Checkpoint 1

Overview of developments

- A graph is represented using a list of adjacency lists
- Definitions of paths and reachability in a graph
- Generic interface for path search functions: graph -> list of startingpoints -> parent_t where parent_t maps each node to its parent in the tree of explored paths. The list of "shortest" paths to each reachable node can be trivially generated from it (and we do so in the statements of lemmas).
- Two implementations of BFS (with proofs of termination)
- A definition of correctness for BFS (not particularly elegant)

Process and problems solved

In the process of choosing a way to represent graphs, different representations were both considered and implemented. The final version was chosen according to what we guessed would be easiest for proving the termination of BFS.

When implementing BFS, again, different approaches were considered, ranging from just rewriting the search algorithm code that would be written in C in Coq to slightly modifying the definition of BFS to have code be as simple as possible. One of the main issues encountered was proving that the algorithm terminates. When trying to just rewrite imperative code in Coq, arguing that the function terminates seemed to almost depend on proving its correctness (that all nodes are touched once). A different approach with slightly modified definition of BFS attempted to avoid the termination issue altogether by just defining one step of the algorithm, taking as an argument fuel and taking the corresponding number of steps. Neither of them were ideal.

The final version was chosen to be a compromise between several attempts, looking fairly similar to what the corresponding imperative pseudocode would be, but at the same time strictly decreasing on one of its arguments, so termination is not a big issue. Additionally, we made sure that Dijsktra's algorithm (and A*) could be implemented using a similar strategy. We hope that we will be able to use significant parts of both the code and the proof of termination, and maybe other proofs.

The BFS correctness definition progressed through several stages as well. The current version seems to include all the necessary details, but as we thought the same about the three-odd previous attempts, we are not really sure. Nevertheless, we think the specification we have is meaningful even if it is not the strictest possible. For improving it, the idea is to break the big definition of correctness into several smaller ones that must all be satisfied while keeping in mind that the pieces can be proved separately.

Parts yet to do

The following big parts of the final project are yet to be done: * Proving the correctness of BFS If there is enough time after proving the correctness of BFS: * Implementing Dijkstra's algorithm * Defining correctness of Dijkstra's algorithm (this would be based on the correctness of BFS and ideally differ as little as possible) * Proving the correctness of Dijkstra's algorithm