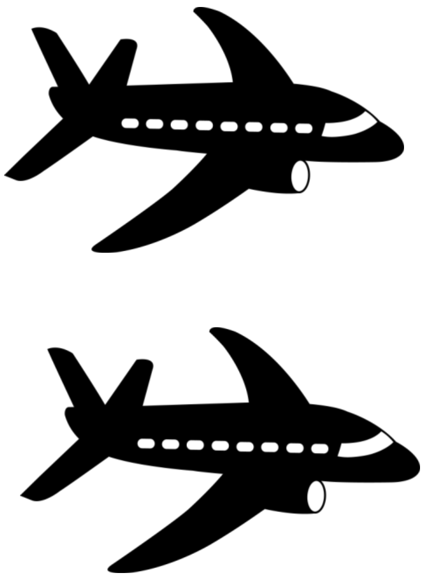
Airline Scheduling



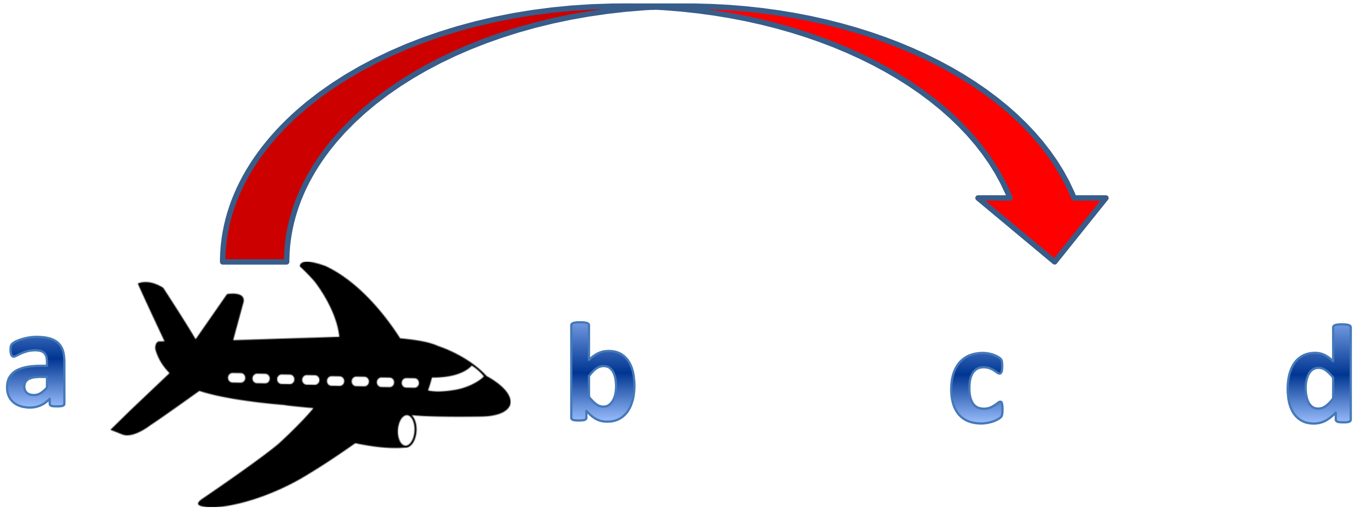
Problem

- Suppose you're in charge of managing a fleet of airplanes and you'd like to create a flight schedule for them.

- *(1) Boston (depart 6 A.M.) – Washington DC (arrive 7 A.M.)*
- *(2) Philadelphia (depart 7 A.M.) – Pittsburgh (arrive 8 A.M.)*
- *(3) Washington DC (depart 8 A.M.) – Los Angeles (arrive 11 A.M.)*
- *(4) Philadelphia (depart 11 A.M.) – San Francisco (arrive 2 P.M.)*
- *(5) San Francisco (depart 2:15 P.M.) – Seattle (arrive 3:15 P.M.)*
- *(6) Las Vegas (depart 5 P.M.) – Seattle (arrive 6 P.M.)*

Problem

- Is it possible to use a single plane for a flight segment *i, and then later for* a flight segment *j?*



Problem

- Is it possible to use a single plane for a flight segment *i, and then later for* a flight segment *j?*

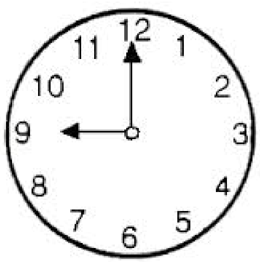
YES

- (a) the destination of *i is the same as the origin of j, and there's enough time* to perform maintenance on the plane in between;

or

- (b) you can add a flight segment in between that gets the plane from the destination of *i to the origin of j with adequate time in between.*

Problem



Bangalore to Mumbai

Can I use the same Plane??

