Partially Ordered Sets, Mirsky's Theorem and Dilwhorth's Theorem

md.mottakin.chowdhury

June 2018

1 Partially Ordered Sets

A set of elements that have relation on an operator, say $\leq$ for example. So if there are any two elements x and y, their relation can be $x \leq y$. Let's assume we have such a poset, P. It has:

Reflexivity—if x is an element in the set, then $x \leq x$.

Antisymmetry—if $x \leq y$ and $y \leq x$, then $x = y$.

Transitivity—if $x \leq y$, $y \leq z$ then $x \leq z$. The term itself talks about partiality, so not all elements are related on the particular operator.

The term itself talks about partiality, so not all elements are related on the particular operator.

2 Chain and Anti-chain

Chain—so we have elements in our poset P. We take a subset C from P. If these elements can be compared on some particular sequence one after another, then this subset is a *chain*.

Suppose, we have taken a, b, c, d from our poset. If we can place the relational operator in between these elements, like $a \leq b \leq c \leq d$, then it is a chain.

Antichain—similarly, if we take a subset A from P in such a way that no two of them can be compared on the relational operator, we call such a subset an *antichain*.

Posets can be reduced to DAG:

- A chain is a path.
- An antichain is an independent set. (An independent set is a set of nodes such that there is no edge between any two of them.)

3 Problem Type 1

Given a poset, which is reduced to DAG, we need to find the minimum number of partitions of the elements into antichains or independent sets. This minimum number is found by **Mirsky's Theorem**.

According to this theorem, the **minimum number of partitions is equal to the length of the maximum sized chain a.k.a length of the longest path in the corresponding DAG**.

If we need the actual partition, we define a value **maxend[u] = the length of the longest path that ends in vertex u**.

We have already noted that the minimum number of partition is the length of the longest path, let this length is **maxLen**.

To find the actual partition, we iterate from i=1 to maxLen, and find such nodes u that have maxend[u]=i. All such nodes fall in one partition.

This works because it can be proved that if there are two nodes u and v for which **maxend[u]=maxend[v]**, then it is impossible that these two nodes have an edge between them. So they can be in the same independent set.

So we get that **minimum partition into antichains = maximum size of a chain**.

4 Problem Type 2

In this type, we are asked to find the partition into minimum number of chains. This corresponds to taking minimum number of paths from the DAG to cover all the vertices such that no vertex is covered twice (edge disjoint paths).

This problem can be mapped to **Dilworth's Theorem**. According to Dilworth, minimum number partition into chains is equal to the maximum size of the antichain. In terms of DAG, we need to find the size of the *maximum independent set.*

(**Maximum Independent Set**—it is an independent set such that adding one more vertex to this set will destroy its independent set property).

This can be solved using **bipartite matching**. The idea is, we take two copies of all of our vertices and put them into two sets. Then we add an edge between u from one set and v from the other set if and only if there is an edge between u to v in the original graph.

We then find the maximum matching in this graph. According to **Konig's Theorem**, this maximum matching equals to the minimum vertex cover of the graph, and minimum vertex cover is the dual of the maximum independent set. So, **minimum number of partition into independent set equals to the (number of nodes – maximum bipartite matching) in the constructed graph**.

Vertex Cover—a vertex cover (sometimes node cover) of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set

5 Problem Type 3

What if it is asked that the edges in the previous problem do not have to be disjoint?

In such case, we find the closure of the original DAG. Then apply the previous solution.

Closure of a DAG is such a graph where for any three vertices u, v and w, if there is an edge from u to v, and v to w in the original graph, then in the closure graph, we will add an edge from u to w.