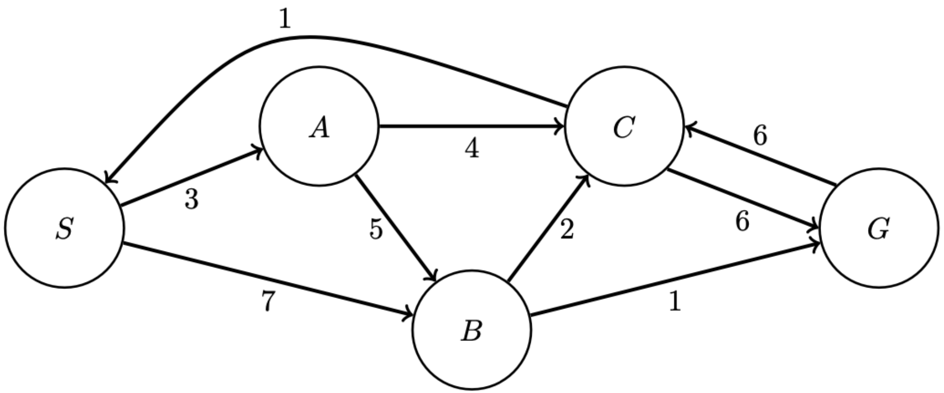


Order: $x_{S \to A}$, $x_{S \to B}$, $x_{A \to B}$, $x_{A \to C}$, $x_{B \to C}$, $x_{B \to G}$, $x_{C \to S}$, $x_{C \to G}$, $x_{G \to C}$

a) i) 9-tuple representation for S→A→B→C→G

(1, 0, 1, 0, 1, 0, 0, 1, 0)

ii) 9-tuple representation for A→C→S→B



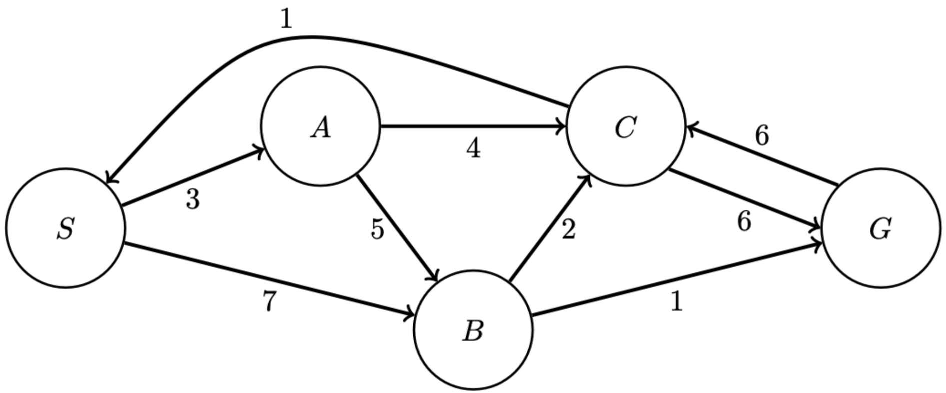
Order: $x_{S \to A}$, $x_{S \to B}$, $x_{A \to B}$, $x_{A \to C}$, $x_{B \to C}$, $x_{B \to G}$, $x_{C \to S}$, $x_{C \to G}$, $x_{G \to C}$

a) i) 9-tuple representation for S→A→B→C→G

(1, 0, 1, 0, 1, 0, 0, 1, 0)

ii) 9-tuple representation for A→C→S→B

(0, 1, 0, 1, 0, 0, 1, 0, 0)



Order: $x_{S\to A}$, $x_{S\to B}$, $x_{A\to B}$, $x_{A\to C}$, $x_{B\to C}$, $x_{B\to G}$, $x_{C\to S}$, $x_{C\to G}$, $x_{G\to C}$

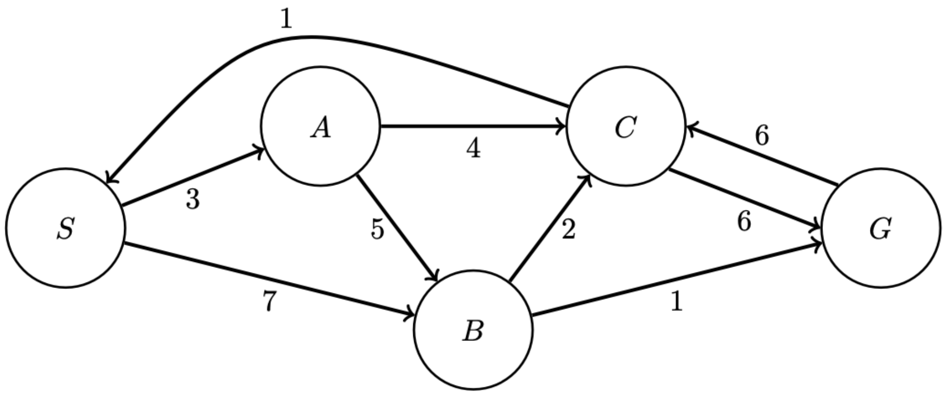
a) i) 9-tuple representation for S→A→B→C→G

(1, 0, 1, 0, 1, 0, 0, 1, 0)

ii) 9-tuple representation for A→C→S→B

(0, 1, 0, 1, 0, 0, 1, 0, 0)

iii) Path that corresponds to (0, 0, 1, 0, 1, 0, 0, 0, 0)



Order: $x_{S\to A}$, $x_{S\to B}$, $x_{A\to B}$, $x_{A\to C}$, $x_{B\to C}$, $x_{B\to G}$, $x_{C\to S}$, $x_{C\to G}$, $x_{G\to C}$

a) i) 9-tuple representation for S→A→B→C→G

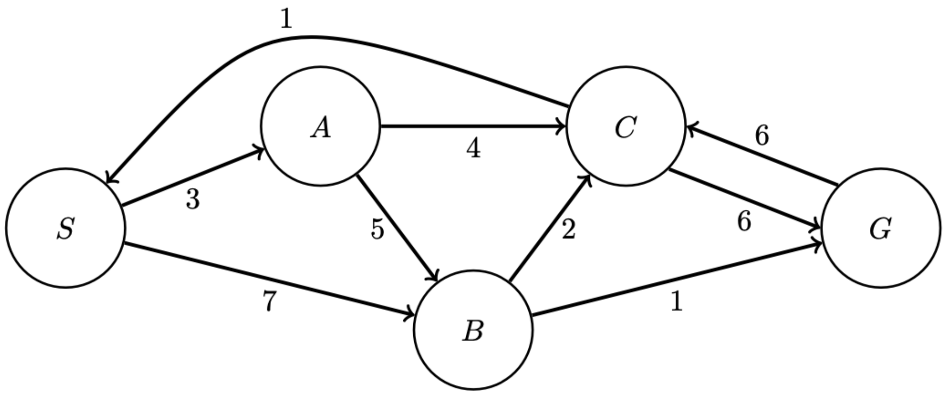
(1, 0, 1, 0, 1, 0, 0, 1, 0)

ii) 9-tuple representation for A→C→S→B

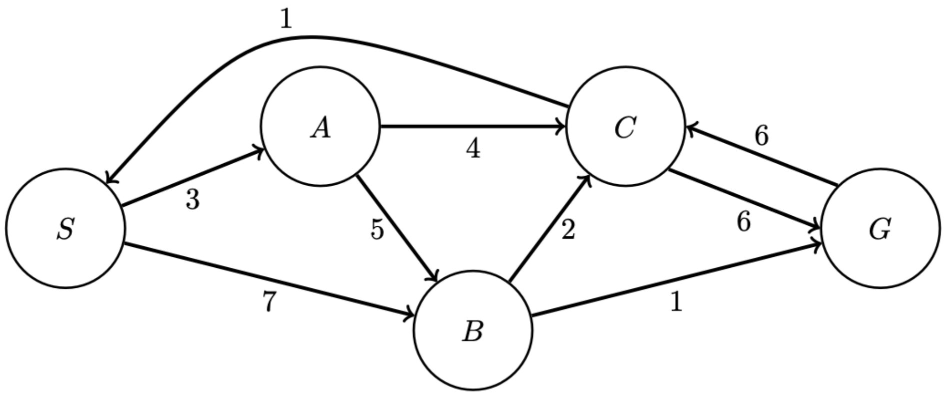
(0, 1, 0, 1, 0, 0, 1, 0, 0)

iii) Path that corresponds to (0, 0, 1, 0, 1, 0, 0, 0, 0)

