CSE 417
Network Flows (pt 5)
Modeling with Min Cost Flows

UNIVERSITY *of* WASHINGTON

Reminders

> **HW6 due yesterday**
  - dynamic programming on subsets arises in practice
  - needs opt over those *to which last element can be legally added*
  - make sub-problems sufficiently fine-grained that <u>either</u> every solution to a sub-problem can legally add last <u>or</u> none can

> **HW7 is due Friday**
  - solve a scheduling problem
  - apply Ford-Fulkerson on paper

Review of last four lectures

- Defined the maximum flow problem
  - find the feasible flow of maximum value
  - flow is feasible if it satisfies edge capacity and node balance constraints
- Described the Ford-Fulkerson algorithm
  - starts with a feasible flow (all zeros) and improves it (by augmentations)
  - essentially optimal if max capacity into t is O(1)
- Many, many other algorithms...

Review of last four lectures

> Modeling with matching, paths, & cuts
  - matching: allow multiple matches, restrict to groups
  - paths: node capacities, lower bounds, etc.
  - cuts: translate min to max, restrict allowed subsets using infinite capacities

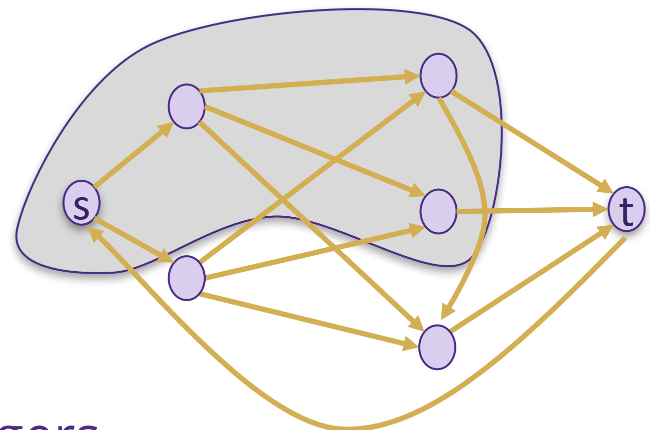
> Many of those graphs have O(1) capacities, so F-F is fast

Review of last four lectures

- Defined the min cost flow problem
  - find the feasible flow with given value and minimum cost
- Described the cycle-cancelling algorithm
  - find a feasible flow with Ford-Fulkerson (or other max flow) algorithm
  - repeatedly push flow along negative cost cycles in G(f) until none exist
    - flow has min cost iff there is no negative cost cycle in G(f)
- Described the successive shortest path algorithm
  - find min cost flow of value 0 (assumed all zeros)
  - repeatedly push 1 unit of flow on *min cost* path in G(f)
    - increases value but stays min cost (neg cycle would imply not shortest)

Review of last four lectures

> **Theorem**: value of max-flow = capacity of min-cut
  - any flow value ≤ any cut capacity
    > flow has to leave via those edges
  - F-F gives us a flow that matches cut value
    > flow value = flow leaving cut – flow entering cut
    > cut edges are saturated
    > backward edges have 0 flow



> **Theorem**: if capacities & costs are all integers, there is an *integral* min-cost flow.
  - likewise for maximum flow

Today

- More modeling with min-cost flow...
- Have already seen two:
  - the assignment problem
    - i.e., minimum cost perfect matching in a bipartite graph
  - the transportation problem
    - i.e., minimum cost flow with demands in a bipartite graph
    - warehouses have negative excess
    - stores have positive excess

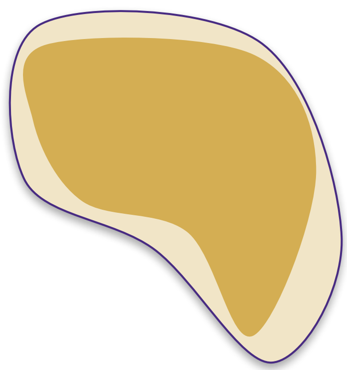
Outline for Today

> Image Recognition
> Island Hopping Airplane
> Scheduling with Deferrals
> LPs with consecutive 1s

Reconstruction from X-Ray Projection

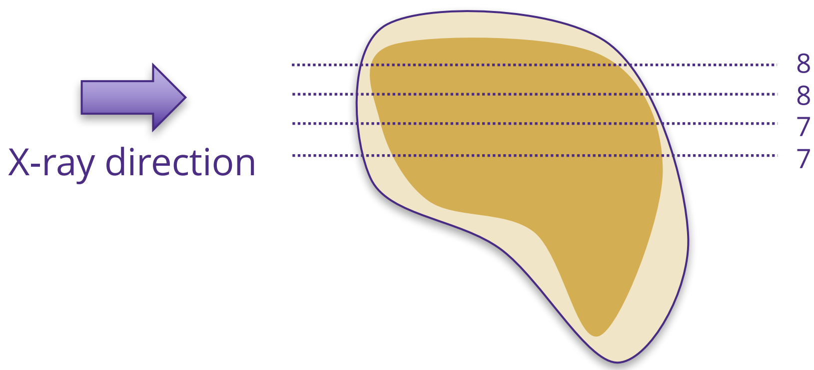
> **Problem**: Determine where blood is flowing through the heart from horizontal and vertical X-ray projections.

- used to detect heart disease
- injected dye that is visible in X-rays shows where there is flow (dark region)



Reconstruction from X-Ray Projection

- **Problem**: Determine where blood is flowing through the heart from horizontal and vertical X-ray projections.
  - injected dye that is visible in X-rays shows where there is flow (dark region)
  - BUT you only get projections that show sum of flow intensities



Reconstruction from X-Ray Projection

> **Re-formulation**: Fill in a table with 0/1 values so that its row and column sums match given values.

