Theoretical Computer Science Coursework

Jake Mortimer

December 6, 2019

1 Complexity and Approximability

1. **Question 1**

   (a) i. Prove that SHORT PATH $\in$ **NL**
   Given an instance of the SHORT PATH we need to show that there is a non-deterministic Turing Machine that decides it in log-space. Starting at the start vertex s of G we have a counter c in memory. At each vertex in the path from s we non-deterministically guess the next vertex. If we reach t and the value in the counter $c \leq k$ then accept. If $c > k$ then reject. This obviously solves SHORT PATH in logarithmic space because the counter can have a maximum value of $|V|$. If the counter were greater than $|V|$ then this would mean that we have visited one vertex of the input graph more than once and, hence, there is a cycle. $\therefore$ We could reduce the path length to some path length $\leq |V|$ by simply removing the cycle $\Rightarrow$ c can be stored in logarithmic space and SHORT PATH $\in$ **NL**.
   ii. Describe the reduction REACHABILITY $\leq_L$ SHORT PATH
   Given an instance of the REACHABILITY problem; a graph $G = (V, E)$ and two vertices s, t, we need a reduction in logarithmic space to the SHORT PATH problem. We do this by creating a new graph $G' = (V', E')$ where $V' = E \cup \{s', t'\}$ and $E' = \{((u_1, u_2), (v_1, v_2)) \in V'^2 \mid u_2 = v_1\} \cup \{(s', (u_1, u_2)) \in V'^2 \mid u_1 = s\} \cup \{((u_1, u_2), t') \in V'^2 \mid u_2 = t\}$ and k in the instance of SHORT PATH would be set such that $k = |V'|$ because if k was greater than $|V'|$ then the algorithm would have visited one vertex more than once $\Rightarrow$ we are in a cycle and if $s \rightarrow t$ on this path then could be reached on the same path with the cycle removed in smaller length. This can clearly be done in logarithmic space.
   iii. Prove the correctness of the above reduction
   To prove the correctness of the above reduction we need to prove the correspondence of yes instances. This means that we have a yes instance of REACHABILITY $\iff$ we have a yes instance of SHORT PATH.
   ($\Rightarrow$) Suppose we have a yes instance of REACHABILITY, this

means a graph G, and two vertices s, t such that there exists a path $s \rightarrow t$ in G. Take the graph G and construct the graph G' using the reduction above. Suppose that the path $s \rightarrow t$ is the set of edges $P \subseteq E$. We will now construct a similar path $P' \subseteq E'$. In E' there is an edge $(s', (s, v))$ and an edge $((u, t), t')$ for some edges in $(s, v), (u, t) \in P$, and these will be placed into the path P'. For every pair of successive edges $(i, j), (j, w) \in P$ we have $((i, j), (j, w)) \in P'$. We terminate when we see the edge $((i, j), (j, t)) \in P'$ because this means our path go $s \rightarrow t$. We would always reach this termination condition because we are simply traversing the path P from G. When we have covered all edges in P we can see that obviously $|P'| \leq k$ otherwise, there would be a cycle as discussed above. $\therefore$ yes instance for REACHABILITY $\Rightarrow$ yes instance for SHORT PATH via our reduction.
($\Leftarrow$) Suppose we have a yes instance of SHORT PATH which conforms to the reduction above. This means that we have a path $P' \subseteq E'$ that goes $s' \rightarrow t'$. We will now construct a path $P \subseteq E$ in G. We first construct G iteratively over the vertices of G'; the edge set is $E = V' \setminus \{s', t'\}$, and the vertex set V is constructed in the obvious iterative way; disregarding duplicates. When constructing the path P over G we disregard the edge $(s', (s, v)) \in P'$ for some vertices $(s, v), (u, t) \in V'$. Iterate over the elements of P', for example $((u, v), (i, j)) \in P'$, and add (u, v) to P. This path P will include all of the edges from $s \rightarrow t$ in G which is a path from s to t, and hence s is reachable from $t \Rightarrow$ we have a yes instance of reachability. $\therefore$ yes instance for SHORT PATH $\Rightarrow$ yes instance for REACHABILITY.
$\therefore$ SHORT PATH is **NL**-complete $\Box$

(b) Show that SHORT-PATH-2019 $\in$ **L**
To show that SHORT-PATH-2019 $\in$ **L** we have to construct a logarithmic-space transducer that solves this problem. This means that given an instance of SHORT-PATH-2019; a graph G, vertices s, t and a value $0 < k \leq 2019$, we have to determine if G has a path P from $s \rightarrow t$, such that $|P| \leq k$.
If $|E| \leq 2019$ then this problem is trivial via a backtracking approach. Start from vertex s and build towards a full solution, if this is not possible then backtrack, overwriting edge memory that

you have backtracked from. As such you never exceed the 2019 memory cells. This means that we use a constant amount of memory in this case and reach a solution deterministically.
If $|E| > 2019$ then we can enumerate all 2019-combinations of the edge set E and run on these one at a time; using the constant-space backtracking approach above. We make sure not to store these enumerated 2019-combinations, instead we run the algorithm each time we need a combination and take only one of the combinations each time in order: we assume deterministic way that we enumerate these combinations.
$\therefore$ SHORT-PATH-2019 $\in \mathbf{L}$ $\Box$

2. **Question 2**

   (a) Prove unconditionally that TWO CLIQUES is not **PSPACE**-complete with respect to log space reductions.

