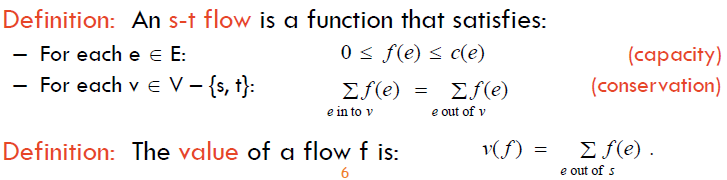
**Flow Networks Summary**

G = (V , E)->A directed graph with no parallel edges. (Simple Directed Graph , with one source and sink, non-negatively capacity for each edges)

**S-T flow**->each edges should have f(e) <= c(e) , which means the flow value shouldn’t exceed its capacity.

**Conservation Property**->the total value of incoming flow = the total value of outgoing flow. 进出相等

**Capacity Property**: 0<=f(e)<=c(e), capacity 不能越界



**The value of the flow** is the total value outgoing from source S.

**S-T cut**: It is a partition (A , B) of the graph, with s in A and t in B.

**The capacity of the cut**: The value of the cut (A , B) is the sum of the capacities of all edges which outgoing from part A. (Ignoring all incoming edges).

**Flow Value Lemma**: Let f be any flow, and let (A , B) be any s-t cut. Then the net flow sent across the cut is equal to the amount leaving s. (now should consider both incoming and outgoing edges).

Value(flow) = flowout(A) – flowin(A)

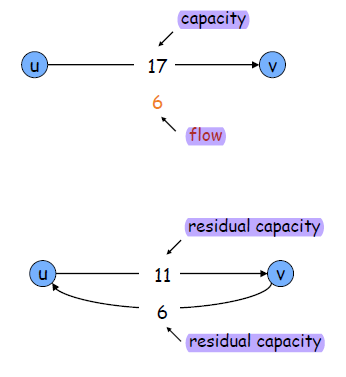
**Weak Duality**: Let f be any flow, and let (A, B) be any s-t cut. Then the value of the flow is at most the capacity of the cut, which means v(f) <= cap(A , B)

**Weak Duality Corollary**: Let f be any flow, and let (A, B) be any cut. If v(f) = cap(A , B) then f is a max flow and (A , B) is a min cut.

**Fuld-Fulkerson Algorithm** (**Used to find the max flow**)

**Residual Graph**: The residual capacity of an edge in the residual graph tells us how much flow we can send, given the current flow.

原始状态

**undo flow sent**~

在流过了一个flow之后，原有的那个edge的capacity对应减少，取而代之的是增加一个新的反向的edge。

**Augmenting Paths**->Path from source S to sink T in the residual graph.

**Bottleneck (P, f)**: minimum residual capacity of any edge on simple s-t path P with respect to the current flow f.

**Ford-Fulkerson satisfies the Integrality Theorem**.

**Integrality Theorem**: If all capacities are integers, then there exists a max flow f for which every flow value f(e) is an integer.

**Ford-Fulkerson-Assumption**: All initial capacities are integer.

**Lemma**: At every intermediate stage of the Ford-Fulkerson Algorithm, the flow values and the residual graph capacities in Gf are integers. Can be proved by induction.

**Augmenting Path Theorem**: Flow f is a max flow if and only if there are no augmenting paths in the residual graph.

**Max-Flow Min-Cut Theorem**: The value of the max flow is equal to the value of the min cut.

**Ford-Fulkerson**: The flow value **strictly increases in an augmentation**. Increase at least 1 each time. (Each augmentation increases flow value by at least 1.)

**Ford-Fulkerson Running Time**: Corollary: Ford-Fulkerson runs in O(Cm) time, if all capacities are integer, the C is the upper bound on the maximum flow, and the m is the number of edges in the graph.

**Edmonds-Karp Algorithm**: Pick the augmenting path with largest capacity (maximum bottleneck path).

Flow increased by at least one at every iteration. So the while loop is repeated Fmax times at most, where Fmax is the maximum flow value.

**Integrality Theorem**: If all capacities are integers then every flow value f(e) and every residual capacity remains an integer throughout the algorithm.