Mumbai to Agra

Ahmedabad to Agra

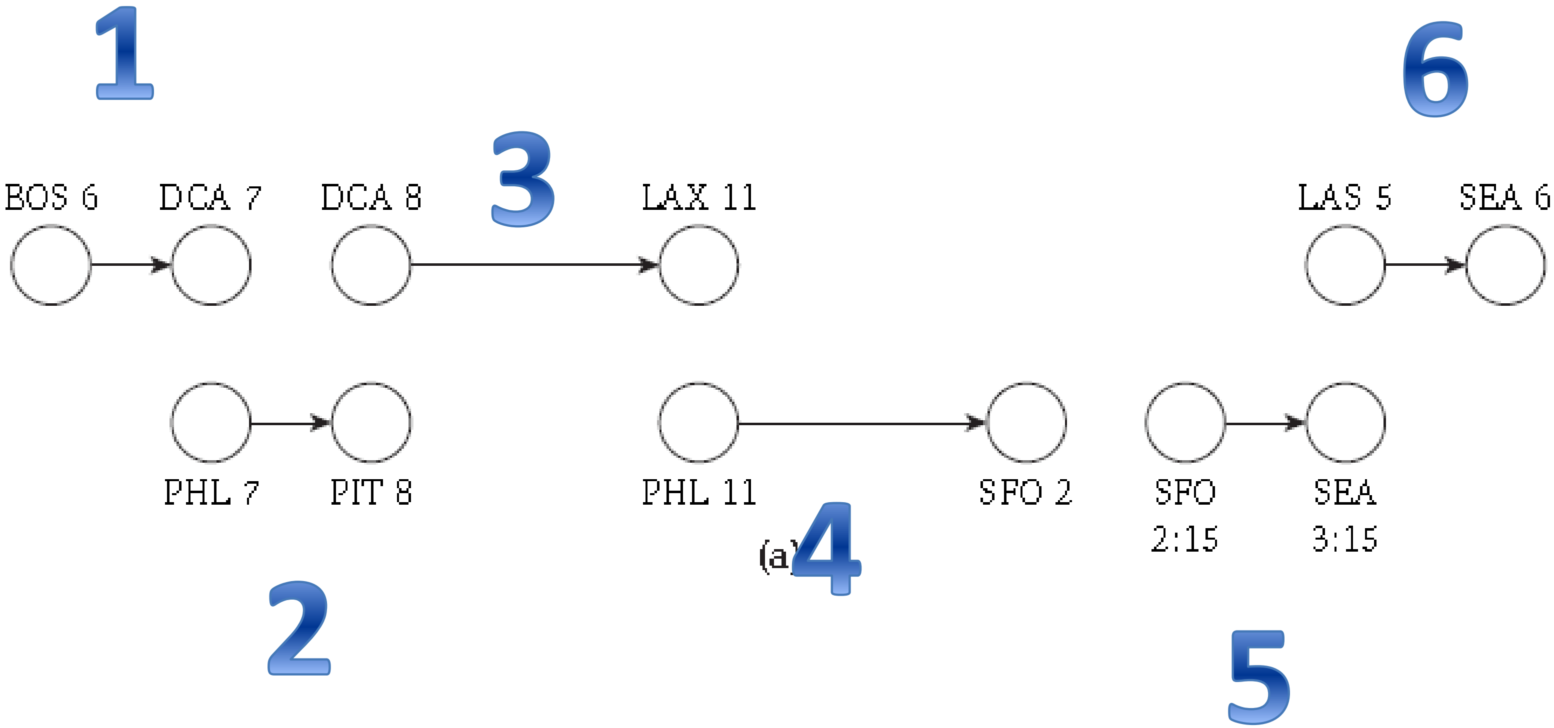
Problem

- Is it possible to use a single plane for a flight segment *i, and then later for* a flight segment *j?*

YES

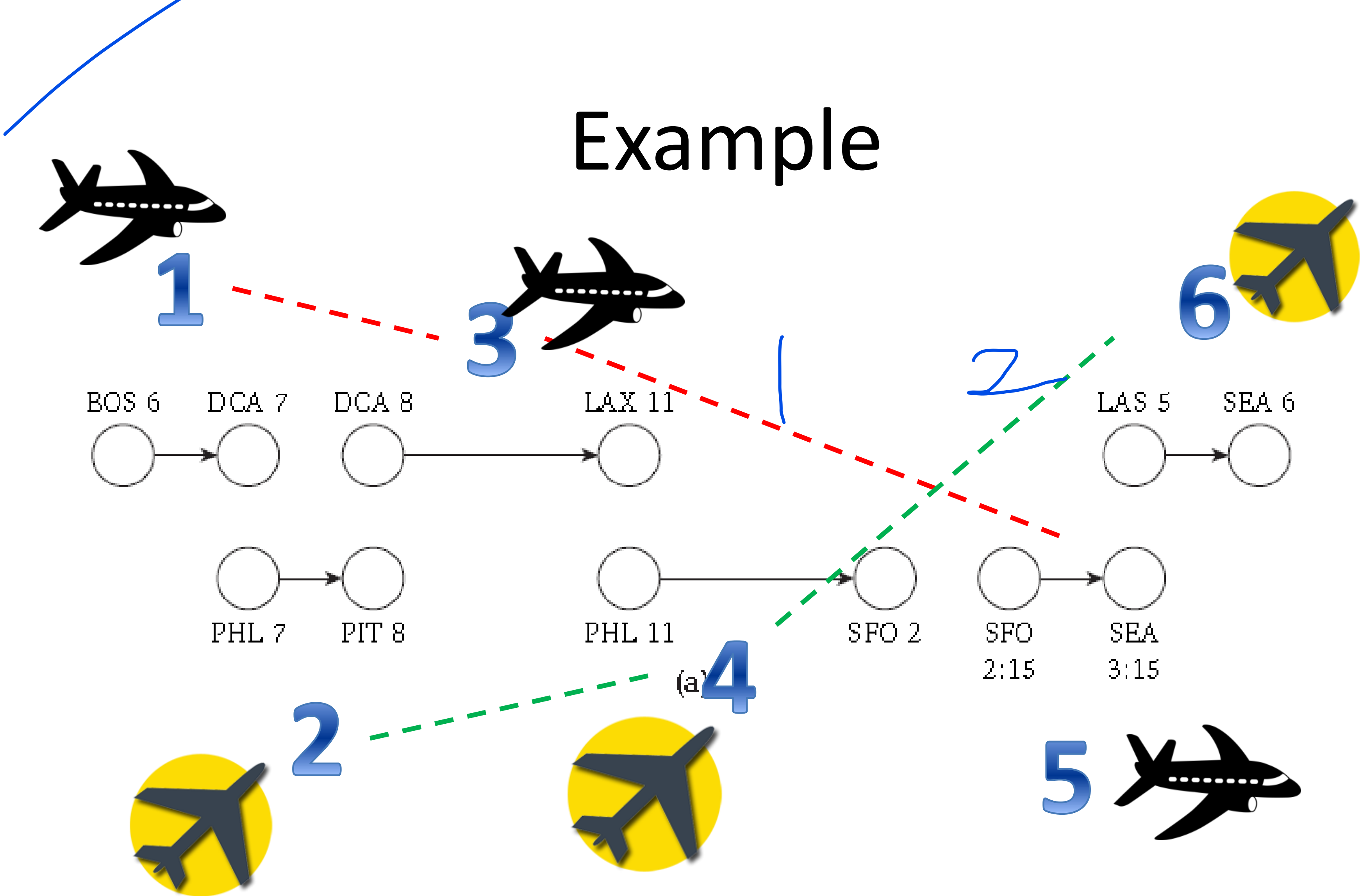
- **Goal**: Optimal number of planes needed for the given flight segements.

