Automata Theory HW5

20181593 계인혜

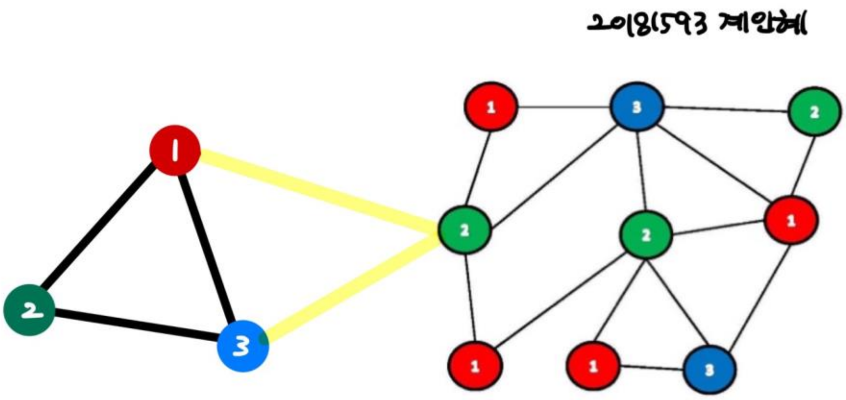
4.2

Before explaining the problem, let's look at the definition of complete graph. A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.

Now let K be a complete graph that has k vertices and G be an original graph, G' be a graph s.t. $G' = G \cup K$. Also let C(v) be vertex v's color.

And connect the vertex v with k-1 vertices, except the c(v). Then, all the vertices in G with a given c(v) have the same color. On the other hand, the vertices with different c(v) have different colors.

The following example is showing when k=3.



Therefore, we can define a new k-colorable graph G' by adding one vertex at a time. We can solve the problem with polynomial number of questions.

4.12

First, there are $2^{10} = 1024$ sublists of the integers. Also, we can infer that the possible sums of at most ten distinct numbers can't be larger than 10*100 = 1000 and sums can't be less than 0(empty list).

So, there are only 1001 possible sums for such a sum.

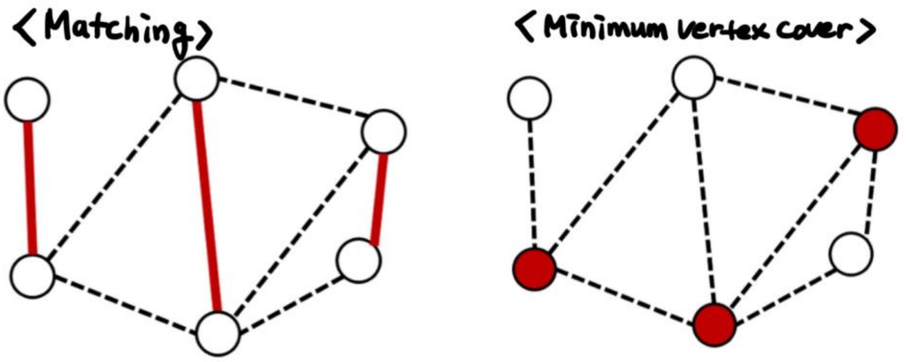
Therefore, there exist two distinct, disjoint sublists $A, B \subset S$ whose elements have the same total by the pigeonhole principle.

1) A and B are disjoint : We are done.
2) A and B have at least one common element : If we remove the common elements, then we can get two distinct, disjoint sublists with the same total.

4.16

Before we start explaining the problem, let’s look at the concept first.

A vertex cover is a set of vertices that includes at least one endpoint of every edge of the graph. And matching is a set of edges without common vertices. The following is an example of these.