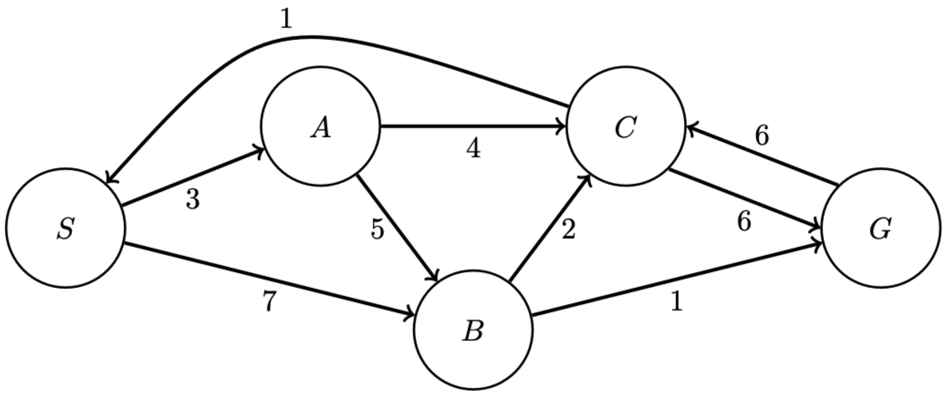
A→B→C



Constraints:

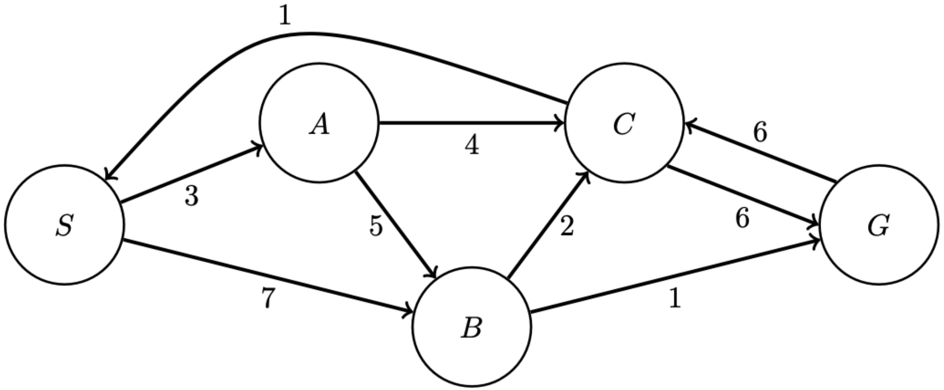


Constraints: need to make sure paths are valid



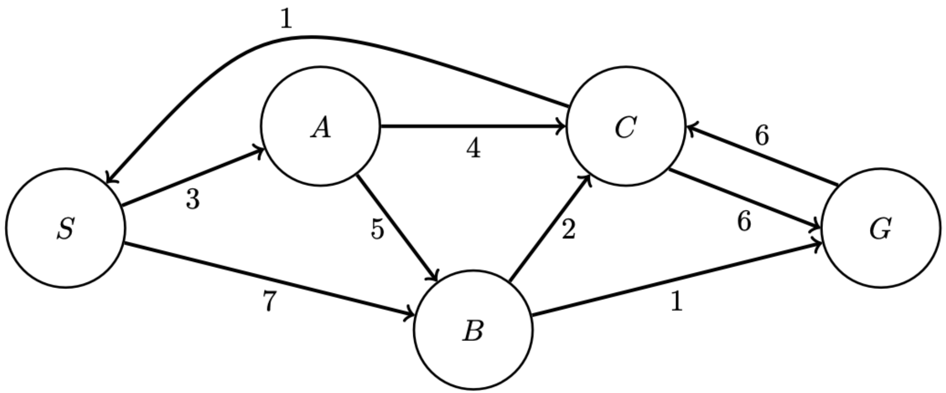
Constraints: need to make sure paths are valid

1) Ensure path starts at S

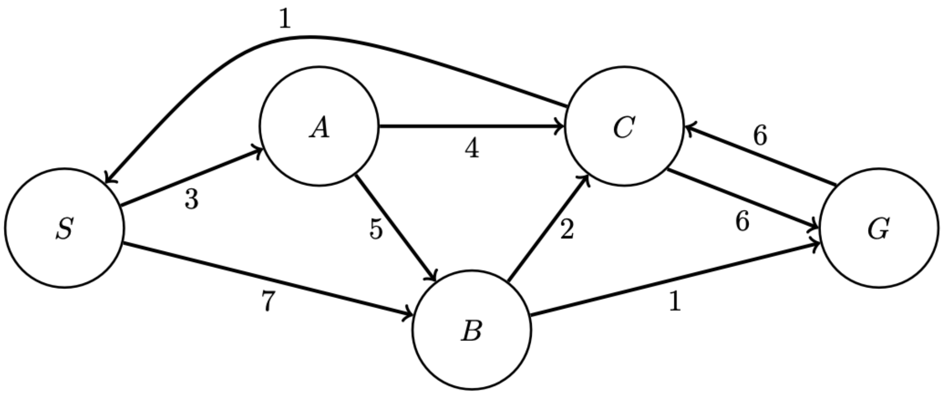


Constraints: need to make sure paths are valid

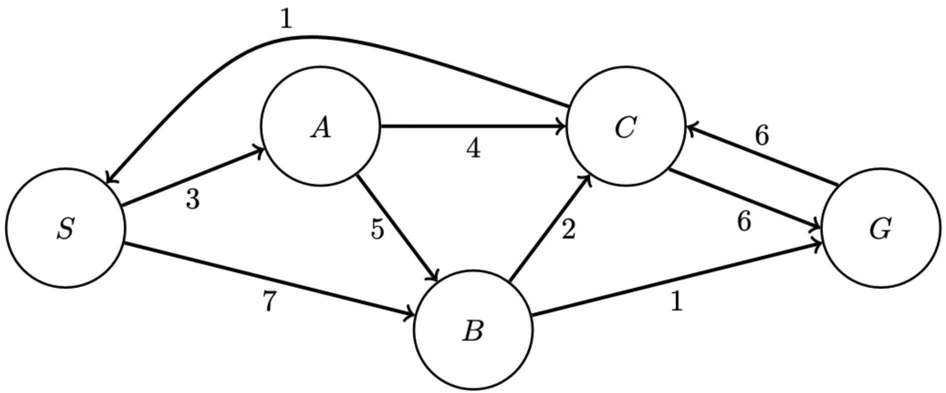
1) Ensure path starts at S
2) Ensure path ends at G



Constraint 1: path starts at S

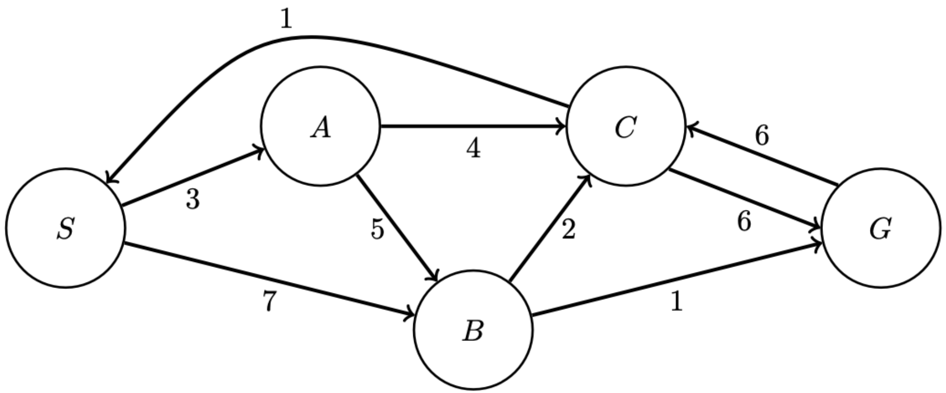


Constraint 1: path starts at S
Two nodes going out of S: A and B



Constraint 1: path starts at S

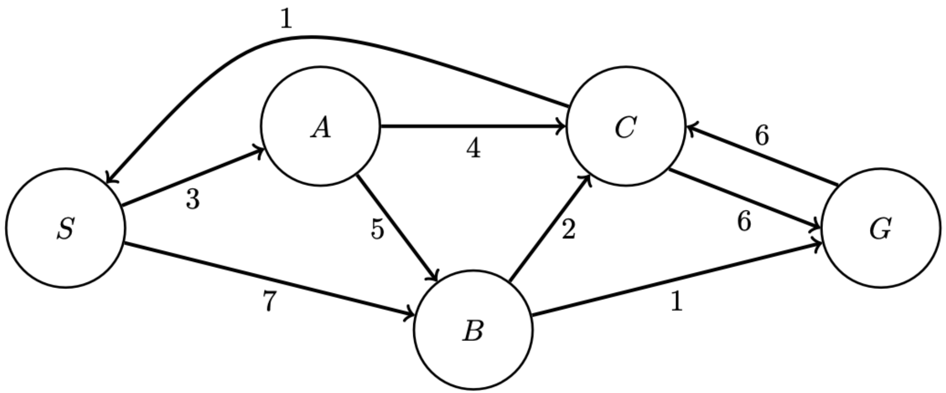
Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1



