Let be a Graph

**Cuts**

A is a partition of into and such that (source) belongs to and (sink) belongs to.

The capacity of the cut is

A minimum cut is a cut whose capacity is minimum.

**Algorithm to find minimum cut**

Find maximum flow and define = {all vertices such that there exists a path from them to s in the final residual network} and . Then will be a minimum cut.

**Coverings, Matching, Independent Set**

Source: <https://www.epfl.ch/labs/dcg/wp-content/uploads/2018/10/GT-4-Covers.pdf>

**Preliminaries**

Bipartiteness:

A graph is bipartite if its vertices can be divided into two disjoints sets such that there is no edge between vertices of the same set.

**Necessary and sufficient condition:**

A graph is bipartite iff it doesn’t have and odd cycle.

**Definitions**

* **Matching :** Is a set such that the edges in M are pairwise disjoint
* **Vertex Cover:** Is a set such that every edge of is incident to a vertex of .
* **Edge Cover:** Is a set such that every vertex of is incident to an edge in (this concept is only defined in graph without isolated vertex)
* **Independent set:** Is a set such that no two vertices in are adjacent.

**Inequalities**

For any arbitrary Graph:

For any arbitrary Graph without isolated vertices:

**Gallai Theorem:**

For any arbitrary Graph:

For any arbitrary Graph without isolated vertices:

**Konig Theorem:**

Source: https://www.epfl.ch/labs/dcg/wp-content/uploads/2018/10/GT-3-Matchings.pdf

If the graph is bipartite,

If, additionally, doesn’t have isolated vertices,

**Hall’s Theorem:**

* **Definition:** A matching “covers” if every vertex in is an endpoint of an edge of the matching.
* **Definition:**  is the set of neighbours of each node of

**Theorem:** Let be a bipartite graph with bipartition . Then has a matching that covers if and only if for all we have .

**Algorithm for finding each of them in Bipartite Graph:**

Let say that our bipartite graph has the partition

* **Maximum matching**: Run the max flow algorithm on. All the edges between that have flow are edges of a maximum matching
* **Minimum edge cover**: Let denote the maximum matching size by |M|. Take the |M| edges of the maximum matching. For the other |V| - 2 |M| unmatched vertices, take one of its edges (the other endpoint must be matched). This set of edges is a minimum edge covering.
* **Minimum vertex cover**: Find a minimum cut . Take all the edges of the cut (those that goes from to ). All the vertices that belong to those edges (except from the source and the sink) form a minimum vertex cover.

(Source: http://theory.stanford.edu/~trevisan/cs261/lecture14.pdf)

* **Maximum Independent set:** Take all the vertices that are not in the minimum vertex cover. These vertices form a maximum independent set.

**Partially Ordered Sets**

**Definitions:**

* **Partial Order:** A (strict) partial order over a set is a binary relation, , over that is:
  1. irreflexive: for all and , implies
  2. transitive: for all , and implies.

Also, if or , then we say that these elements are comparable; otherwise they are incomparable.

We can represent a poset (partially ordered set) as a DAG.

* **Chain:** Is a subset of such that every pair of elements is comparable
* **Antichain:** Is a set of V such that every pair of elements is incomparable.

Note: A one element is both a chain and an antichain

* **Chain partition:** Is a partition of (group of pairwise disjoint non-empty subsets of ) such that each subset is a chain.
* **Antichain partition:** Is a partition of such that each subset is an antichain.
* **Height:** The size of the maximum chain
* **Width:** The size of the maximum antichain

**Inequations:**

**Mirsky’s Theorem:**

**Statement**: In a poset, it holds that

That means that a poset of **height**  can be partitioned in chains

**Construction of the minimum antichain partition:** Recursively remove the minimal (maximal) elements of the poset. Note that all minimal (maximal) elements at each iteration, form an antichain.

Minimal (maximal) elements in a DAG are the ones with outdegree (indegree) equals 0.

**Construction of maximum chain**: We can start with the nodes with indegree 0 and trying to pick the best choice of the chain using dp (or topological sorting).

**Dilworth Theorem:**

Inductive proof : <https://pwp.gatech.edu/math3012openresources/lecture-videos/lecture-14/>

Constructive proof : <https://web.stanford.edu/class/cs361b/files/cs261-Jan2014-notes.pdf>

**Statement**: In a poset, it holds that

That means that a poset of **width**  can be partitioned in chains.

Also

**Construction:**

Let’s denote the DAG of the poset as

Let’s construct the bipartite graph where

, that means we create 2 nodes in for each node in G.

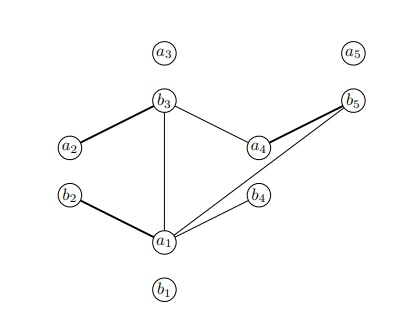
that means that we create and edge in the for each pair of vertex in G such that is an ancestor of .

If we denote . Then it holds that

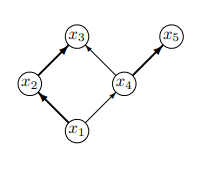
* For any matching in , we can project each edge of the matching to an edge in and it forms a chain partition . Each chain of the partition if forms by the maximal union of edges that are adjacent in the projection of .

Moreover:

See the example below:



Bipartite Graph with a matching in bold



Original Graph G with the chain partition in bold

* For any vertex cover in , there exists an antichain in such that . The antichain is form in the following way: Project in G an denote this as .

Then

* If we denote as the maximum matching, as the minimum vertex cover , as the maximum antichain , as the minimum chain partition.

Then

**Construction of minimum chain partition:**

First build the maximum matching in with max flow algorithm. Then map each edge of this matching with an edge in . If you consider only the mapped edges in , each connected component form a chain, and the union of all of them is the minimum chain partition.

**Construction of maximum antichain:**

First build the minimum vertex cover in using the nodes of the min cut. Then map each node of this vertex cover with a node in (some may be repeated) and call this set . Then the antichain is form by the set of vertex that is not in .