Introduction to Algorithms: 6.006
Massachusetts Institute of Technology — November 8, 2007
Professors Ron Rivest and Srini Devadas — Handout 15

Problem Set 5

This problem set is due **November 21** at **11:59PM**.
Solutions should be turned in through the course website in PS or PDF form using LaTeX or scanned handwritten solutions, or they may be handwritten and turned in to a member of the 6.006 course staff on or before the due date. Hand-drawn diagrams may also be referenced in your LaTeX writeup and turned in at the next day's recitation.
A template for writing up solutions in LaTeX is available on the course website.

Exercises are for extra practice and should not be turned in.
Exercises:

- Exercise 22.2-3 from CLRS.
- Exercise 22.2-8 from CLRS.
- Exercise 22.3-1 from CLRS.
- Exercise 22.3-8 from CLRS.
- Exercise 22.3-9 from CLRS.

---

1. **(8 points)** Bipartite graphs

   A graph is called *bipartite* if its nodes can each be assigned a color, either red or blue, such that no red node is adjacent to another red node, and no blue node is adjacent to another blue node. Give an efficient algorithm to determine if a graph is bipartite. What is its running time?

2. **(22 points)** Cycles in a directed graph

   You are given a directed graph $G = (V, E)$ in which every vertex has at least one outgoing edge.

   Your task is to find a vertex v that is part of a directed cycle (a cycle is a path of edges from a node to itself).

   (a) **(6 points)** Argue that G must contain at least one directed cycle.

   (b) **(8 points)** Can you use a single BFS to find such a vertex? If so, how? If not, why not?

   (c) **(8 points)** Can you use a single DFS to find such a vertex? If so, how? If not, why not?

3. **(30 points)** 2x2x2 Rubik's Cube

For this problem, you only need to turn in code.

We say that a configuration of the cube is k levels from the solved position if you can reach the position in exactly k twists (quarter-turns, either clockwise or counterclockwise), but you cannot reach the position in fewer twists.

(a) **(15 points)** Use breadth first search to recreate the chart seen on `http://en.wikipedia.org/wiki/Pocket_Cube`. You will write a function that takes `moves`, the set of legal permutations of the cube, and `level`. It should return the number of configurations of the cube `level` levels from the solved position. Using the set of quarter-twists provided to you should give you the same output as column q in the chart.

Note: if your machine does not have much RAM, your program may not work for higher levels. In this case, don't worry, and just add a comment telling us the largest level you solved, and how much RAM you have.

(b) **(15 points)** Given two configurations of the cube, write a function that returns the shortest path between the two configurations. Instead of doing a simple breadth first search as in part (a), use less memory and time by doing a 2-way breadth first search. That is, do two breadth first searches, one starting at each end, and find the configuration where they meet. This will be discussed in recitation.