In a group of people, it is to be expected that some of them may not want to work with each other. Assuming that each person has at most d other people that they don't want to work with, divide the people into d+1 groups so that everyone is in exactly one group and no one is in a group with someone they don't want to work with.

Establish the problem.

- specifications

Task: assign people to *d*+1 groups so that each person is in exactly one group and no one is in a group with someone they don't want to work with

Input: *n* people and, for each, the up to *d* other people they don't want to work with

Output: an assignment of people to groups (i.e. a group number for each person)

- examples

Identify avenues of attack.

- targets
- approach
- paradigms and patterns
    - process input
    - produce output
    - narrow the search space

Define the algorithm.

- main steps

for each person, put them into the first group not containing someone they don't want to work with

- exit condition

when every person has been added to a group

- setup
- wrapup
- special cases
- algorithm

Show termination and correctness.

- termination
    - measure of progress
    - making progress
    - the end is reached
- correctness
    - loop invariant

the first k people have been assigned to groups so that no one is in a group with someone they don't want to work with and there are at most d+1 groups

- establish the loop invariant

k=1 – person 1 is assigned to group 1

- person 1 is the only person in group 1, they cannot be in a group with someone they don't want to work with

- only have 1 group, d >= 0 so we cannot have more than d+1 groups

- maintain the loop invariant

assume the invariant holds for k, show for k+1

assume: the first k people have been assigned to groups so that no one is in a group with someone they don't want to work with and there are at most d+1 groups

show: then first k+1 people have been assigned to groups so that no one is in a group with someone they don't want to work with and there are at most d+1 groups

- alg puts person k+1 into the first group without someone they don't want to work with → no one is in a group with someone they don't want to work with

- if person k+1 goes into an existing group → don't change the number of groups, still <= d+1

- if person k+1 goes into a new group → they have a conflict with someone in each of the existing groups, but there can be at most d of those, so new group makes at most d+1

- final answer: show setup + loop invariant being true when the exit condition is reached + wrapup = correct overall answer

when exit condition is true: k = n, thus the first n people have been assigned to groups so that no one is in a group with someone they don't want to work with and there are at most d+1 groups → "first n" = all

Determine efficiency.

- implementation

build a map person → group – O(1) assigning group, retrieving a person's group

groups → set for each group, allows O(1) contains for that group

for each group, → up to d+1 repetitions

  for each do-not-work-with person for the current person → up to d repetitions

    see if they are in the current group → contains operation, O(1) for hash set

    if so, move on to the next group

  (if we haven't moved on, put the current person in the current group) → O(1) insert into hash set

- time and space

n repetitions of the main loop x time per repetition

time per repetition: up to d+1 groups x up to d don't-work-with x O(1) to check + 1 O(1) add to group → O(d^2) time per repetition

→ O(d^2 n) total

- room for improvement