Constraint 1: path starts at S

Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1

$x_{S \to A} + x_{S \to B} = 1$

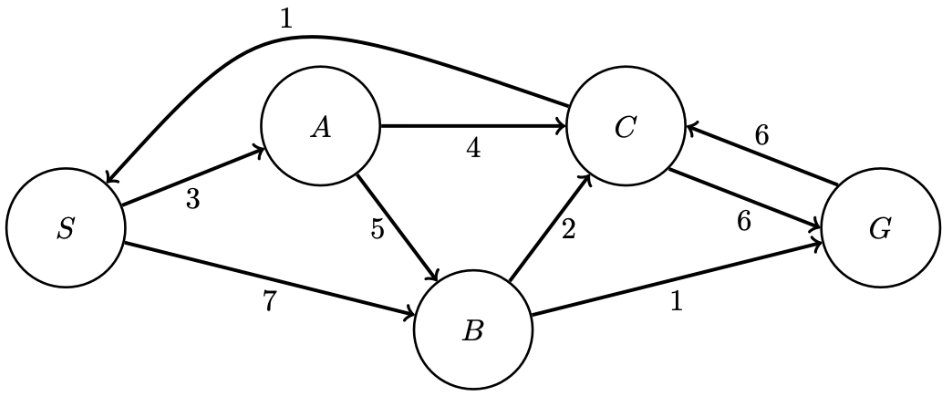


Constraint 1: path starts at S

Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1

$x_{S \to A} + x_{S \to B} = 1$

Inequality form: $x_{S \to A} + x_{S \to B} <= 1$ and $-x_{S \to A} - x_{S \to B} <= -1$



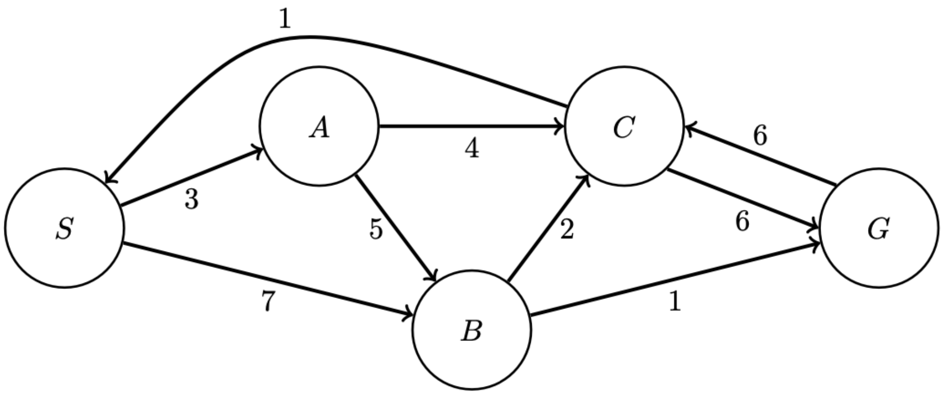
Constraint 1: path starts at S

Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1

$x_{S \to A} + x_{S \to B} = 1$

Inequality form: $x_{S \to A} + x_{S \to B} <= 1$ and $-x_{S \to A} - x_{S \to B} <= -1$

One node going into S: C



Constraint 1: path starts at S

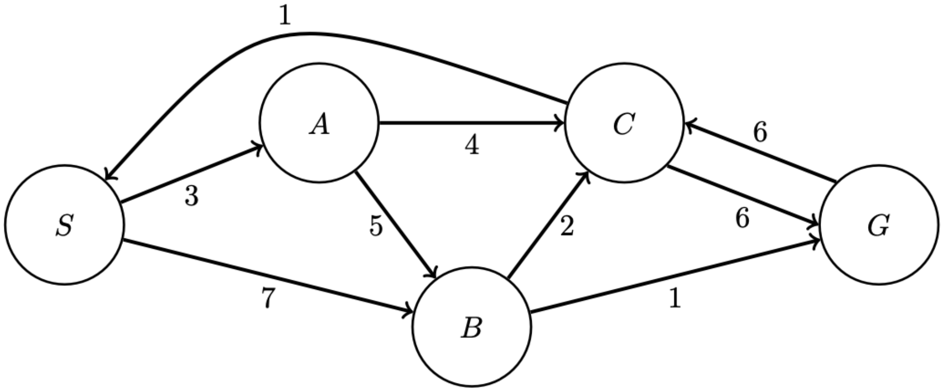
Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1

$x_{S \to A} + x_{S \to B} = 1$

Inequality form: $x_{S \to A} + x_{S \to B} <= 1$ and $-x_{S \to A} - x_{S \to B} <= -1$

One node going into S: C

$x_{C \to S} = 0$



Constraint 1: path starts at S

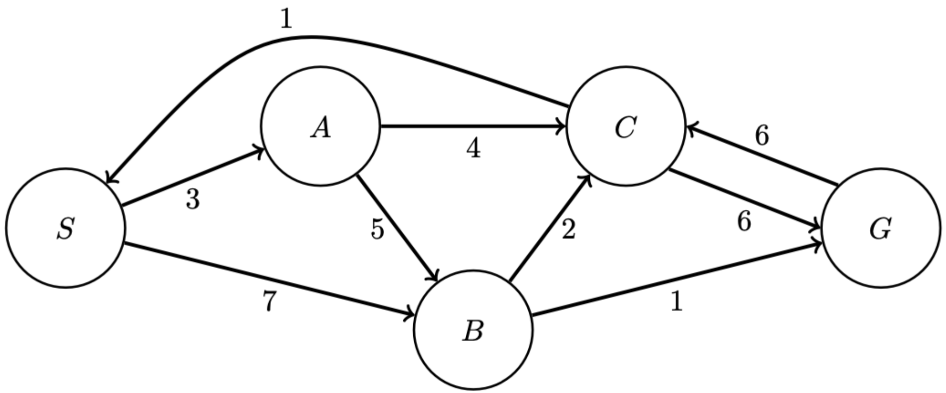
Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1

$x_{S \to A} + x_{S \to B} = 1$

Inequality form: $x_{S \to A} + x_{S \to B} <= 1$ and $-x_{S \to A} - x_{S \to B} <= -1$

One node going into S: C

$x_{C \to S} <= 0$ and $-x_{C \to S} <= 0$



Constraint 1: path starts at S

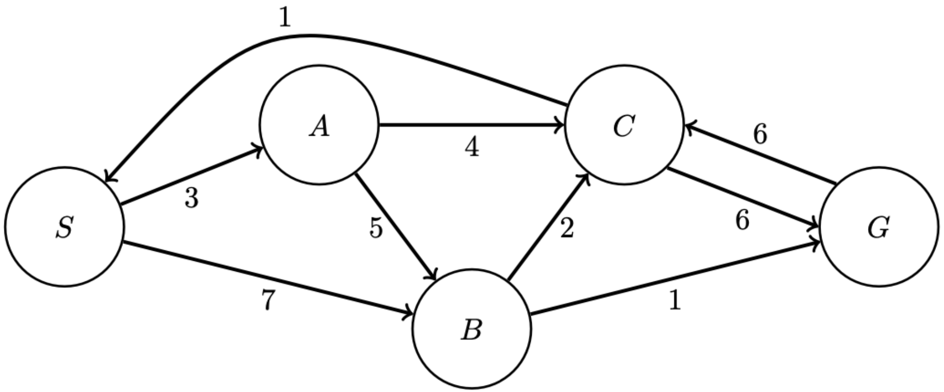
Two nodes going out of S: A and B → either $x_{S \to A}$ or $x_{S \to B}$ must be 1