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**1**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	

Reconstruction from X-Ray Projection

- **Re-formulation**: Fill in a table with 0/1 values so that its row and column sums match given values.

- In general, there will be many possible tables that achieve given row and column sums…
  - we need to change the problem so that it picks the most likely one
  - assign a measure $L_{i,j}$ of the likelihood that flow appears in position (i,j)
    - this can be based on data from same person or people in general
    - we will use the log of the odds ratio: log(p/(1-p))
  - goal is to maximize sum of $L_{i,j}$ in positions where 1s appear
    - equivalently, minimize the sum of $-L_{i,j}$'s

Reconstruction from X-Ray Projection

> **Re-formulation**: Given likelihoods, $L_{i,j}$ for each position, and row and column sums, fill in a table with 0/1 values so the sums match and the sum of likelihoods where 1s appear is maximized.

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**1**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	

Reconstruction from X-Ray Projection

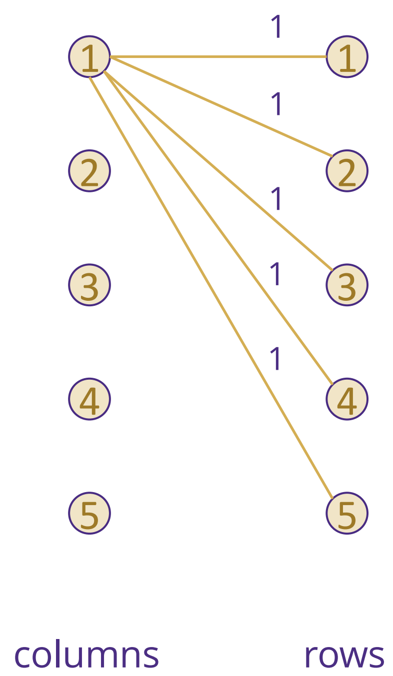
- > **Q**: How do we model this?
- > **A**: Looks broadly like a matching problem
  - **Q**: what is matched with what?
  - **A**: each 1 is a match of a row & column

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	

Reconstruction from X-Ray Projection

> Start with the matching network…

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	

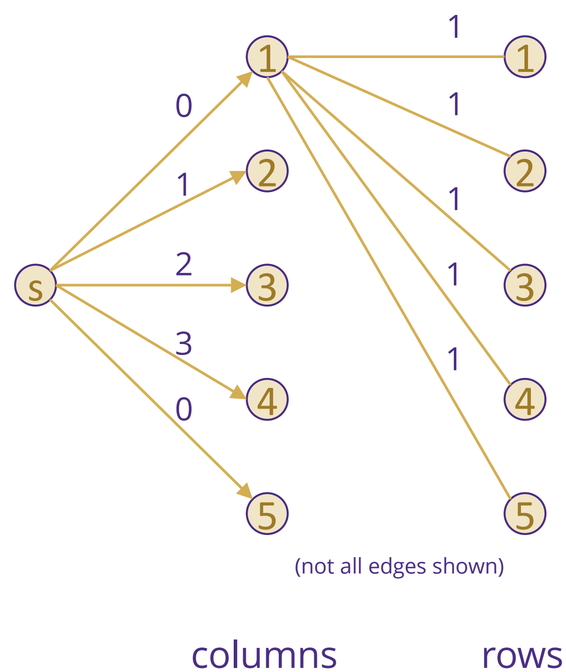


connect every row & column with capacity 1 and cost $-L_{i,j}$

Reconstruction from X-Ray Projection

connect source to each column
with capacity = column sum

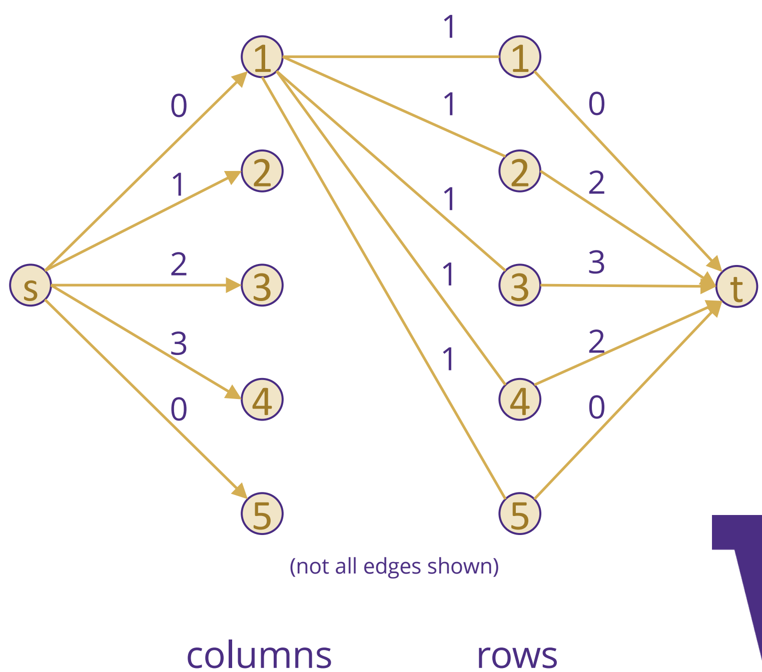
0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	