Example



How many planes are needed to satisfy this fleet segment??

Example



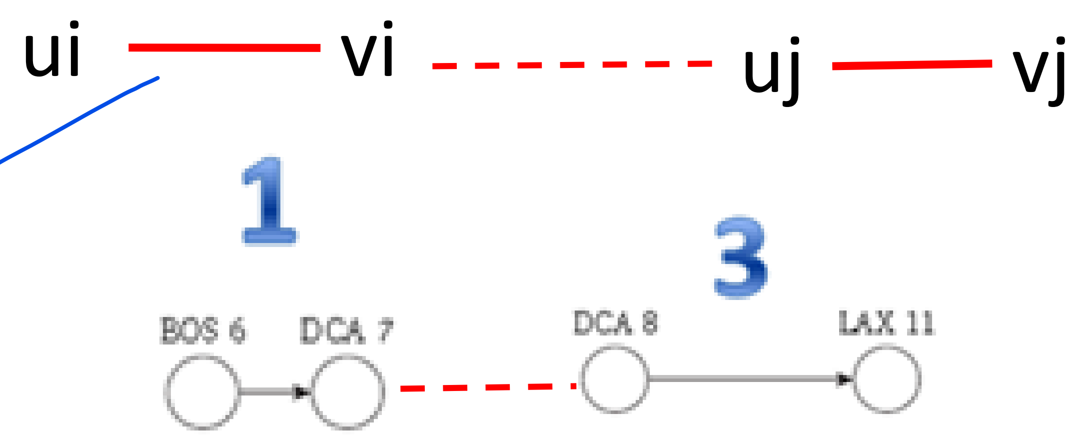
How many planes are needed to satisfy this fleet segment??