$x_{S \to A} + x_{S \to B} = 1$

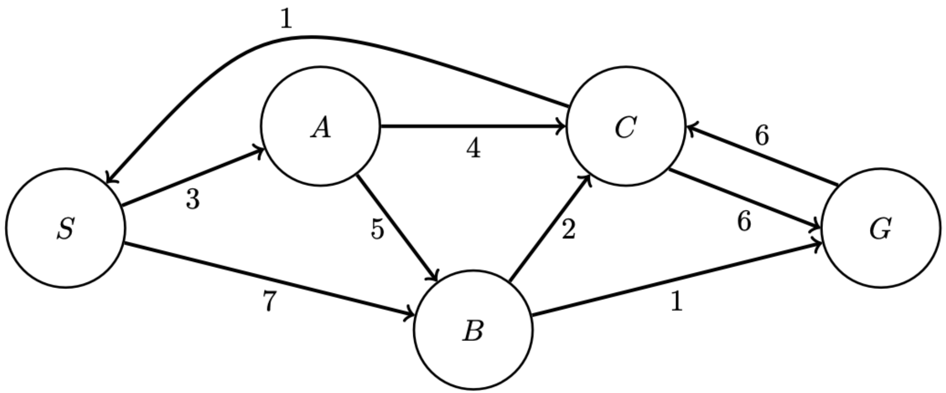
Inequality form: $x_{S \to A} + x_{S \to B} <= 1$ and $-x_{S \to A} - x_{S \to B} <= -1$

One node going into S: C

$x_{C \to S} <= 0$ and $-x_{C \to S} <= 0$



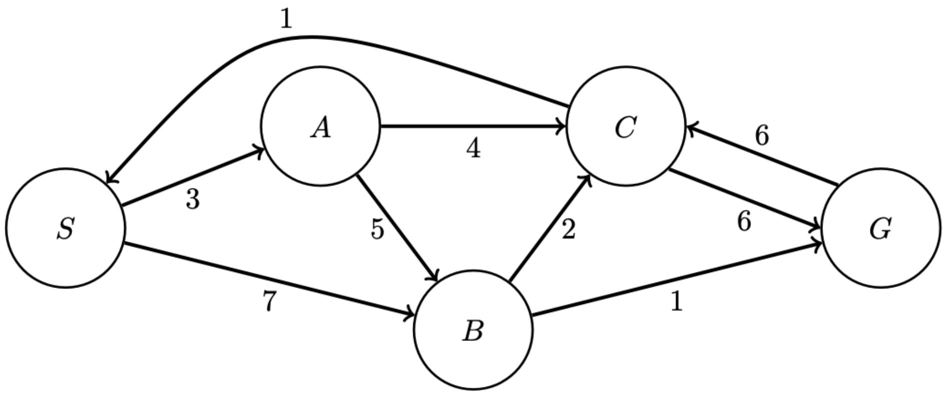
Constraint 2: path ends at G



Constraint 2: path ends at G

Two nodes going into G: C and B → either $x_{C \to G}$ or $x_{S \to B}$ must be 1

$x_{C \to G} + x_{S \to B} <= 1$ and $-x_{C \to G} - x_{S \to B} <= -1$



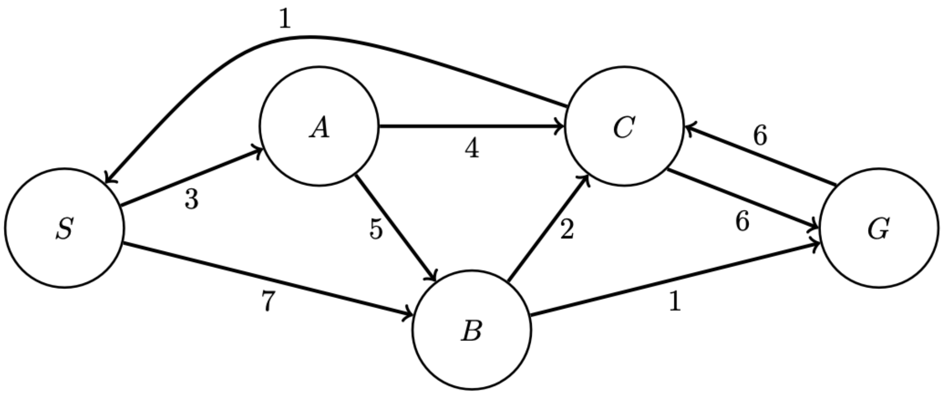
Constraint 2: path ends at G

Two nodes going into G: C and B → either $x_{C \to G}$ or $x_{S \to B}$ must be 1

$x_{C \to G} + x_{S \to B} <= 1$ and $-x_{C \to G} - x_{S \to B} <= -1$

One node coming out of G: C → $x_{G \to C}$ must be 0

$x_{C \to G} <= 0$ and $-x_{C \to G} <= 0$



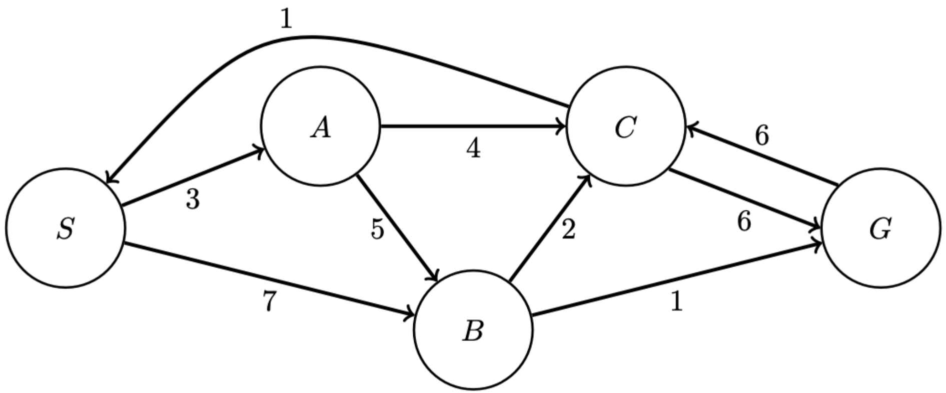
Constraint 2: path ends at G

Two nodes going into G: C and B → either $x_{C \to G}$ or $x_{S \to B}$ must be 1

$x_{C \to G} + x_{S \to B} <= 1$ and $-x_{C \to G} - x_{S \to B} <= -1$

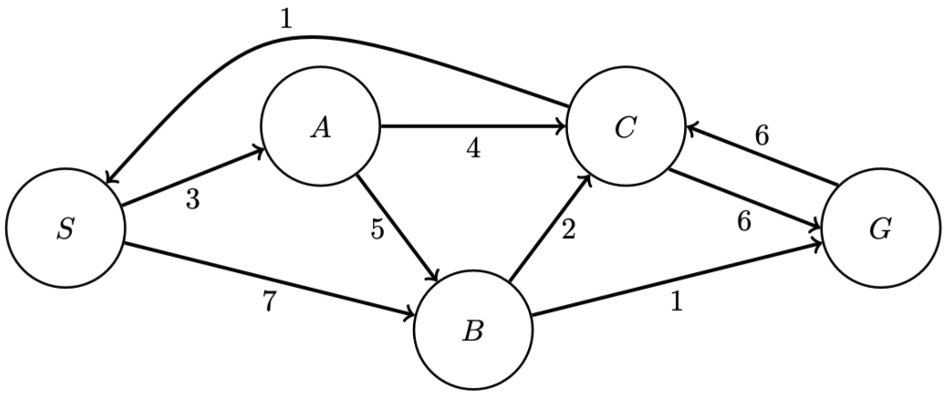
One node coming out of G: C → $x_{G \to C}$ must be 0

$x_{C \to G} <= 0$ and $-x_{C \to G} <= 0$



Constraints: need to make sure paths are valid

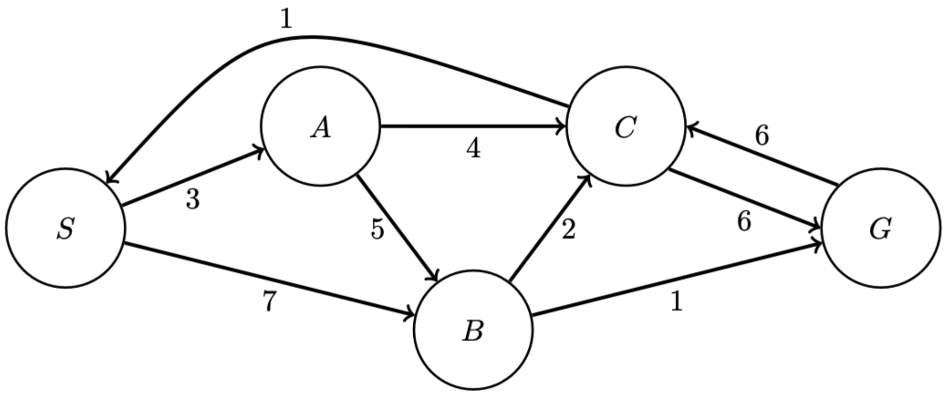
1) Ensure path starts at S - **done**
2) Ensure path ends at G - **done**



