How to Design Iterative Algorithms

- *Correctness.* Show that the algorithm is correct.
  - *Loop invariant.*
    State a loop invariant.
  - *Establish the loop invariant.*
    Explain why the loop invariant holds at the beginning of the first and second iterations of the loop.
  - *Maintain the loop invariant.*
    Explain why the loop invariant continues to be true after each iteration — assuming that it holds at the beginning of iteration k, explain why it also holds at the beginning of the next iteration $(k+1)$.
  - *Final answer.*
    Explain why the whole algorithm — setup, loop, wrapup — means that the final result is a correct answer to the problem.

Loop Invariants

- a *loop invariant* is a statement about the program state that is true every time the loop condition is checked
  - it is true before the first iteration
  - it remains true after every iteration
  - it helps explain why the algorithm works
    - showing that the loop invariant is true when the loop exits helps us establish that the overall algorithm is correct
- a good loop invariant is
  - *strong enough* to help prove the final correctness of the algorithm
  - *about correct progress* – it addresses the portion of the problem that has already been solved

How to Design Iterative Algorithms

- *Correctness.* Show that the algorithm is correct.
  - *Loop invariant.*
    State a loop invariant.
  - *Establish the loop invariant.*
    Explain why the loop invariant holds at the beginning of the first and second iterations of the loop.
  - *Maintain the loop invariant.*
    Explain why the loop invariant continues to be true after each iteration — assuming that it holds at the beginning of iteration k, explain why it also holds at the beginning of the next iteration $(k+1)$.
  - *Final answer.*
    Explain why the whole algorithm — setup, loop, wrapup — means that the final result is a correct answer to the problem.

pattern	loop invariant
process input	have a correct solution for the first k input elements, or (alternatively) haven't gone wrong yet (solution so far is consistent with a solution for the whole problem)
produce output	have produced the first k elements of the correct output
narrow the search space	either the element is within the current search space / set of solutions or it was never present at all, or (alternatively) not all of the solutions (if there are any) have been eliminated

- common forms for loop invariants
  - the first *k* elements satisfy some property
  - a variable stores the correct result for the elements processed so far

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.

pattern	loop invariant
process input	have a correct solution for the first k input elements, or (alternatively) haven't gone wrong yet (solution so far is consistent with a solution for the whole problem)

- loop invariant
  -  the first *k* people have been assigned to groups
  -  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
  -  there are at most *d*+1 groups
  -  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
  -  have a correct solution for the first *k* input items
  -  haven't gone wrong yet
  -  have produced the first *k* items of the output
  -  the first *k* groups have been assigned people
  -  the first *k* groups have been assigned people without anyone being in a group with someone they don't want to work with

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.

- setup and main steps
  - create d+1 empty groups
  - for each person, put them into a group not containing someone they don't want to work with
- 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
  - show that the loop invariant holds before and after the first iteration (beginning of first and second iterations)

at the beginning of the first iteration, we have d+1 empty groups and no one has been assigned to a group so k=0 → there are at most d+1 groups and no one has been assigned to a group with someone they don't want to work with

the first iteration assigns the first person to a group – since all of the groups are empty, they can be put in any group without being in a group with someone they don't want to work with, and since $d \geq 0$, there is at least one group available

thus, after the first iteration / at the beginning of the second iteration, we still have d+1 groups (no new groups are added), k=1 (one person has been assigned to a group), and no one is in a group with someone they don't want to work with

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.

- setup and main steps
  - create d+1 empty groups
  - for each person, put them into a group not containing someone they don't want to work with
- 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
- maintain the loop invariant
  - show that if the loop invariant holds at the beginning of iteration k, it still holds after iteration k (at the beginning of iteration k+1)

assume that 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 that there are at most d+1 groups

consider person k+1 – since there are at most d people they don't want to work with and d+1 groups, there is at least one group they can be assigned to without ending up in a group with someone they don't want to work with … as long as "don't want to work with" is mutual – if there are d people person k+1 doesn't want to work with but other people that don't want to work with them, we may not be able to find a group for person k+1 without creating more than d+1 groups

thus, after k+1 people have been assigned to groups, there are at most d+1 groups and no one is in a group with someone they don't want to work with (as long as there are at most d people that can't be in a group with any given person)

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.

- setup and main steps
  - create d+1 empty groups
  - for each person, put them into a group not containing someone they don't want to work with
- 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
- final answer
  - show that if the loop invariant holds when the loop exits, the final answer is correct

the loop exits when all of the people have been added to groups i.e. k=n

the loop invariant being true each time the loop condition is checked means that when the loop exits, the first k=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 … i.e. everyone is in a group, no one is in a group with someone they don't want to work with, and there are at most d+1 groups

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.

Determine efficiency. Evaluate the running time and space requirements of the algorithm.

- *Implementation.*
  Identify data structures and, as necessary, specific implementations of those data structures to efficiently support the algorithm. Also fill in any algorithmic details that are needed in order to establish the running time.
- *Time and space.*
  Assess the running time and space requirements of the algorithm given the implementation identified.
- *Room for improvement.*
  Are the targets met? Is it necessary to do better? If improvements in running time and/or space are needed, identify possible avenues for improvement.

- input
  - n people
  - for each person, the up to d other people they don't want to work with
- main steps
  - for each person, put them into a group not containing someone they don't want to work with