**Matching**

**(Bipartite) Matching**: M, which is a subset of edge E, is a matching if each node appears in at most one edge in M.

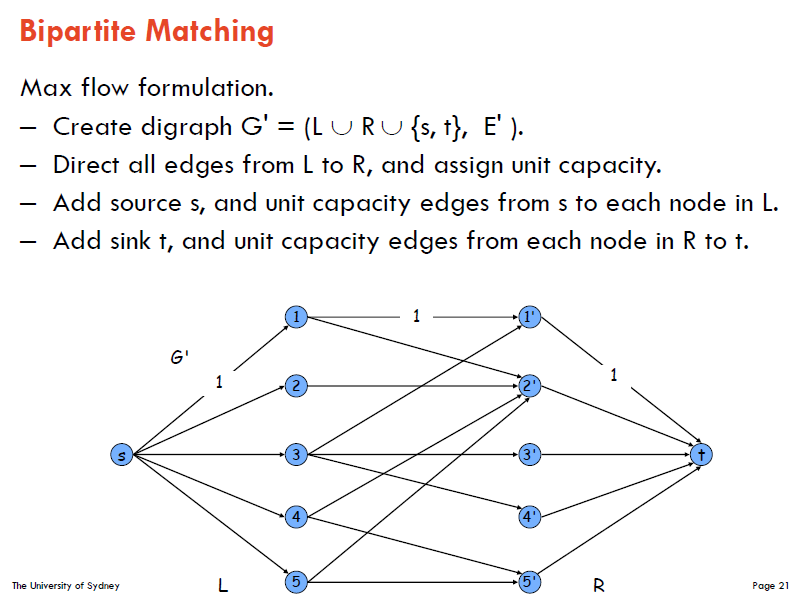
**Bipartite Matching:** In the undirected bipartite graph G, M, which is a subset of edge E, is a matching if each node appears in **at most one** edge in M.

**Bipartite Matching Problem**: Given an undirected graph G = (V, E), find a max cardinality matching.

**Max Matching**: The max cardinality matching.

**Bipartite Matching Running time** is O(mC).

**Bipartite Matching Formulation**:



**Theorem**: the max cardinality matching in G is the value of max flow in G’.

**General Proofing Idea for Max Flow Problem**: Prove it in two directions. For example, if the max flow value is k, then there exists a best Christmas tree selling schedule which sells k trees, and if the best selling schedule sells k trees, then there exists a max flow with value k.

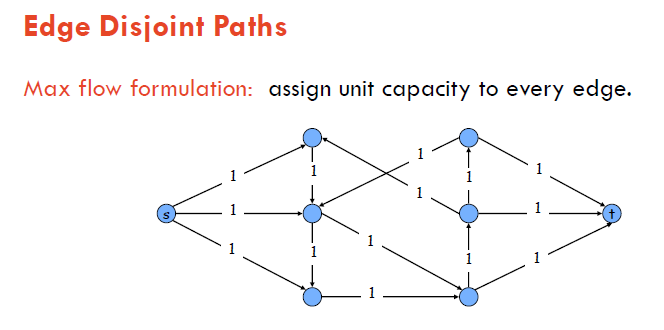
**Perfect Matching**: A matching is perfect if each node appears in **exactly one** edge in M.

**Perfect Matching Observation**: Let S be a subset of nodes, and let N(s) be a set of nodes adjacent to nodes in S. If a bipartite graph G has a perfect matching, then |N(s)| >= |S| for all subsets S which is subset of LHS.

**Marriage Theorem**: Let G be a bipartite graph with |L| = |R|. The G has a perfect matching if and only if |N(S)|>=|S| for all subsets S which is the subset of L.

**Edge Disjoint**: Two paths are edge-disjoint if they have no edge in common.

**Edge-Disjoint Paths problem**: Given a digraph G = (V , E) and two nodes s and t, find the max number of edge-disjoint s-t paths.

**Theorem**: Max number edge-disjoint s-t paths equals max flow value.

Theorem: Max number edge-disjoint S-T paths equals to max flow value.

**Network Connectivity**: Given a digraph G = (V , E) and two nodes s and t, find the min number of edges whose removal disconnects t from s.

A set of edges F (which is a subset of edge set E) disconnects t from s if all s-t paths uses at least one edge in F.

**Theorem**: The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.