Reconstruction from X-Ray Projection

connect each row to the sink
with capacity = row sum

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	

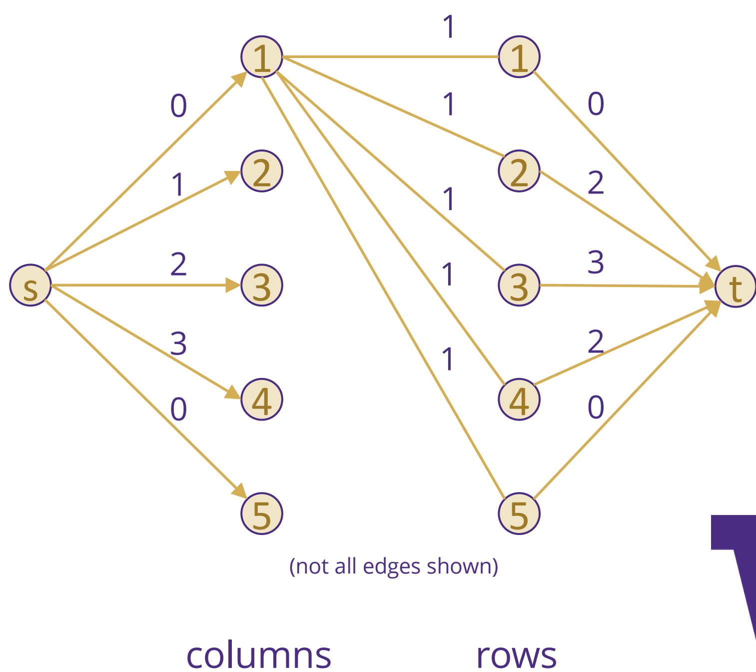


Reconstruction from X-Ray Projection

integral flows in 1-to-1
correspondence with
tables that match sums

saturated (col, row) edges
indicate the 1s in the table

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	



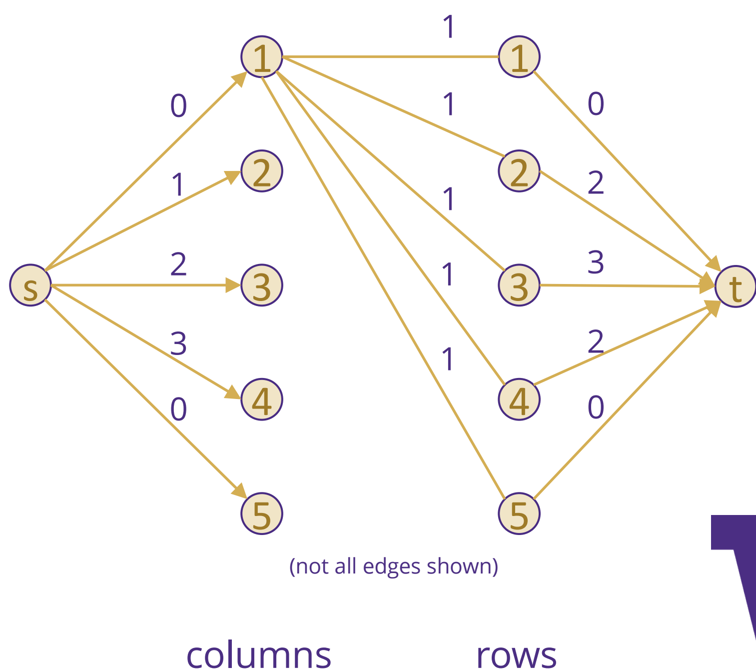
Reconstruction from X-Ray Projection

cost of flow is sum of
flow value * cost over edges

only non-zero costs are
on (col, row) edges

only non-zero flows are
on saturated (col, row) edges

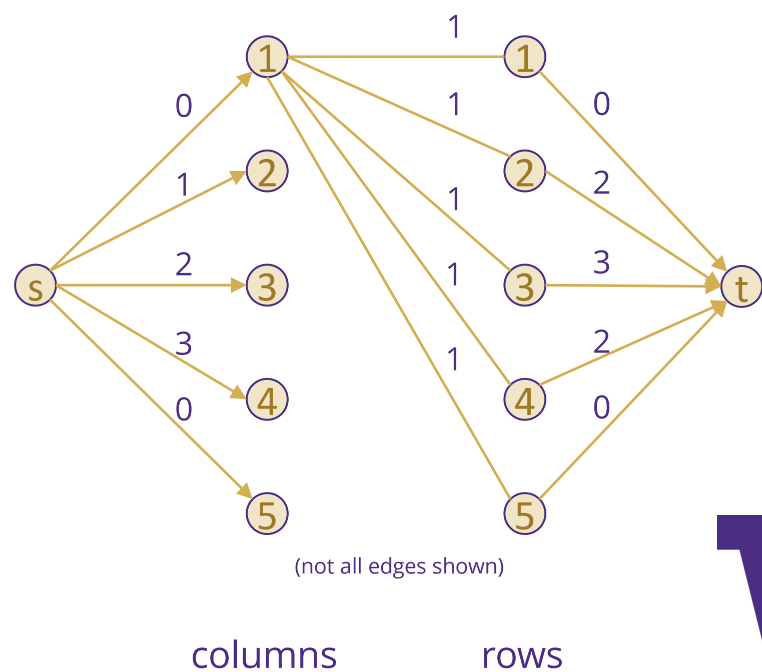
0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	



Reconstruction from X-Ray Projection

cost of flow is sum of
$-L_{i,j}$'s over where the 1s appear

0	0	0	0	0	**0**
0	0	1	1	0	**2**
0	1	1	1	0	**3**
0	0	0	1	0	**2**
0	0	0	0	0	**0**
0	**1**	**2**	**3**	**0**	



Outline for Today

- > Image Recognition
- > Island Hopping Airplane
- > Scheduling with Deferrals
- > LPs with consecutive 1s

Scheduling an Island Hopping Airplane

- You pilot a seaplane for Kenmore Air making a fixed schedule of hops between the San Juan islands each day....