Constraints: need to make sure paths are valid

1) Ensure path starts at S - **done**
2) Ensure path ends at G - **done**

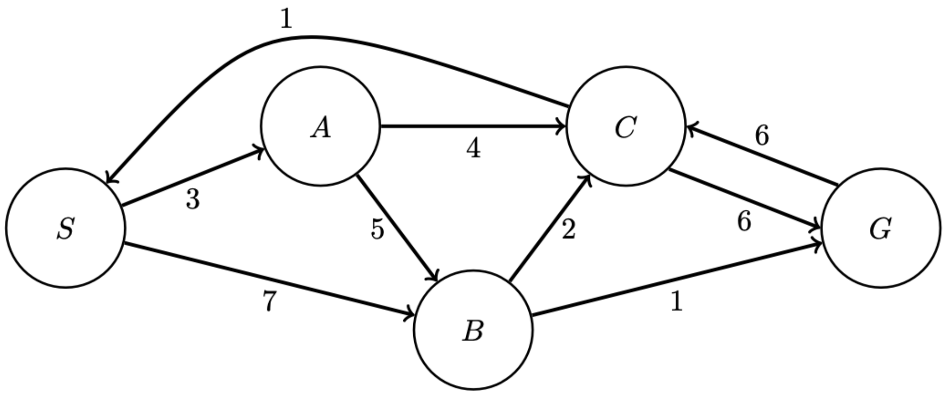
These two constraints are not enough :(



Constraints: need to make sure paths are valid

1) Ensure path starts at S - **done**
2) Ensure path ends at G - **done**

Question: 9-tuple that satisfies these constraints but does **not** represent a valid path from S to G

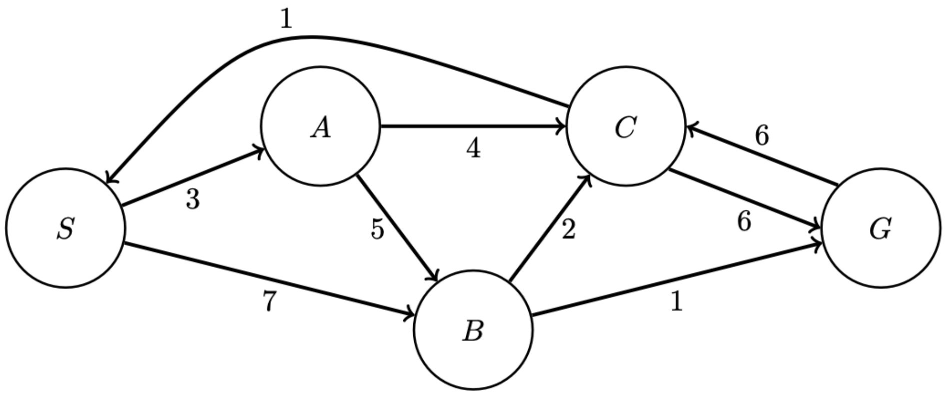


Constraints: need to make sure paths are valid

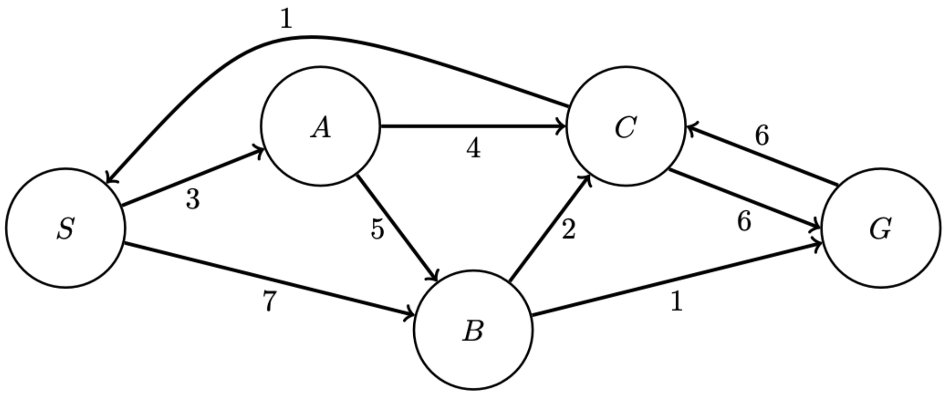
1) Ensure path starts at S - **done**
2) Ensure path ends at G - **done**

Question: 9-tuple that satisfies these constraints but does **not** represent a valid path from S to G

{S→A, C→G}: (1, 0, 0, 0, 0, 0, 0, 1, 0)

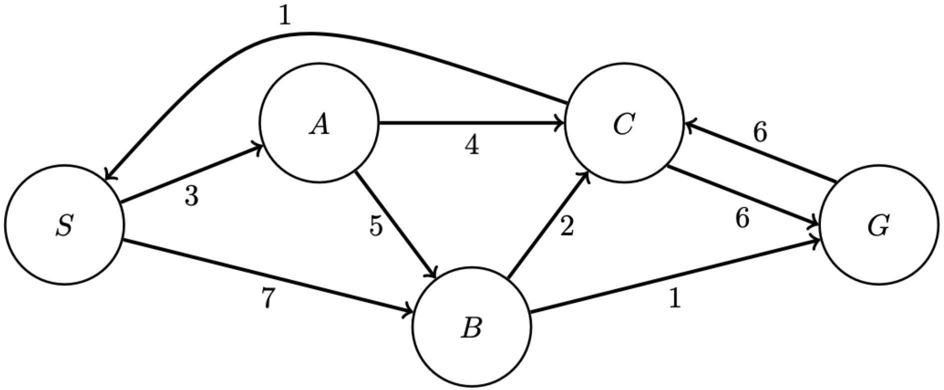


More constraints: ensure all other nodes are **non-terminal** (not start or goal)



More constraints: ensure all other nodes are **non-terminal** (not start or goal)

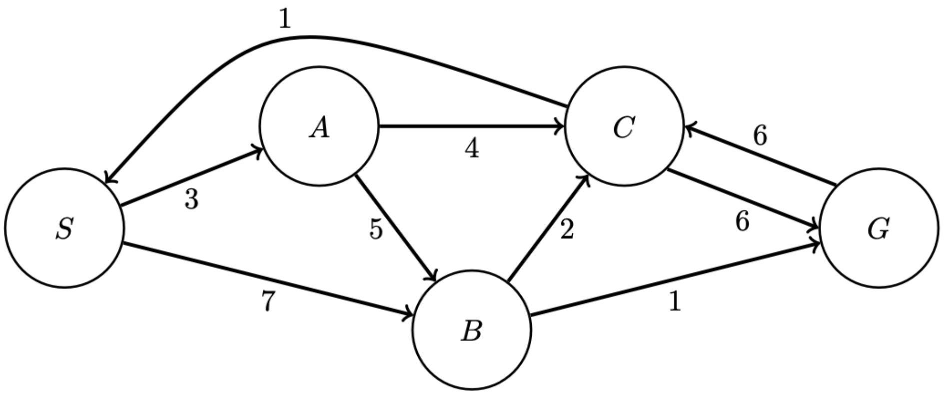
- Path can only pass through each non-terminal node at most once



More constraints: ensure all other nodes are **non-terminal** (not start or goal)

