1. ***Ice-sliding puzzle.*** Let IceSliding denote the following problem:

> IceSliding
> - ***Input:*** *The $n \times n$ grid together with the positions of all the obstacles, items, entrances, and exits, as well as an integer k.*
> - ***Output:*** *Is there a way to slide across the ice from an entrance to an exit while collecting all the items, using at most k steps/key-presses?*

(a) Prove that IceSliding is in NP.

Solution: The prover can provide an entrance, an exit, and a sequence of key presses. The proof itself will be polynomial in length as the number of key presses needed is at most the number of all possible positions in the map. The verifier can check if the sequence of key presses is indeed at most k steps, and actually takes the character from the entrance to the exit while collecting all the items. The check can be carried out in time polynomial to the size of the puzzle. ■

(b) Prove that we can find a sequence of at most k key presses solving the puzzle in polynomial time given an oracle to IceSliding.

Solution: We prove that the IceSliding problem is self-reducible. Given an oracle that decides any instance of the IceSliding problem, we will find the sequence of k key presses using the following algorithm. Given an instance of IceSliding, create four new instances as follows:

- Keep the whole map unchanged.
- Move the character to a new position by simulating the four possible key presses, one for each direction; set the new position of the character as the new entrance. This creates four instances, one for each direction.
- Set the original exit as exit.
- Set the integer parameter as $k - 1$.

Now query the oracle with all four new instances. If the oracle answers *no* to all of them, then the original instance has no solutions. Otherwise, choose an instance which the oracle answered *yes* to, and record the corresponding key press. Now repeat the algorithm on such yes-instance, unless the number of remaining stpes is already 0, in which case we have the sequence of k key presses recorded.

Generating each new instance takes time proportional to the input size, and we generate $4k$ new instances in total. This implies that we find the right sequence of k key presses in polynomial time. ■

(c) Either prove that IceSliding is NP-hard, or solve IceSliding in polynomial time.

Solution: It turns out that if we drop the restriction on the number of key presses, then the problem becomes solvable in polynomial time.

> IceSlidingUnlimited
> - ***Input:*** *The $n \times n$ grid together with the positions of all the obstacles, items, entrances, and exits.*
> - ***Output:*** *Is there a way to slide across the ice from an entrance to an exit while collecting all the items?*

We will solve IceSlidingUnlimited in polynomial time. Create a (directed) *traversal* graph G based on the IceSlidingUnlimited instance: Create a vertex for each possible staying positions on the map. Add a directed edge from one position to the other if the latter can be reached by one key press from the former. For each item in the map (which we assumed to be an ice-free tile), put an item on the corresponding vertex in the graph. Name the vertex of the starting position as s and the one of the ending position as t. The graph has at most n^2 vertices

and $O(n^2)$ edges (because each vertex has out-degree at most 4), and can be constructed in polynomial time.

Now the original problem turns into whether we can walk from s to t while collecting all the items. We compute the *strongly-connected components* in G, that is, equivalent classes of vertices in G under the "reachable" relation: two vertices x and y are in the same component if and only if there is a directed path from x to y and a directed path from y to x. This can be computed in polynomial time by performing graph-traversal algorithm (like depth-first search) on every vertex and group the vertices based on the set of vertices reachable. Order the components as $C_1, \ldots, C_\ell$ where every edge between two different components is directed from the smaller-index one to the bigger-index one. Let C_i and C_j be the components containing s and t respectively. Now there is a path from s to t collecting all the items if and only if $s \le t$ and all items lie on vertices in $C_i \cup \cdots \cup C_j$. This can be checked in time linear to G. ■

Rubric: It turns out the the problem is *subtle*. On one hand, IceSliding is NP-hard become we can reduce HamiltonianPath to it, by setting k to be just right so that there is just enough key presses to collect all the items; on the other hand, once the restriction on k is dropped, the problem can be solved in polynomial time.

In the ice-sliding puzzle, the character might *reuse* the path they have already taken. In such case, if we construct and IceSliding puzzle based on a graph, when we convert the key press sequence to a path in the graph, such path might not be Hamiltonian as some vertices might be visited more than once. One can resolve the issue by introducing *single-pass tiles* or putting a restriction on the maximum number of moves/key-presses, and indeed in those cases the problem becomes NP-hard.

One can actually compute the strongly-connected components in G using one single depth-first search in linear time (standard CS 31 material); but this is not necessary here as our goal is just polynomial time.

Standard 5-point grading scale (plus deadly-sins and sudden-death rules) ***for each subproblem*** for first two subproblems, each scaled to 2.5 points; ***everyone gets full credit on the last question*** because my initial official solution was incorrect.