

# C1. Task Force

[김석진, 홍원표]

At first, we tried to add qualified soldiers into special force, which is initially empty, using **backtracking**. However, considering **limited time** and the size of input, we realized that another approach is more suitable. For another approach, we needed to **reverse** our paradigm. Our initial paradigm was adding soldiers who are qualified, but it was logically same as **removing soldiers who are not qualified**.

To put it more specific, what the problem requests was not finding the maximum size of special force by adding qualified soldiers, who have at least  $k$  friends in special force, from empty special force, but finding the maximum size of special force consisting of soldiers after **repeating the removal process**, which **removes the soldier who has less than  $k$  friends** from special force and decrement the number of friends of another soldier who is friend of the removed soldier, **until every soldier in special force is qualified**.


In terms of algorithm, removal process can be said that **removing unqualified node and its edges from graph**. In worst case, this algorithm needs to remove every nodes and edges from graph, so time complexity is  $O(f * n^2)$  when  $f$  is the number of friendship (same as the number of edges) and  $n$  is the number of soldiers. We implemented graph using **adjacency matrix**, which is two-dimensional array, so the space complexity is  $O(n^2)$  when  $n$  is the number of soldiers.

Space Complexity :  $O(f * n^2)$

Time Complexity:  $O(n^2)$

[  $f$  : # of friendship (# of edges),  $n$  : # of soldiers (# of nodes) ]

This algorithm can be applied to **grouping-related problem**. For example, in HGU, every team groups several families on every semester. However, someone might have uncomfortable relationship with someone in their team. In this case, team leader can group families with the least uncomfortable relationship using this algorithm.

Soldier	1	2	3	4	5
Relation	1->0	1->0			
		1	1		
			1	1	
				1	1
		1			1
		1		1	
Sum	1->0	4	2	3	2