COMP 285: Analysis of Algorithms
North Carolina A&T State University — DFS and BFS
Dr. Allison Sullivan — Due: In-Class Exercise

DFS and BFS

Problem 1: Weighted Shortest Paths

Let G be an undirected weighted graph with weight function $w : E \rightarrow \{1, 2, 3, ..., k\}$, where k is a small constant. Let u be an arbitrary node in G. Provide an algorithm which computes the length of the shortest paths between u and any other node in the graph in time $O(k(n + m))$.

Problem 2: Making Teams

Over the course of the year, the sixth grade teachers build up information about which students work well together (and, more importantly, which students do not work well together). Assume the teachers maintain a list r that contains pairs of students who *do not* work well together. (That is, if Tom and Sally do *not* work well together, the pair $\langle$Tom, Sally$\rangle$ will be in r.) Assuming n students, the maximum size of r is $n(n-1)$ (none of the students work well together); the minimum size of r is 0 (everyone works well with everyone else).

On a given day in the year, a teacher might take this list of pairs and try to generate two teams for a class project. Give an $O(n+r)$ time algorithm that determines whether, given the pairs in r today, it is possible to divide the students into two teams such that any pair of students assigned to the same team work well together (i.e., if Alice and Bob are on the same team, then the pair $\langle$Alice, Bob$\rangle$ is *not* in r). That is, the teacher will create Team A and Team B such that every student is assigned to one of the two teams. The teams need not be equally sized, but on a given team, there must be no pair of students that do not work well together.

Briefly justify that the runtime of your algorithm is $O(n+r)$.

Problem 3: Discipline in Groups of Children - Take Two

In the homework, your job was to arrange n rambunctious children in a straight line, facing front. You are given a list of m statements of the form "i hates j". If i hates j, then you do not want to put i somewhere behind j because then i is capable of throwing something at j. (***spoilers:*** *the solution to this was a topological sort*).

Suppose that instead you want to arrange the children in rows such that if i hates j, then i must be in a lower numbered row than j. Give an efficient algorithm to find the minimum number of rows needed, if it is possible.