- Path can only pass through each non-terminal node at most once

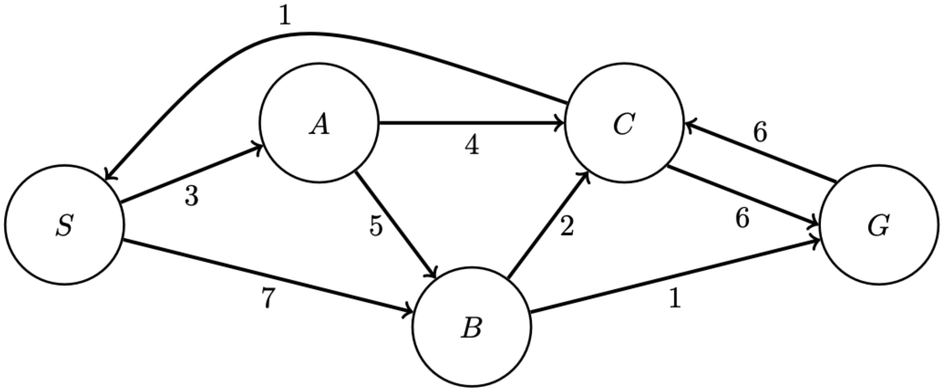
Constraint that node B can only appear on the path at most once:



More constraints: ensure all other nodes are **non-terminal** (not start or goal)

- Path can only pass through each non-terminal node at most once

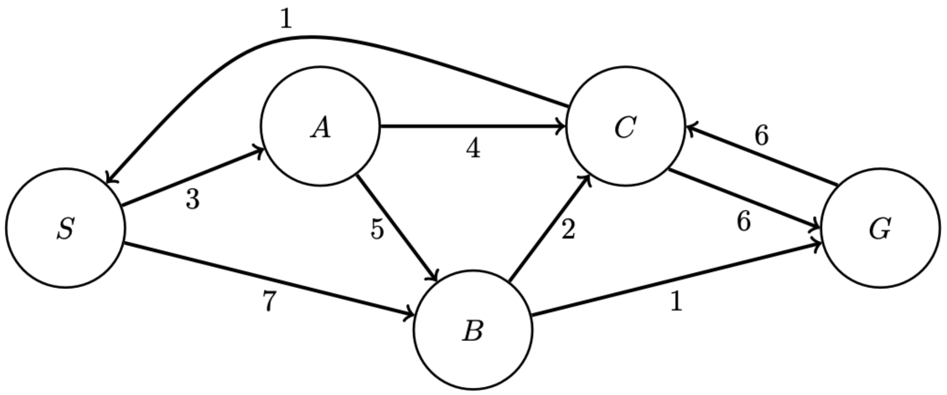
Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A



More constraints: ensure all other nodes are **non-terminal** (not start or goal)

- Path can only pass through each non-terminal node at most once

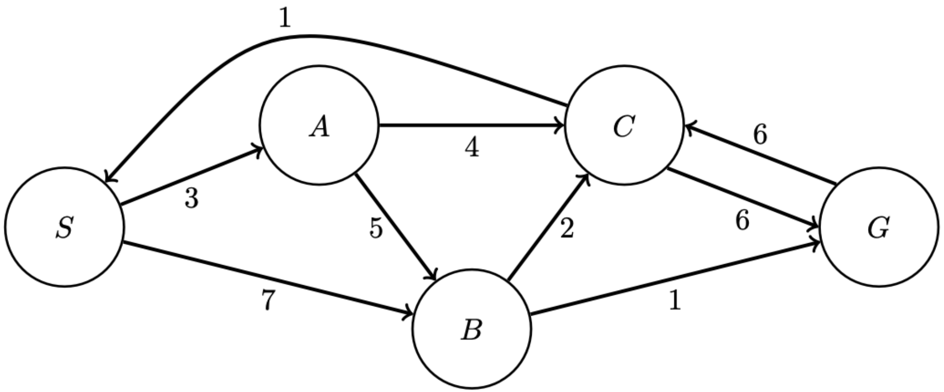
Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S \to B}$ or $x_{A \to B}$ must be 1, but both cannot be 1



More constraints: ensure all other nodes are **non-terminal** (not start or goal)

- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S \to B}$ or $x_{A \to B}$ must be 1, but both cannot be 1
$x_{S \to B} + x_{A \to B} <= 1$

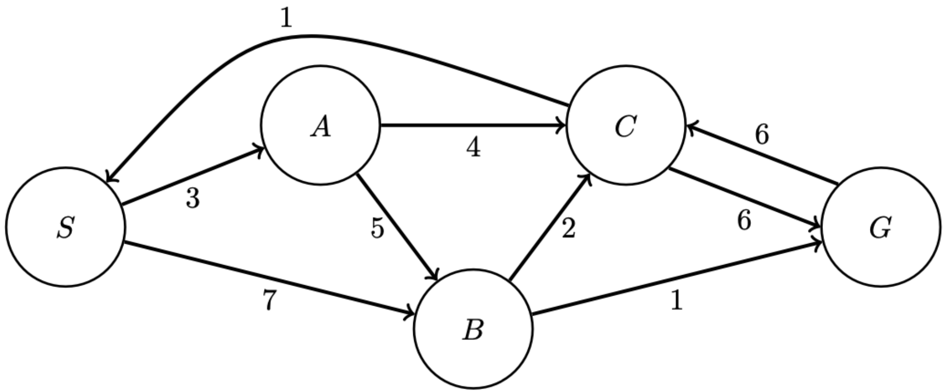


More constraints: ensure all other nodes are **non-terminal** (not start or goal)

- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S\to B}$ or $x_{A\to B}$ must be 1, but both cannot be 1
$x_{S\to B} + x_{A\to B} <= 1$

Two nodes coming out of B: C, G → either $x_{B\to C}$ or $x_{B\to G}$ must be 1, but both cannot be 1

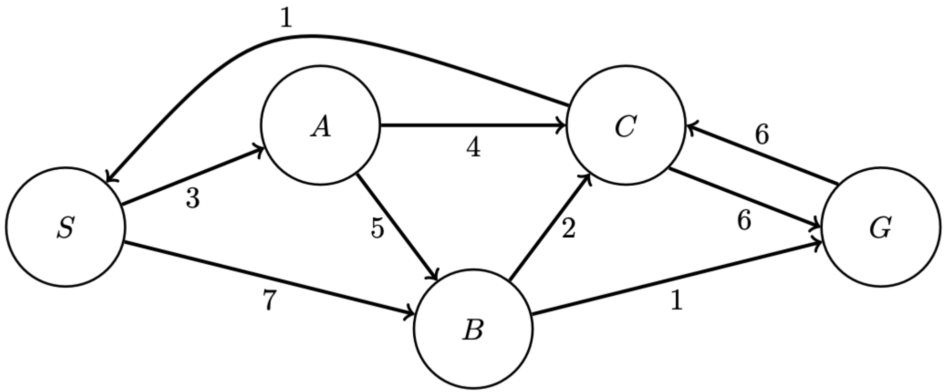


More constraints: ensure all other nodes are **non-terminal** (not start or goal)

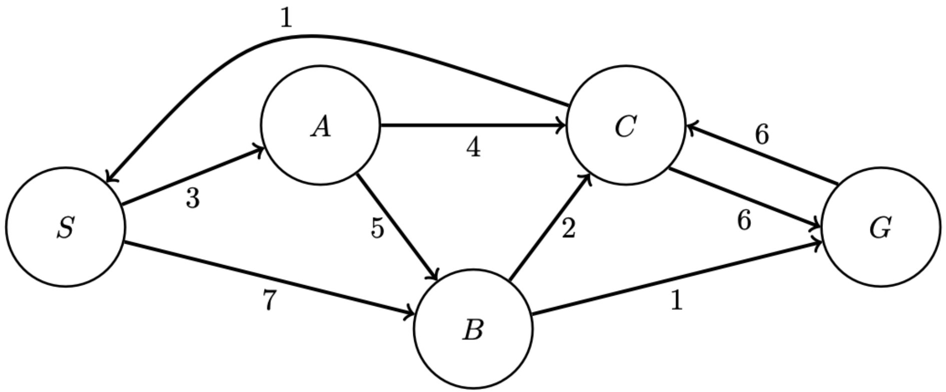
- Path can only pass through each non-terminal node at most once

Constraint that node B can only appear on the path at most once:
Two nodes going into B: S, A → either $x_{S\to B}$ or $x_{A\to B}$ must be 1, but both cannot be 1
$x_{S\to B} + x_{A\to B} <= 1$

Two nodes coming out of B: C, G → either $x_{B\to C}$ or $x_{B\to G}$ must be 1, but both cannot be 1
$x_{B\to C} + x_{B\to G} <= 1$