Designing the algorithm

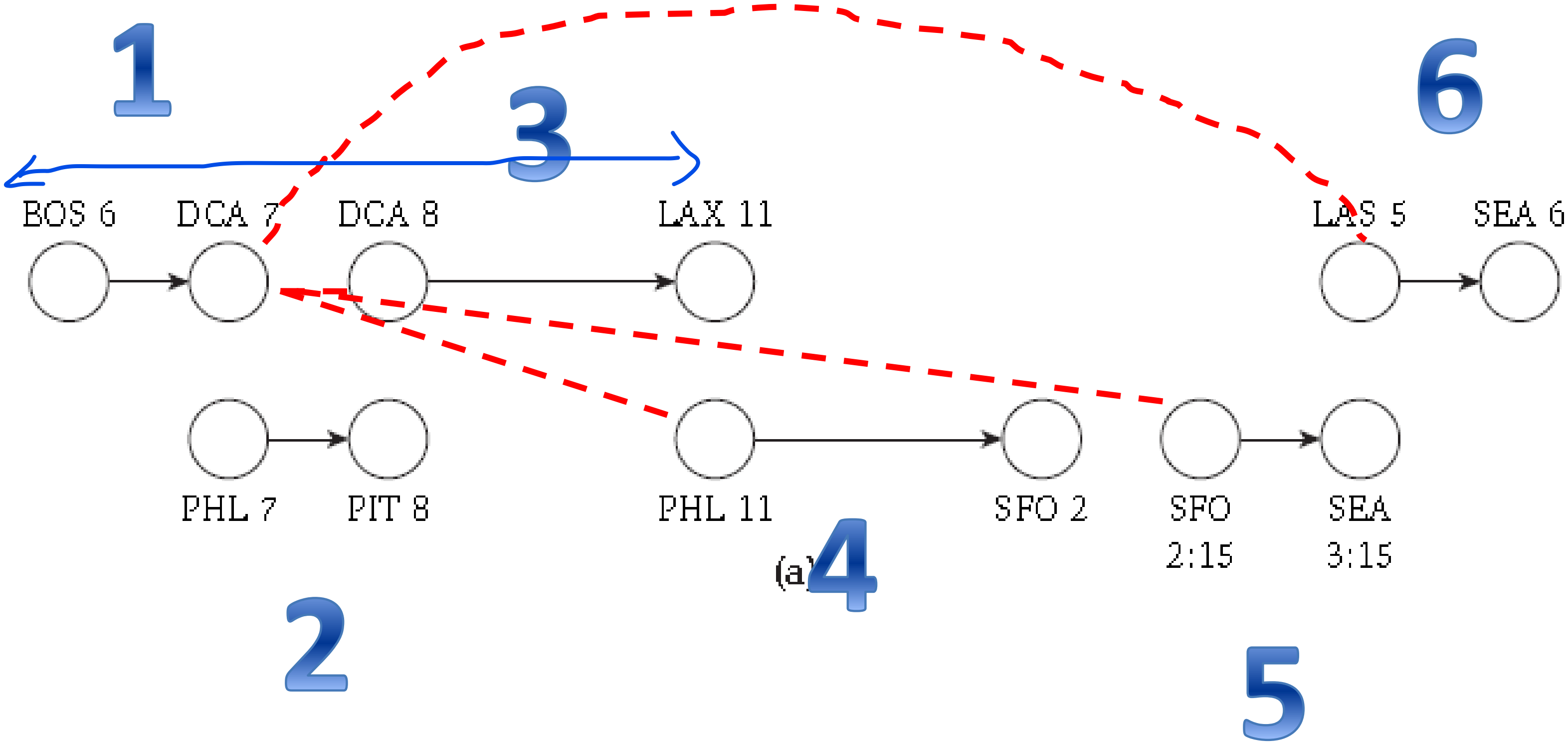
- We will have an edge for each flight, and upper and lower capacity bounds of 1 on these edges to require that exactly one unit of flow crosses this edge. In other words, each flight must be served by one of the planes.

Designing the algorithm

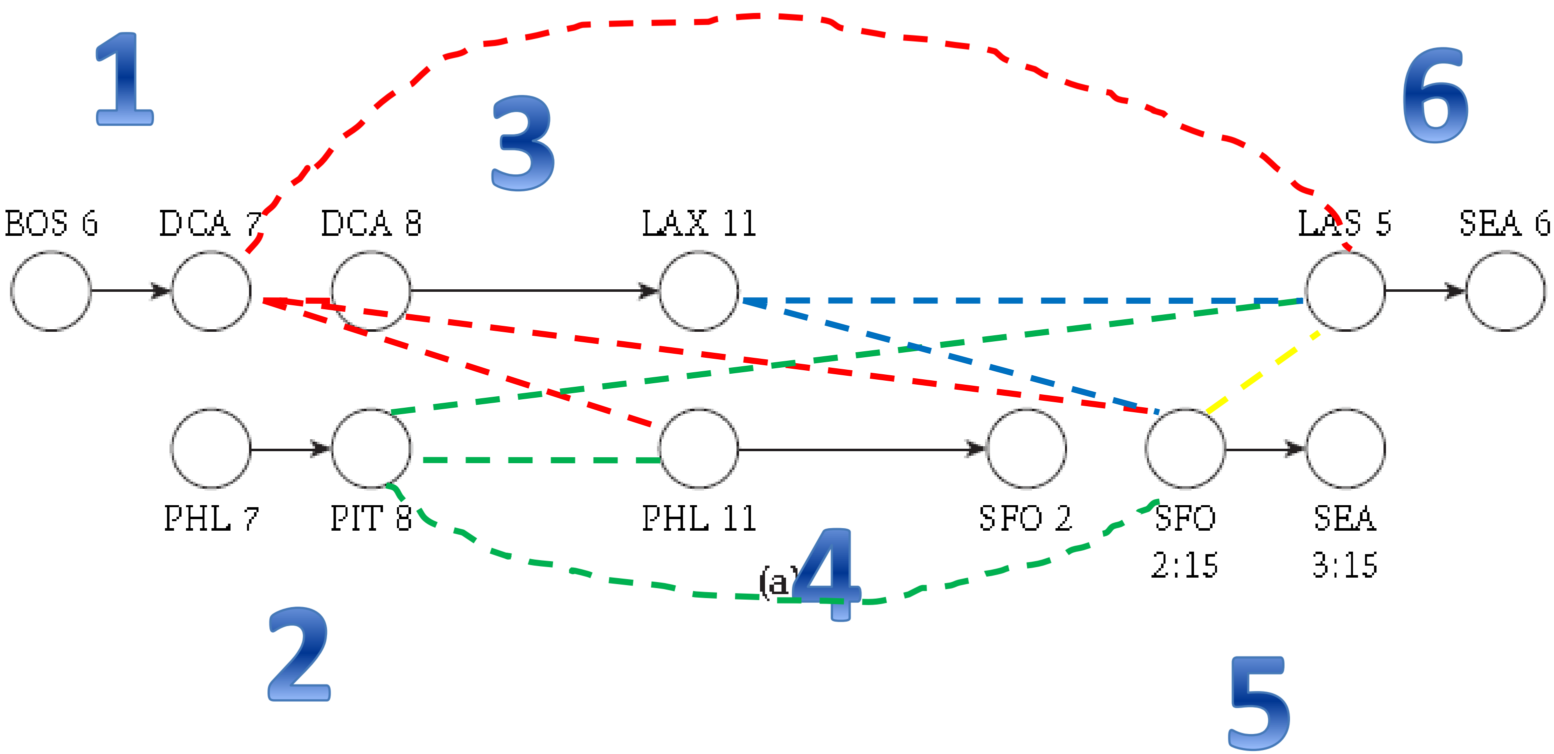
- If *(ui , vi)* is the edge representing flight *i, and (uj , vj) is the edge representing flight j,*
- and flight *j is reachable from flight i, then we will have an edge from vi to uj with capacity 1.*



