Problem Set 7 Solutions

Calvin Walker

Problem 1:

(a) This problem is in NP. Consider the following non-deterministic algorithm A:

1. Choose a random path p from $s \to t$
2. Output YES if p uses at most 1 vertex from each Z_i for all $i \in [k]$, otherwise output NO.

If there is no valid st path in G, A always outputs NO regardless of the guess p. If there is a valid st path, then A has a nonzero chance of making this guess and outputting YES. So A is a non-deterministic polynomial time algorithm and this problem is in NP.

(b) This problem is not in NP. To verify a YES answer to this problem, we must check all of the possible subsets of G when removing $\leq k$ edges. There are $\sum_{j=1}^{k} \binom{n}{n-j}$ such subsets, each of which take linear time to verify. So there is no polynomial time verifier for this problem, and it is not in NP.

(c) This problem is in NP. Consider the following non-deterministic algorithm A:

1. Randomly partition the verticies into subsets L and R
2. Count the edges between L and R, and output YES if this number is k, otherwise output NO.

If there is no such cut of G, then A always outputs NO. If there is a valid cut C, then there is a nonzero chace A's random cut equals C and outputs YES. The edges crossing (L, R) can be counted in $O(n^2)$ time so A is a non-deterministic polynomial time algorithm and this problem is in NP.

(d) This problem is not in NP. To verify a YES answer, we must check all the sugraphs with $\geq k$ verticies to ensure there is a clique of size k, and check if there is a clique of size greater than k. There are $\sum_{j=k}^{n} \binom{n}{j}$ such subgraphs which each take polynomial time to verify. So there is no polynomial time verifier for this problem, and it is not in NP.

Problem 2:

Algorithm: Initialize a graph $G = (V, E)$ with a vertex for each space $i \in [n]$. For each each vertex create a directed edge to each of the spaces reachable from it. Weight the edges equal to the negative value of the space represented by the vertex they are directed towards. Then, use Bellman-Ford to find the shortest path using k edges from v_1 to v_n, given by the subproblem $d_k(v_n)$. Return the absolute value of $d_k(v_n)$, which is the maximum total number of points.

Runtime: