MAC 281 Discrete Structures Spring 2020 ()

Project: Blow-Ups of Graphs

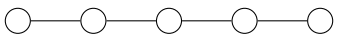
Instructor: Prof. Oehrlein

- The project is graded as successful or not yet successful. You can do revisions on projects.
- Successful completion of this project meets Learning Targets C7, C8, and S3.

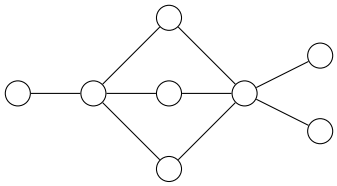
This project builds off the graph theory portion of the course, using student knowledge of bipartite graphs and graph theory proofs. I tried to structure the project like how a mathematician might approach a new research idea. The project starts with a definition and some examples, and the student then generates some more examples. The project then asks the students to start thinking about certain families of graphs, in particular bipartite graphs. From there, I left the questions more open-ended to see what students can figure out. I think there's a nice characterization to answer question 7, but I haven't proven it! The final question suggests an extension of this idea.

What's a Blow-Up?

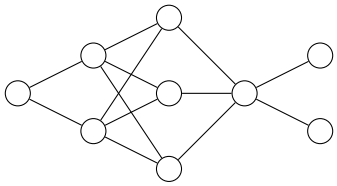
Suppose we start with a path on n vertices. For example, below is P_5, the path on five vertices.



A **blow-up** of the path replaces each vertex v with some number of vertices, not connected to each other but all connected to all of v's neighbors (or their replacements). We write how many vertices we replace each vertex with in an array. So, for example, $[1, 1, 3, 1, 2]$ would look like:



And $[1, 2, 3, 1, 2]$ would look like:



To Turn In

1. Draw some more blow-ups of paths of different lengths. What does $[2, 2, 2, 2]$ look like? What about $[3, 1, 1, 1, 1, 2]$?
2. A path is always a tree. Draw some other blow-ups of paths that are trees. What kinds of arrays give us blow-ups that are trees? Describe the family trees that can be blow-ups of paths.
3. A graph is bipartite if we can color its vertices with two colors such that adjacent vertices have different colors. Prove by mathematical induction on n that the path of length n, P_n, is bipartite for all $n \geq 1$.

4. Using that paths are bipartite, prove that all blow-ups of paths are bipartite.
5. Trees are bipartite, but some trees aren't blow-ups of paths. Are all other bipartite graphs blow-ups of some path? Provide a proof or a counterexample.
6. Draw some new graphs that are definitely not blow-ups of a path.
7. If I were to hand you an arbitrary graph, how would you tell whether it was the blow-up of a path or not? (If you think you couldn't be entirely sure, what are some cases where you would know?)
8. Here, we looked at blow-ups of paths. How would you define the blow-up of a cycle? Draw some examples. (Note: blow-ups of paths can include cycles, but here we're looking at blow-ups *of* cycles.)

MAC 281 Discrete Structures Spring 2020 ()

Project: A* Search

Instructor: Prof. Oehrlein

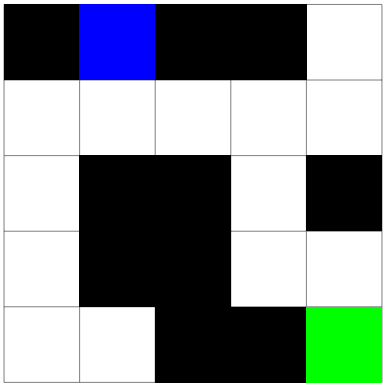
- The project is graded as successful or not yet successful. You can do revisions on projects.
- Successful completion of this project meets Learning Targets C10, S6, and an additional supplementary learning target.

This project builds off the final portion of the course, graph algorithms. It asks them to learn about an algorithm not covered in class and build material that might be used to briefly introduce the algorithm to a classmate: the key differences from a known algorithm, two examples, and a description of why we might use this algorithm. If I were to teach this course again, I would consider having all students complete a project like this for different algorithms and build some kind of wiki/book.

What is A* Search?

A* search (pronounced "A star") is a way to find a path between a starting position and a goal position on a graph or a map. It uses many of the ideas of Dijkstra's algorithm, but with a preference for paths that seem like they'll be shorter for reaching the goal position.

Suppose that in the grid below, we start at the blue square, want to reach the green square, and can't walk through the black squares; they're obstacles. (The map below was suggested for this problem by Spring 2020 student George Webster.)



Dijkstra's algorithm would help us find the shortest path from blue to green, but it might do a lot of unnecessary work along the way. A* search can help.

To Read

In the other item in the folder, there are links to a few readings.

- The introduction summarizes Dijkstra's algorithm, greedy breadth-first search, and A* search on grids. Read all of this.
- The page on heuristics describes different approaches to the heuristic function used in A* search. Skim this, paying closest attention to the sections labeled "Manhattan distance" and "Diagonal distance."

To Turn In

For each of the following, write your answer in a few sentences (showing/explaining your work), and include any helpful drawings.

1. Suppose you're talking to someone who understands Dijkstra's algorithm but has never heard of A* search. Explain the key differences to them in a few sentences.

2. For the example grid above, suppose you can only move in four directions (not diagonally). Using the appropriate heuristic function, write out the steps of A* search for this example.

3. Now suppose you can move diagonally as well. Using the appropriate heuristic function, write out the steps of A* search for this example.

4. Answer either this question or the next one: A* search is commonly used in video game programming. Why do you think that is? (If you're interested in more information, "pathfinding" is a useful search term here.)

5. Answer either this question or the previous one: A* search was developed in robotics. Why do you think it's relevant in that field? (If you're interested in more information, "path planning" or "motion planning" are useful search terms.)