Designing the algorithm-Example

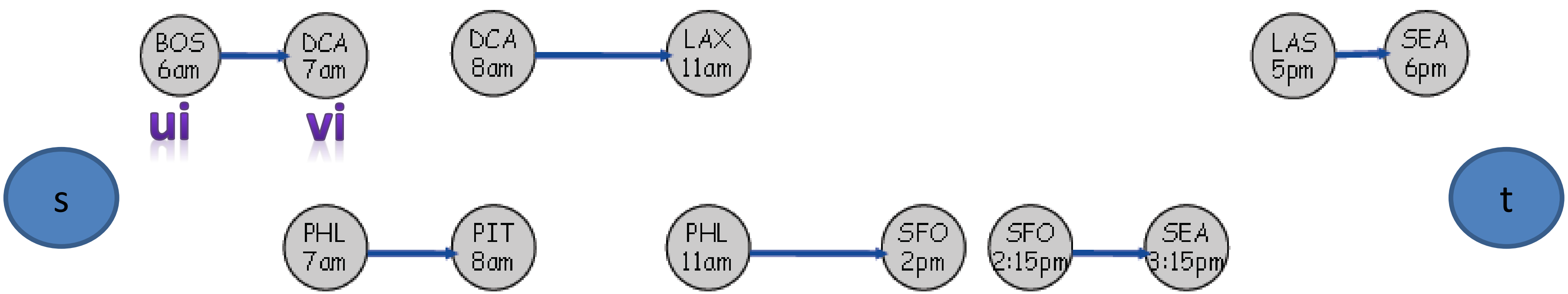


Designing the algorithm-Example



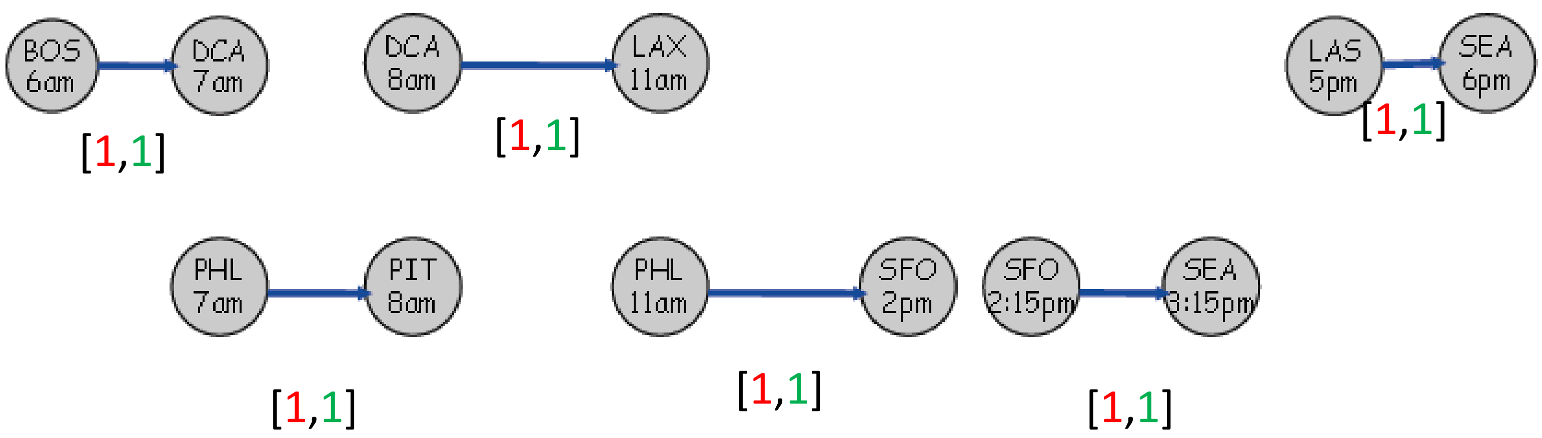
Designing the algorithm

- The node set of the underlying graph *G is* defined as follows.
  - For each flight *i, the graph G will have the* ***two nodes ui and vi.***
  - *G will also have a* ***distinct source node s and sink node t.***