- **Problem**: Given the available passengers and fare per passenger for the trip between each pair of islands, determine which ones you should service in order to maximize revenue.

Scheduling an Island Hopping Airplane

> E.g., your schedule is Anacortes > Lopez > Friday Harbor > Orcas > Rosario

	Lopez	Friday	Orcas	Rosario
Anacortes	1, $10	2, $12		1, $80
Lopez		1, $5	1, $8	2, $10
Friday			2, $5	1, $5
Orcas				3, $10
Rosario				

Scheduling an Island Hopping Airplane

> Problem is easy so far…
> **Q**: How do you solve it?
> **A**: Fly everyone!

	Lopez	Friday	Orcas	Rosario
Anacortes	1, \$10	2, \$12		1, \$80
Lopez		1, \$5	1, \$8	2, \$10
Friday			2, \$5	1, \$5
Orcas				3, \$10
Rosario				

Scheduling an Island Hopping Airplane

- You pilot a seaplane for Kenmore Air making a fixed schedule of hops between the San Juan islands each day....

- **Problem**: Given the available passengers and fare per passenger for the trip between each pair of islands and the <u>airplane capacity</u>, determine which ones you should service in order to maximize revenue.

Scheduling an Island Hopping Airplane

> If plane capacity is 1, just take the passenger from Anacortes to Rosario…

	Lopez	Friday	Orcas	Rosario
Anacortes	1, \$10	2, \$12		1, **\$80**
Lopez		1, \$5	1, \$8	2, \$10
Friday			2, \$5	1, \$5
Orcas				3, \$10
Rosario				

Scheduling an Island Hopping Airplane

> If that fare is only $10 instead, then optimal solution is $30. Can get that by all 1-hops.

	Lopez	Friday	Orcas	Rosario
Anacortes	1, $10	2, $12		1, $10
Lopez		1, $5	1, $8	2, $10
Friday			2, $5	1, $5
Orcas				3, $10
Rosario				

Scheduling an Island Hopping Airplane

- **Exercise**: solve capacity 1 case with dynamic programming
- **Exercise**: expand to any capacity with dynamic programming (with running time proportional to capacity)

	Lopez	Friday	Orcas	Rosario
Anacortes	1, $10	2, $12		1, $10
Lopez		1, $5	1, $8	2, $10
Friday			2, $5	1, $5
Orcas				3, $10
Rosario				

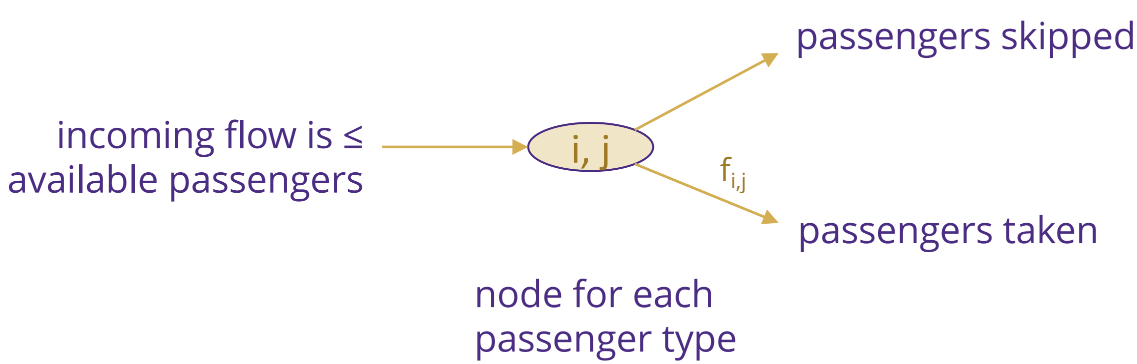
Scheduling an Island Hopping Airplane

> Will model this directly with flows (no matchings)

> Want the flow to tells us whether / how many passengers to pick up of each type.

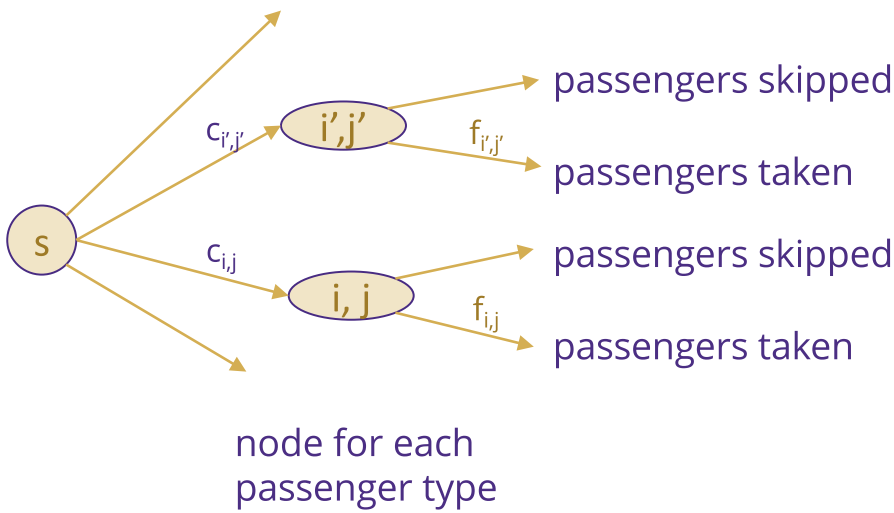
Scheduling an Island Hopping Airplane

- Flow tells us how many passengers to pick up of each type.
- Start by using flow to indicate this decision…
  - cost only appears on when passengers are taken