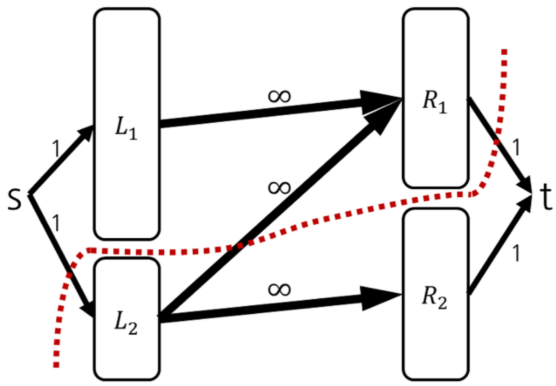


In order to solve the maximum matching problem, it must be converted into a flow network. Since the size of the minimum cut and the maximum matching is the same, in order to solve the problem, you have to find a vertex cover with the same size as the minimum cut.

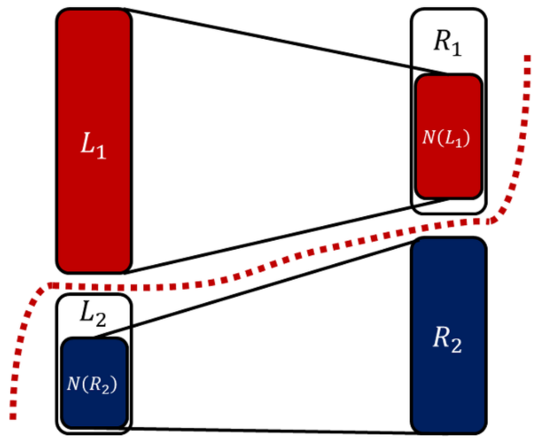
<Kőnig's theorem.>

In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.

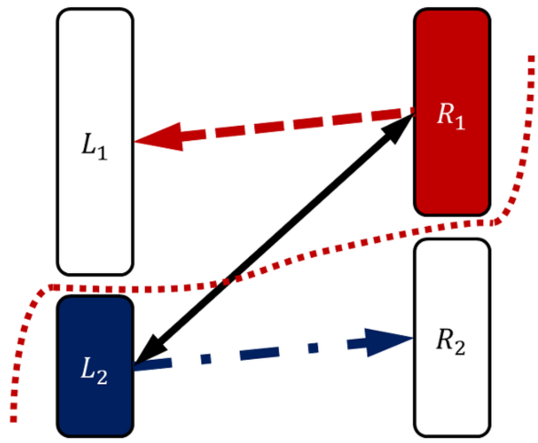
Pf) Let’s consider the following example of the minimum cut.



As shown in the figure below, the vertices adjacent to L1 all belong to R1. Since there is no edge between L1 and R2, we can see that all the vertices adjacent to R2 belong to L2.



The edges connecting L1 and R1 are all processed by R1. In addition, all edges connecting L2 and R2 can be processed by L2. There is no edge connecting L1 and R2, and you can observe that endpoints of the edge connecting L2 and R1 are pulled out. Thus, X becomes vertex cover.



Now we have to consider that the size of minimum cut and vertex cover equal. However, it's natural that the size of minimum cut is $|L_2| \cup |R_1|$.

Hence, VERTEX COVER is in P for bipartite graphs.

4.17

Show that for any constant k, the property VC_k that a graph has a vertex cover of size at most k is minor-closed.

Let a graph G has a vertex cover S of size k. If we remove an edge, then S is still a vertex cover. If we contract an edge (a,b), then at least one of a or b must have been in S. If we include the new, combined vertex in the vertex cover, then it covers all the edges that a or b had before. In either case we get a vertex cover S' of the new graph G', where $|S'| \leq |S|$.

Give an explicit algorithm that takes $O(2^k n)$ time.

For the explicit algorithm, choose any edge (a,b). Branch into two subproblems. In one, include a in S and remove a and its edges from the graph. In the other, do the same with b. In each case, ask whether the remaining graph has a vertex cover of size k-1. Working recursively produces a binary tree of depth k. At each leaf, we ask whether a graph has a vertex cover of size 0, which it does if and only if it has no edges. Each step requires poly(n) bookkeeping to remove a vertex from the graph, so the total running time is $2^k poly(n)$.