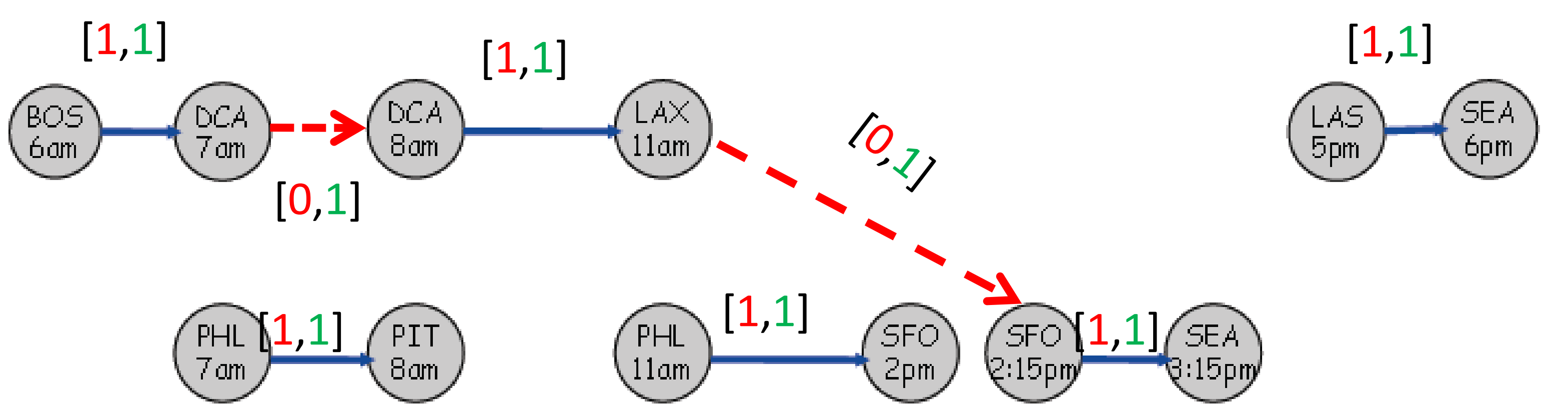
Designing the algorithm

- The edge set of *G is defined as follows.*
  - For each *i*, ***there is an edge (ui , vi)*** *with a lower bound of 1 and a capacity* of 1. *(Each flight on the list must be served.)*



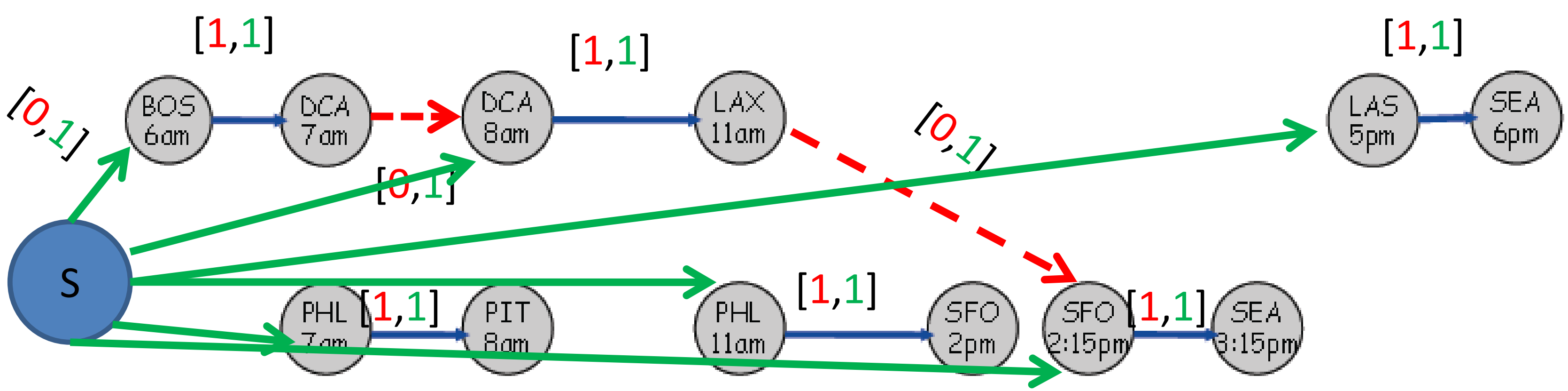
Designing the algorithm

- The edge set of *G is defined as follows.*
  - For each *i and j so that **flight j is reachable from flight i**, there is an edge **(vi , uj)** with a lower bound of 0 and a capacity of 1. (The same plane can perform flights i and j.)*