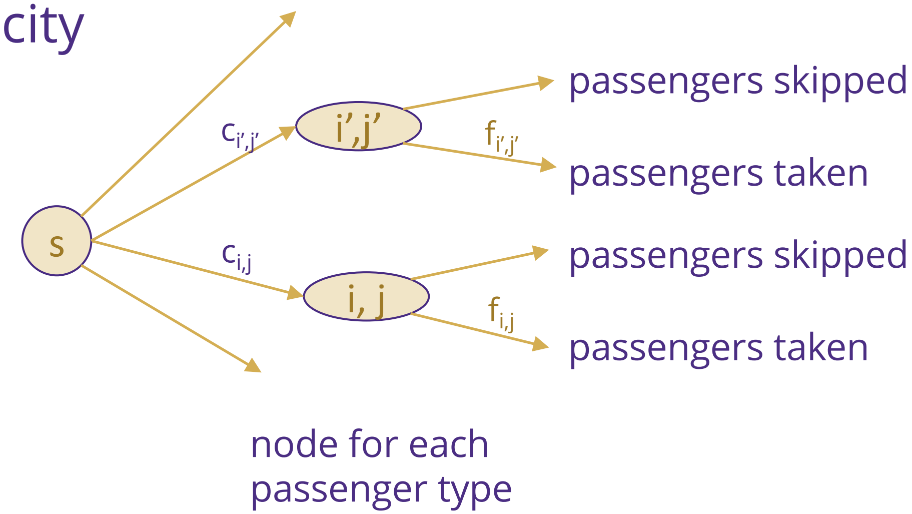
Scheduling an Island Hopping Airplane

> Source supplies flow to each decision node limited by the total number of passengers available…



Scheduling an Island Hopping Airplane

- We could just connect all flows to the sink.
- **Q**: What's missing?
- **A**: airplane capacity



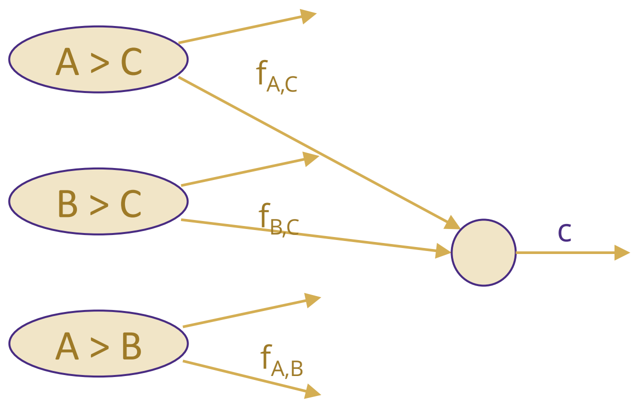
Scheduling an Island Hopping Airplane

> Flows for all passengers that travel together must go through a single edge so that we can limit it

> Not a simple matter of grouping…
  - flow for A > C passengers needs to go through an edge with A > B passengers <u>and also</u> an edge with B > C passengers
  - we can't copy the flow into two places…

Scheduling an Island Hopping Airplane

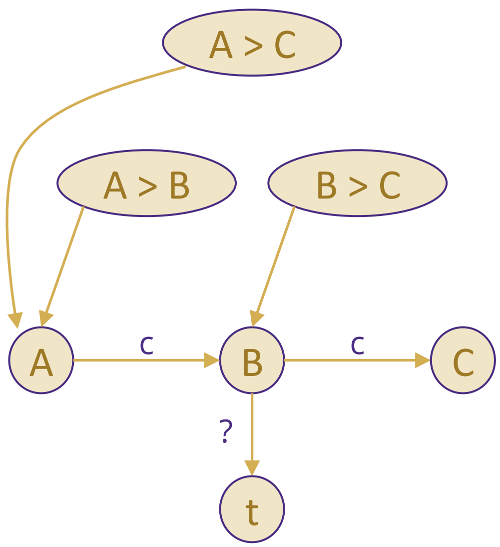
- Not a simple matter of grouping...
  - flow for A > C passengers needs to go through an edge with A > B passengers <u>and also</u> an edge with B > C passengers
  - we can't copy the flow into two places...



this limits number on B > C hop
but not the number on A > B hop

Scheduling an Island Hopping Airplane

> Take advantage of the structure: hops occur in order

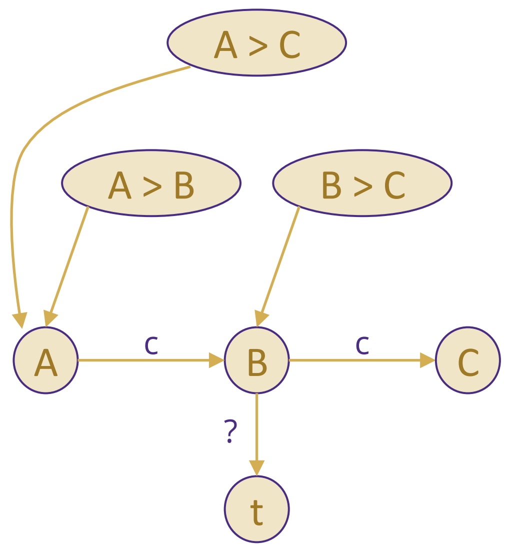


want the A > B passengers to go into t

Problem: we don't know number of them, which we need to set the capacity...