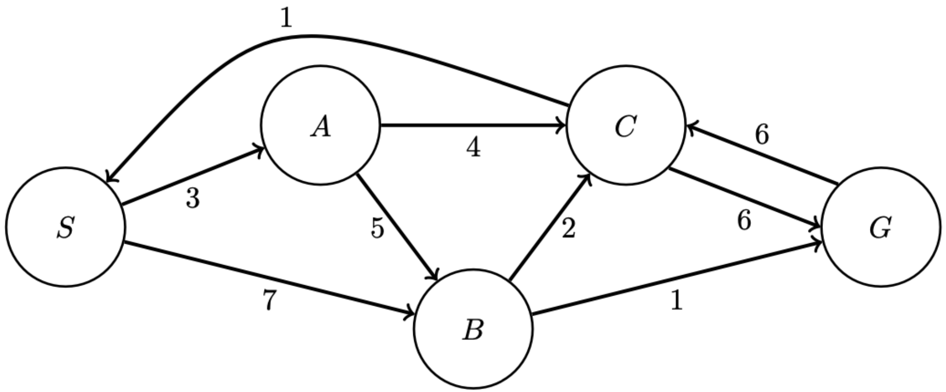


More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)



More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

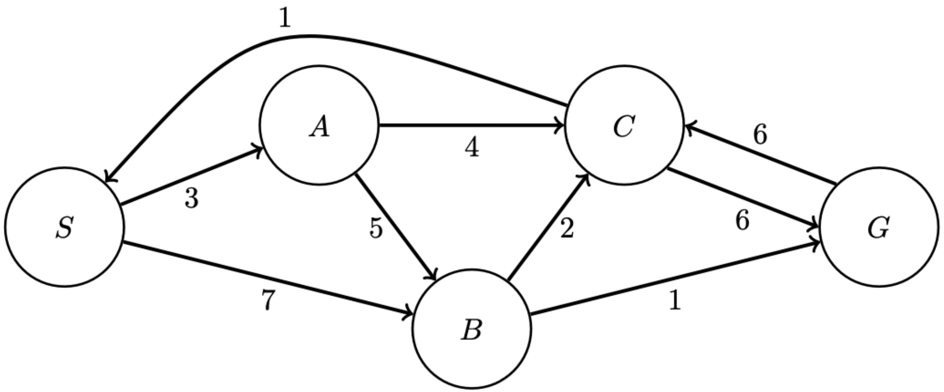
Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)



More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)

$x_{S \to B} + x_{A \to B} = x_{B \to C} + x_{B \to G}$

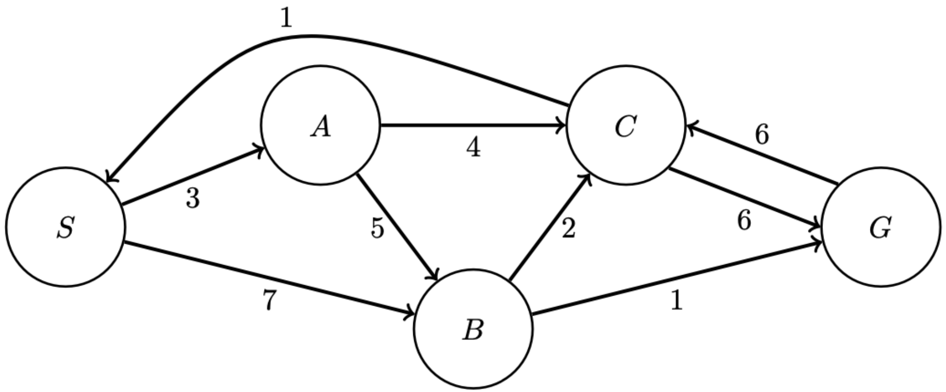


More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)

$x_{S \to B} + x_{A \to B} <= x_{B \to C} + x_{B \to G}$

$x_{S \to B} + x_{A \to B} >= x_{B \to C} + x_{B \to G}$

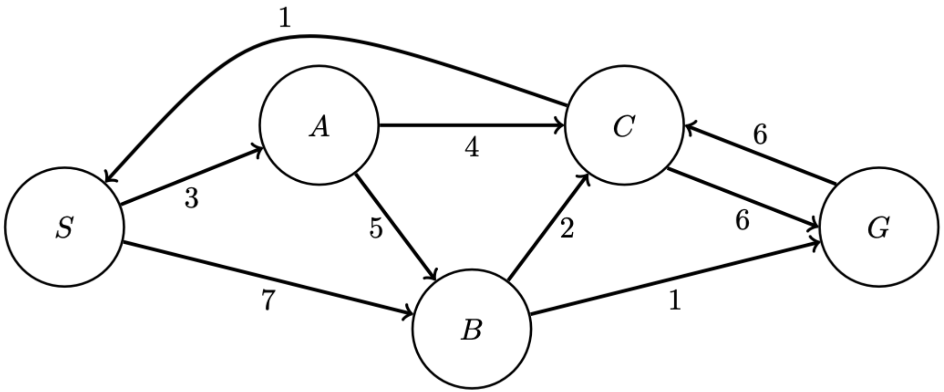


More constraints: If there is an edge to B, then there must be an edge out of B (otherwise, B is either a dead end or a start)

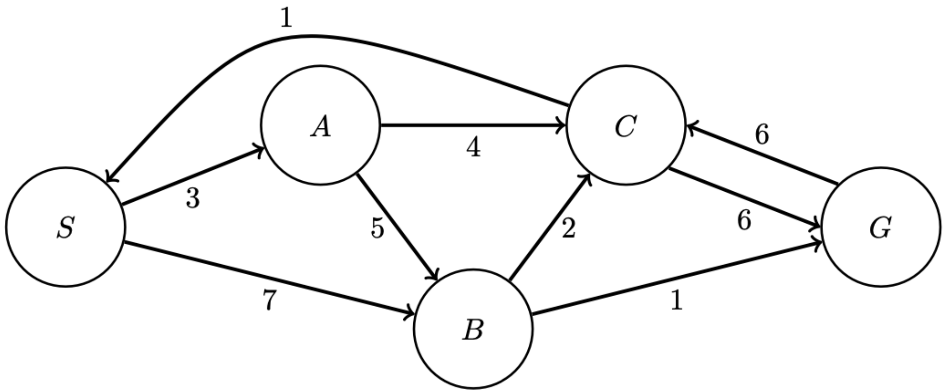
Idea: number of edges into B = number of edges out of B (we already constrained that you can only have one of those edges)

$x_{S \to B} + x_{A \to B} - x_{B \to C} - x_{B \to G} <= 0$

$-x_{S \to B} - x_{A \to B} + x_{B \to C} + x_{B \to G} <= 0$

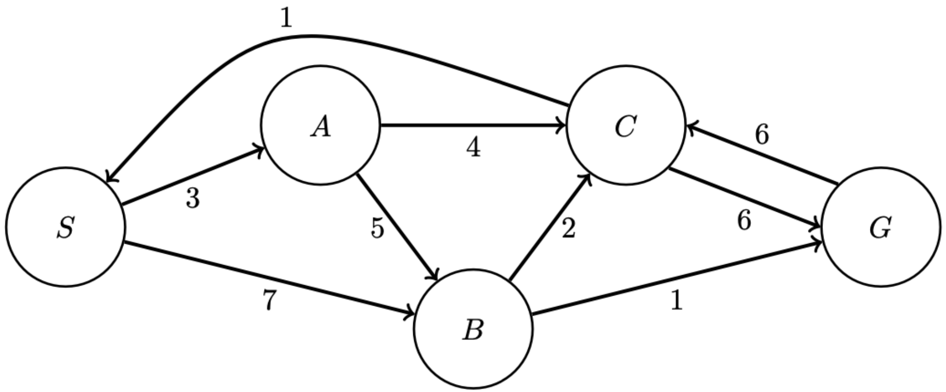


Objective function:



Objective function:

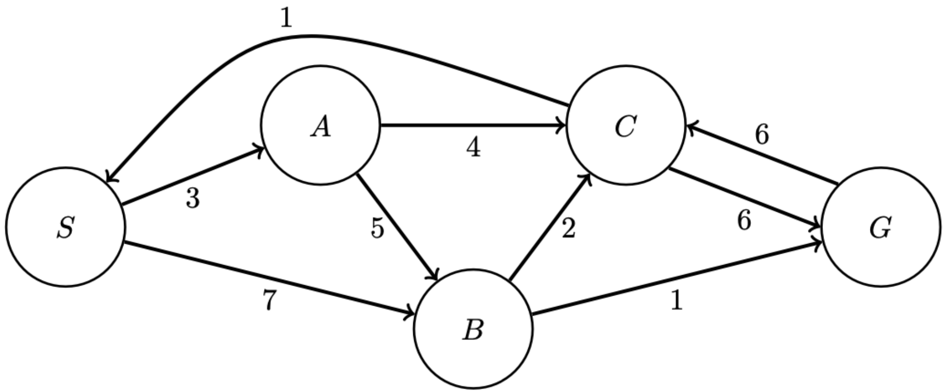
Idea: coefficient for each edge is the cost of that edge