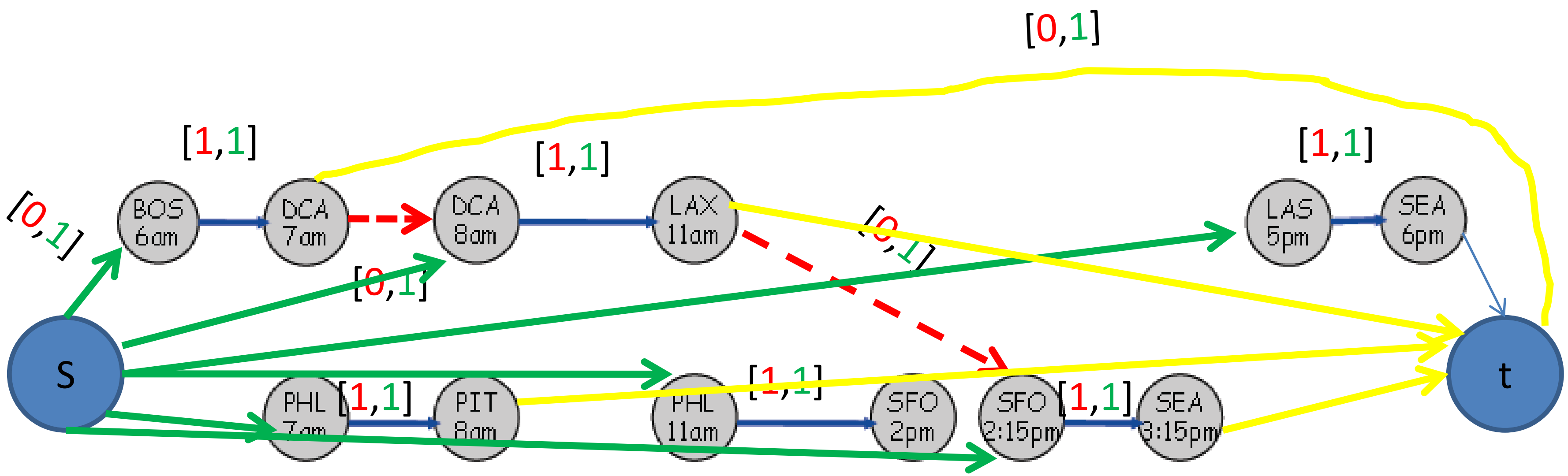
Designing the algorithm

- The edge set of *G is defined as follows.*
  - For each *i, there is an **edge (s, ui)** with a lower **bound of 0 and a capacity** **of 1**. (Any plane can begin the day with flight i.)*



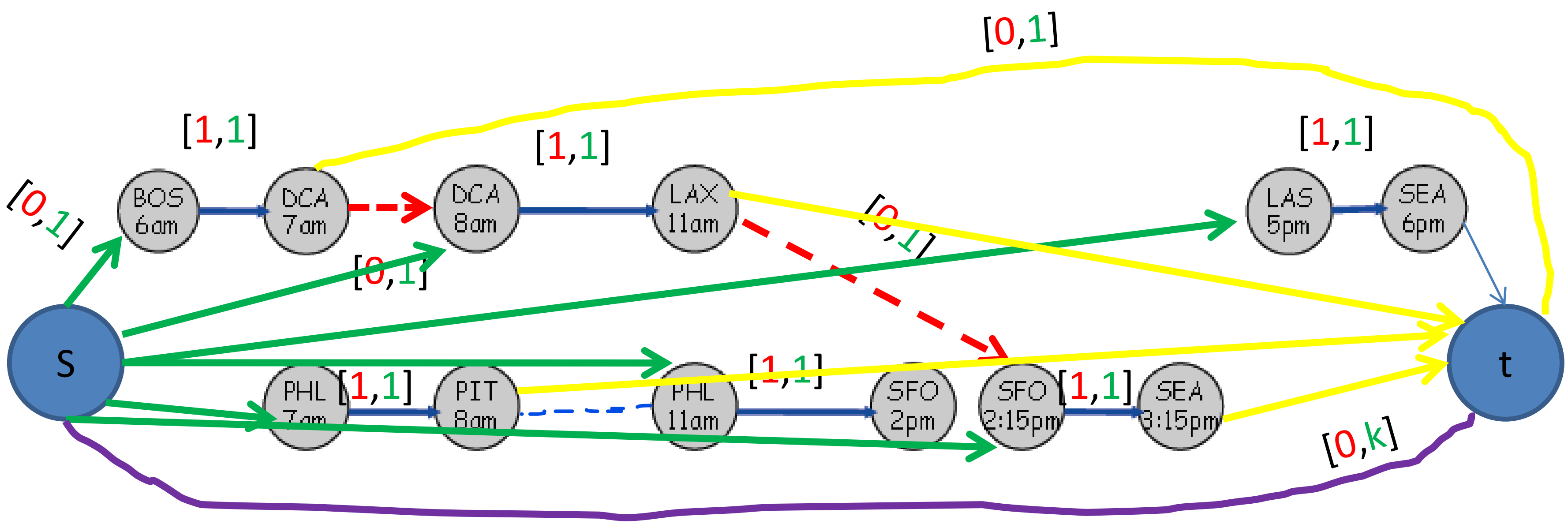
Designing the algorithm

- The edge set of *G is defined as follows.*
  - For each *j, there is an* ***edge (vj, t)*** *with a lower bound of 0 and a capacity* of 1. *(Any plane can end the day with flight j.)*



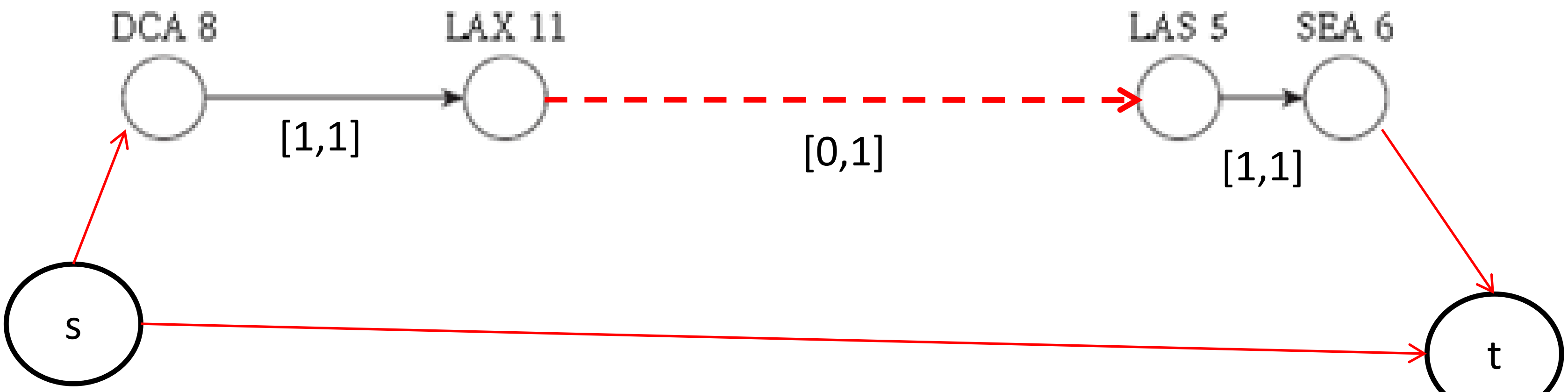
Designing the algorithm

- The edge set of *G is defined as follows.*
  - There is an **edge *(s, t)*** *with lower bound 0 and capacity k. (If we have extra planes, we don't need to use them for any of the flights.)*