Scheduling an Island Hopping Airplane

> Take advantage of the structure: hops occur in order



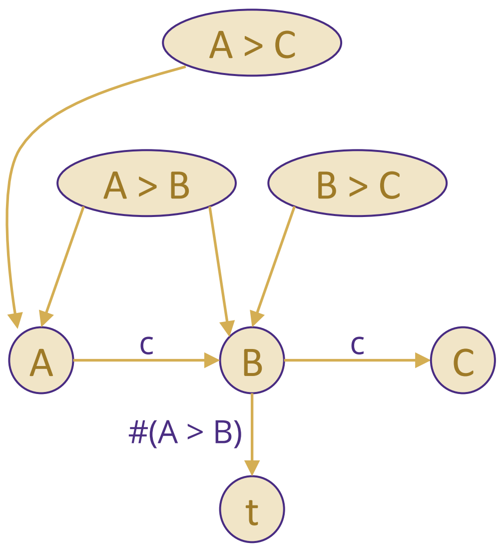
want the A > B passengers to go into t

Problem: we don't know number of them, which we need to set the capacity…

Solution: send skipped A > B passengers to node B also, so every passenger is there

Scheduling an Island Hopping Airplane

> Take advantage of the structure: hops occur in order



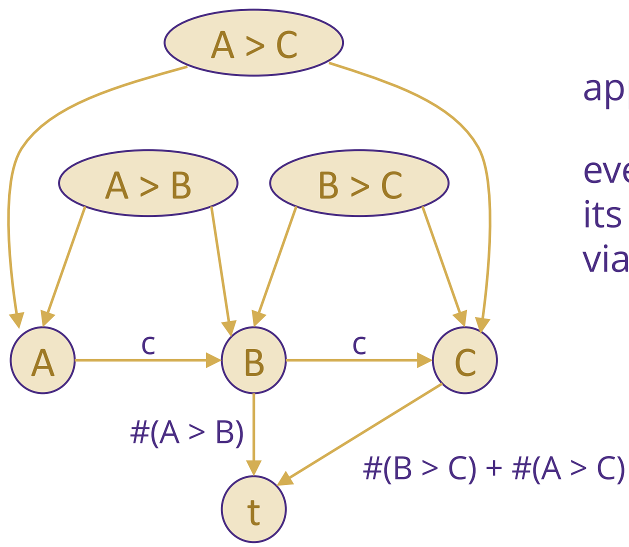
want the A > B passengers to go into t

Problem: we don't know number of them, which we need to set the capacity...

Solution: send skipped A > B passengers to node B also, so every passenger is there

Scheduling an Island Hopping Airplane

> Take advantage of the structure: hops occur in order

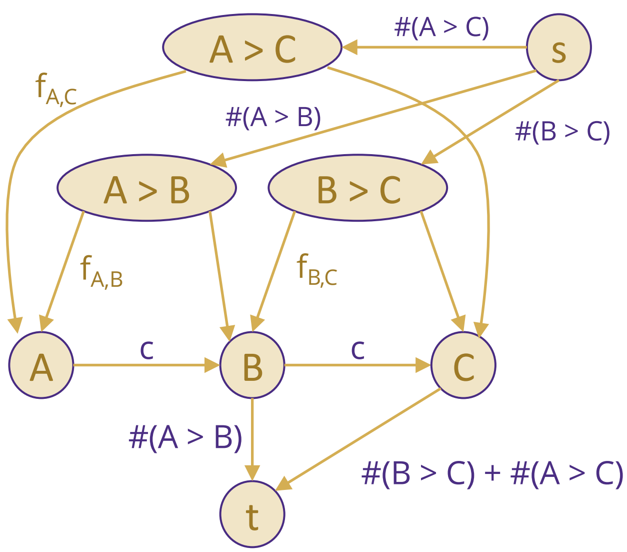


apply the same thing at other nodes

every passenger ends up at the node for its destination, either directly (skipped) or via the hopping path with costs & capacity

Scheduling an Island Hopping Airplane

> Final solution looks like this when there are only 3 hops (A, B, C).



Outline for Today

> Image Recognition
> Island Hopping Airplane
> Scheduling with Deferrals
> LPs with consecutive 1s

Scheduling with Deferral Costs

- **Problem**: Given n programs, each of which takes T seconds to run, and m machines along with
  - times s_j at which program j becomes available to run
  - times e_j by which program j must be run
  - cost functions $c_j(t)$ for completing program j at time t

  Find the allowed schedule for the programs with min total cost.