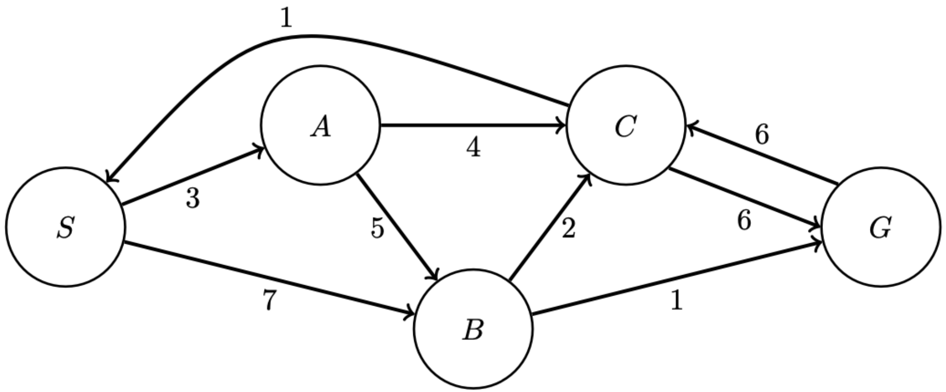
Objective function:

Idea: coefficient for each edge is the cost of that edge

$3x_{S\to A} + 7x_{S\to B} + 5x_{A\to B} + 4x_{A\to C} + 2x_{B\to C} + 1x_{B\to G} + 1x_{C\to S} + 6x_{C\to G} + 6x_{G\to C}$

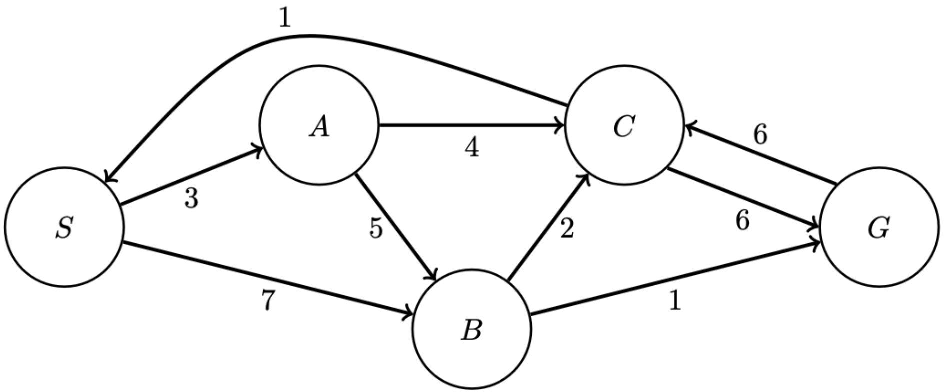


Still not enough to ensure a valid path :(



Still not enough to ensure a valid path :(

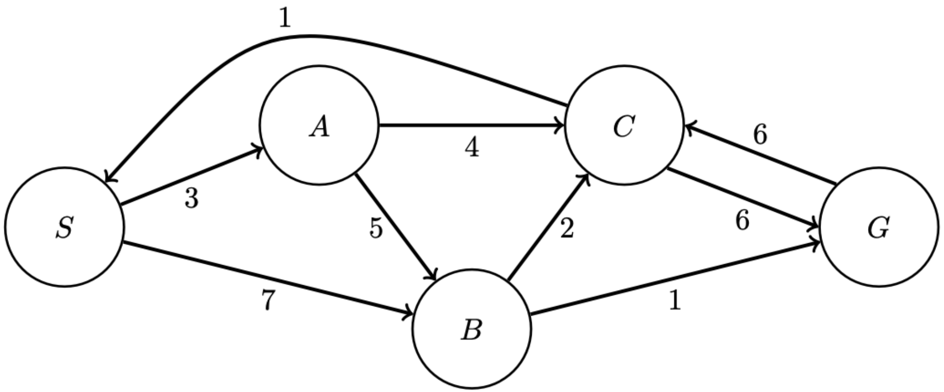
Counterexample:



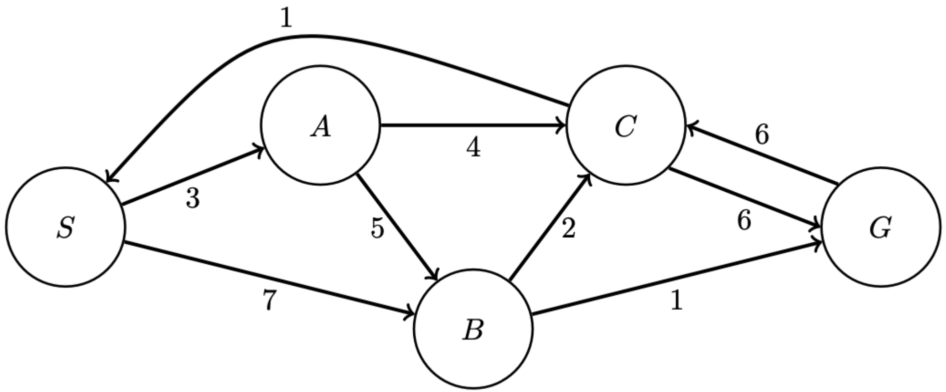
Still not enough to ensure a valid path :(

Counterexample:

Idea: anything with a loop outside the path is still allowed by our constraints

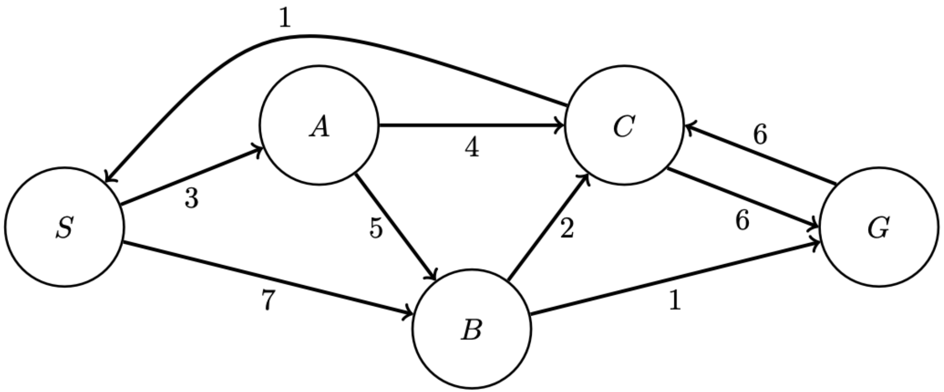


How can we fix this?



How can we fix this?

Answer: we don’t have to :)



How can we fix this?

Answer: we don't have to :)

Idea: If we have an extra cycle, that would just increase the total path cost. Because we are trying to minimize cost, this would only hurt us, so we wouldn't return such a solution anyway.

Cost-Based Search as IP

- Now let's put everything together, and define the following search algorithm
  - First convert the search problem into the IP representation
  - Then run an IP-solver (which is complete and optimal) on the representation
  - Reconstruct the path from start to goal by getting all the ones in the variables

- Is this is complete?
- Is this is optimal?

Cost-Based Search as IP

- Now let's put everything together, and define the following search algorithm
  - First convert the search problem into the IP representation
  - Then run an IP-solver (which is complete and optimal) on the representation
  - Reconstruct the path from start to goal by getting all the ones in the variables

- Is this is complete? Yes
- Is this is optimal?

Cost-Based Search as IP

- Now let's put everything together, and define the following search algorithm
  - First convert the search problem into the IP representation
  - Then run an IP-solver (which is complete and optimal) on the representation
  - Reconstruct the path from start to goal by getting all the ones in the variables

- Is this is complete? Yes
- Is this is optimal? Yes

Take Home Messages

- Cost-based search can be expressed, and solved with IP
- IP is very expressive, we can do many interesting things with it

- Want some more?

    Minimax as IP!!!   (Bonus question on the course website)