Analysis

There is a way to perform all flights using at most k planes if and only if there is a feasible circulation in the network G.

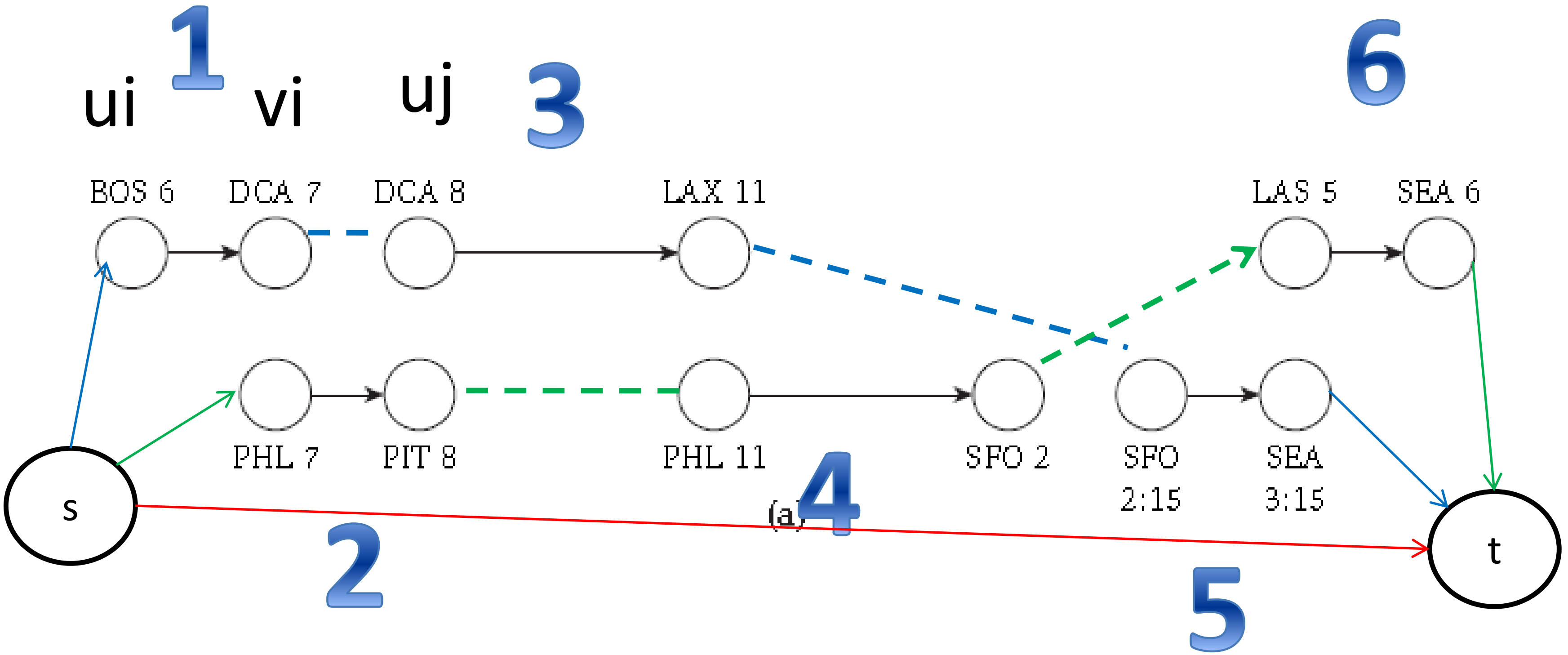


Feasible circulation means supply is equal to the demand

Proof

- Consider a feasible circulation in the network *G.*
- Suppose that *k' units of flow are sent on edges other than (s, t). Since all other edges* have a capacity bound of 1, and the circulation is integer-valued, each such edge that carries flow has exactly one unit of flow on it.
- This flow can be converted to collection of paths.

Proof


ui
1
vi
uj
3
6
BOS 6
DCA 7
DCA 8
LAX 11
LAS 5
SEA 6
PHL 7
PIT 8
PHL 11
4
SFO 2
SFO 2:15
SEA 3:15
(a)
s
t
2
5

Proof

- Consider an edge *(s, ui) that carries one unit of flow. It follows by conservation that (ui , vi)* carries one unit of flow, and that there is a unique edge out of *vi that carries* one unit of flow.
- If we continue in this way, we construct a path *P from s to t, so that each edge on this path carries one unit of flow.*
- *We can apply this* construction to each edge of the form

(s, uj) carrying one unit of flow; in this way, we produce *k' paths from s to t, each consisting of edges that carry one* unit of flow.

- Now, for each path *P we create in this way, we can assign a single* plane to perform all the flights contained in this path.