- We've seen similar problems before, but without costs

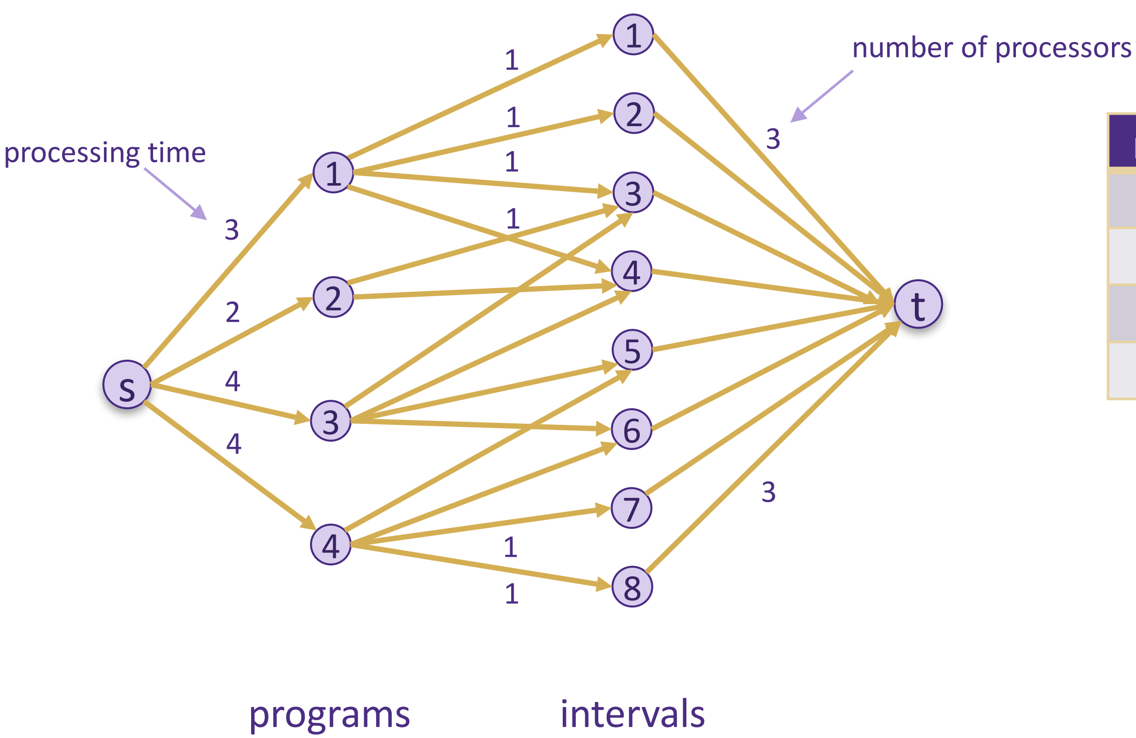
Scheduling with Deferral Costs

- **Problem**: Given n programs, each of which takes T seconds to run, and m machines along with
  - times s_j at which program j becomes available to run
  - times e_j by which program j must be run
  - cost functions $c_j(t)$ for completing program j at time t

  Find the allowed schedule for the programs with min total cost.

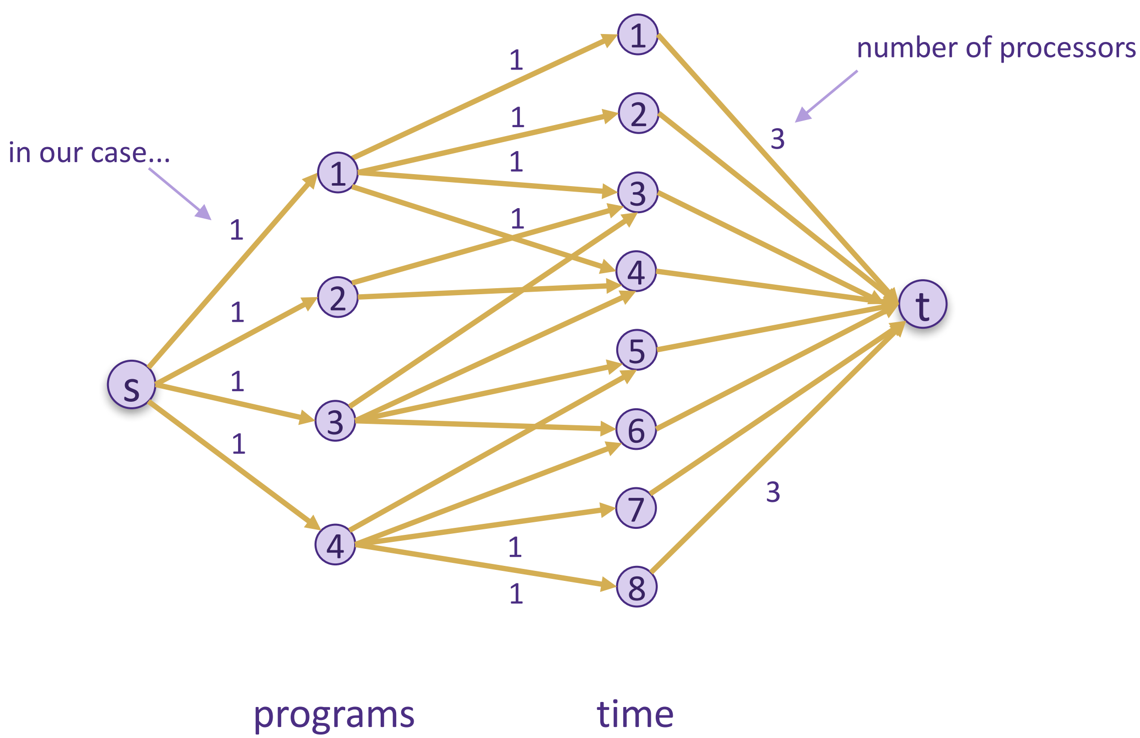
- Key part of this example is <u>arbitrary</u> cost functions
  - we do not need them to be, e.g., linear functions

Recall: Processor scheduling



program	start	end	req time
1	1	5	3
2	3	5	2
3	3	7	4
4	5	9	4

Scheduling with Deferral Costs



program	start	end
1	1	5
2	3	5
3	3	7
4	5	9

Scheduling with Deferral Costs

- Simplified earlier construction by setting all intervals to T sec
- Now, just add cost on (program j, time t) edge with cost $c_j(t)$
  - the exact shape of the function does not matter to us
- (The assumption of equal time to run each program matters…
  - if we don't assume that, then the problem gets harder
  - would need to model how programs are run within an interval
    - doesn't fit well with network flows)

Outline for Today

> Image Recognition
> Island Hopping Airplane
> Scheduling with Deferrals
> LPs with consecutive 1s

Linear Programs (out of scope)

- A linear programming problem asks you to minimize a linear function subject to linear equality and inequality constraints

- Example (from "Network Flows"):

$$
\begin{array}{ll}
\text{minimize} & x_1 + 2\,x_2 - x_3 + x_4 + 3\,x_5 \\
\text{subj. to} & x_2 + x_4 + x_5 \geq 5 \\
 & x_1 + x_2 + x_5 \geq 12 \\
 & x_1 + x_2 + x_3 \geq 10 \\
 & x_1 + x_2 + x_3 \geq 6 \\
\text{and} & x_1, x_2, x_3, x_4, x_5 \geq 0
\end{array}
$$

Linear Programs (out of scope)

> Rewritten in table / matrix form:

minimize $x_1 + 2\,x_2 - x_3 + x_4 + 3\,x_5$
subj. to $A\,x \geq b$
and $x \geq 0$

where $A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$, and $b = \begin{bmatrix} 5 \\ 12 \\ 10 \\ 16 \end{bmatrix}$

Linear Programs (out of scope)

- All constraint information is in this table:
  - (line separates A and **B**)

subj. to $x_2 + x_4 + x_5 \geq 5$

$x_1 + x_2 + x_5 \geq 12$

$x_1 + x_2 + x_3 \geq 10$

$x_1 + x_2 + x_3 \geq 6$

and $x_1, x_2, x_3, x_4, x_5 \geq 0$

0	1	0	1	1	5
1	1	0	0	1	12
1	1	1	0	0	10
1	1	1	0	0	6

Linear Programs (out of scope)

- All network flow problems we have seen are special types of linear programming problems
  - shortest paths
  - max flow
  - min cost flow

- Nearly all of the algorithms we have seen are special cases of techniques for linear programs
  - specifically the “primal dual algorithm”
  - (historically, the LP techniques are generalizations of those)

Linear Programs (out of scope)

- All network flow problems we have seen are special types of linear programming problems
- Nearly all of the algorithms we have seen are special cases of techniques for linear programs
  - primal dual algorithm is widely applicable and worth learning
- Max-flow Min-cut theorem is a special case of LP duality
  - (historically, the LP result is a generalization of this)

Linear Programs (out of scope)

> Why study not just study LPs?

1. Algorithms for network flows are **much** faster than for LPs
   - true even for the "network simplex method", which is (in principle) a special case of the simplex method for LPs
2. It's hard to see when LPs have **integer** solutions
   - in principle, just check if the matrix A is "totally unimodular"
   - in practice, hard to see if this is true for your problem
   - BUT we know network flow problems have integer solutions

Linear Programs (out of scope)

- If you know that you need integer solutions (e.g., matching), often best to try to model the problem directly as a network flow
- If you can, you get integer solutions (much more quickly)

Linear Programs (out of scope)

- An easy case: LPs with consecutive 1s in the A matrix

0	1	0	1	1	5
1	1	0	0	1	12
1	1	1	0	0	10
1	1	1	0	0	6

- Will see how to convert this to min cost flow…

Linear Programs (out of scope)

> Step 1: turn $Ax \geq b$ into $A x' = b$ by introducing new variables

0	1	0	1	1	1	0	0	0	5
1	1	0	0	1	0	1	0	0	12
1	1	1	0	0	0	0	1	0	10
1	1	1	0	0	0	0	0	1	6

> Changes, e.g., $x_2 + x_4 + x_5 \geq 5$ into
$x_2 + x_4 + x_5 + y_1 = 5$

– equivalent since we also have $y1 \geq 0$

Linear Programs (out of scope)

- Step 2: add a row of zeros at the bottom

0	1	0	1	1	1	0	0	0	5
1	1	0	0	1	0	1	0	0	12
1	1	1	0	0	0	0	1	0	10
1	1	1	0	0	0	0	0	1	6
0	0	0	0	0	0	0	0	0	0

Linear Programs (out of scope)

- Step 3: each row from the next one
  - (this is just a change of variables)

0	1	0	1	1	1	0	0	0	5
1	0	0	-1	0	-1	1	0	0	12
0	0	1	0	-1	0	-1	1	0	10
0	0	0	0	0	0	0	-1	1	6
-1	-1	-1	0	0	0	0	0	-1	0

Linear Programs (out of scope)

> Result: LP with one +1 and one -1 in every column

0	1	0	1	1	1	0	0	0	5
1	0	0	-1	0	-1	1	0	0	12
0	0	1	0	-1	0	-1	1	0	10
0	0	0	0	0	0	0	-1	1	6
-1	-1	-1	0	0	0	0	0	-1	0

Linear Programs (out of scope)

- These are node balance constraints of a min cost flow problem:
  - each column represents an edge
    - +1 where the edge is incoming
    - -1 where the edge is outgoing
  - each row requires the excess at the node equals a given value (demand)

0	1	0	1	1	1	0	0	0	5
1	0	0	-1	0	-1	1	0	0	12
0	0	1	0	-1	0	-1	1	0	10
0	0	0	0	0	0	0	-1	1	6
-1	-1	-1	0	0	0	0	0	-1	0

Linear Programs (out of scope)

- LPs with consecutive 1s in the A matrix are node balance constraints of a min cost flow problem (once transformed)

- Intuition without LPs: problems with an *ordered* sequence of constraints, where each decision affects a *consecutive* range of the constraints, can be modelled as a network flow problem
  - see airline hopping example
  - see HW1 again!