   We can show that TWO CLIQUES problem is a member of **L**. This can be shown by the proving the reduction TWO CLIQUES $\leq_L$ BIPARTITE which is a member of **L**. BIPARTITE is defined as follows:
   Instance: Graph $G = (V, E)$
   Question: Is the graph G bipartite?
   The reduction is simple; given an instance of TWO CLIQUES (an undirected graph G) we take the complement of the graph, resulting in graph $\bar{G}$ which we give as the instance of BIPARTITE.
   Proof that BIPARTITE $\in$ L
   We will be using Reingold's Theorem that $s - t$ connectivity in an undirected graph is in **L**, seen here - *SL and Reingold*. With the knowledge that UPATH (does there exist a path from $s - t$ in G?), we prove that BIPARTITE $\in \mathbf{L}$ by reducing to co-UPATH (The complement of UPATH; there is no path from $s - t$ in G'), as $\mathbf{L} = co - \mathbf{L}$ trivially.
   For a given bipartite graph $G = (V, E)$ we can define the graph G' such that $V' = \{v_1, v_2 \mid v \in V\} \cup \{s, t\}$ and $E' = \{(u_1, w_2), (u_2, w_1) \mid (u, w) \in E\} \cup \{(s, v_1) \mid \forall v \in V\} \cup \{(v_2, t) \mid \forall v \in V\}$. This reduction can clearly be done in logarithmic space. Now we have a reduction we need to prove the correspondence of yes-instances:
   ($\Rightarrow$) Given a yes instance of BIPARTITE, a graph G, we can use

the reduction above to get the graph G'. We substitute this graph into the co-UPATH problem. We pass in the vertices s, t, and if it accepts; there are no paths between any of the expanded vertices. This is the case because for there to be a path between s and t there would need to be a path between two vertices such as $v_1, v_2 \in V'$ there must be an edge within one of the sets of vertices of G.
Assume for a contradiction that there was not an edge between any two vertices of the same set in G but there was a path $v_1 \to v_2$. The same disjoint sets in G can be applied to G', and we have established that there are no edges within a set in G, similarly, this is a fact for these sets in G'. This means that v_1, v_2 would have to be connected by some vertex in the opposite set $(v_1, d_1), (v_2, d_1)$ for there to be a path between them. This is not possible: it would mean that in the original graph G there would have been the edges $(v, d), (d, v) \in E$, and in an undirected graph there would only be one of these edges. Therefore, we have a contradiction meaning that for there to be a path $v_1 \to v_2$ there would have to be an edge within the bipartite sets in G, and in turn a yes instance of BIPARTITE $\Rightarrow$ yes instance of co-UPATH.
($\Leftarrow$) Given a yes instance of co-UPATH which corresponds to the above reduction we transform G' to G. For the instance to be a yes instance, there can be no path $s \to t$ which means no pair of vertices $v_1, v_2 \in V'$ can have a path between them, otherwise there would be a path from $s - t$. When we reduce G' to G we first remove s, t, this leaves two sets of disjoint vertices with no edges between vertices of the same set. Assume for a contradiction that the two sets had an edge within one of the sets, this would mean that (v_1, w_2) would be an edge for some $v_1, w_2 \in V'$ which are in the same set. From the construction, this would mean that there is a path from $s - t$. $\therefore$ we have a contradiction $\Rightarrow$ there can be no edges within these disjoint sets. We then construct $G = (V, E)$ from the pairs of vertices $v_1, v_2 \in V'$, and the edges $(v_1, w_2), (v_2, w_1) \in E'$ as described in the reduction. This results in a graph G which, trivially, is bipartite: two disjoint sets maintained from G' and there can be no edges between two vertices of a set. $\therefore$ yes instance of co-UPATH $\Rightarrow$ yes instance of BIPARTITE.
$\therefore$ BIPARTITE $\leq_L$ co-UPATH $\leq_L$ UPATH $\Rightarrow$ BIPARTITE $\in \mathbf{L}$

$\square$
Continuing from before proof...
We now prove the reduction TWO CLIQUES $\leq_L$ BIPARTITE.
Proof of Reduction
To prove the correctness of this reduction we need to prove the correspondence of yes instances. This means that we have a yes instance of TWO CLIQUES $\iff$ we have a yes instance of BIPARTITE.
($\Rightarrow$) Given a yes instance of TWO CLIQUES G, V_1, V_2 where G is the graph, and V_1, V_2 are the two cliques that the graph can be subdivided into. We take the complement of G, $\bar{G}$ as in the reduction. In the initial yes-instance for TWO CLIQUES V_1 and V_2 are completely connected sub-graphs; each vertex of either is connected to every other vertex of the same clique. When we take the complement of G, the edge set contains the edges which were not present in the initial graph. This means that V_1, V_2 will have no edges internally in $\bar{G}$ and only edges between vertices of opposite sets. As such the graph $\bar{G}$ is bipartite. $\therefore$ yes instance of TWO CLIQUES $\Rightarrow$ yes instance of BIPARTITE.
($\Leftarrow$)Given a yes instance of BIPARTITE $\bar{G}, V_1, V_2$ we take the complement of the graph $\bar{G}$ and get G. By the reverse of the above argument it is clear that a yes instance of BIPARTITE $\Rightarrow$ yes instance of TWO CLIQUES.
$\therefore$ TWO CLIQUES $\in$ **L** $\square$
Continuing from before proof...
We now have that TWO CLIQUES $\in$ **L**. It is known that **L** $\subseteq$ **NL** $\subseteq$ **PSPACE** because if a problem is solvable in logarithmic space on a deterministic machine it is obviously solvable in logarithmic space on a *non*-deterministic machine $\Rightarrow$ **L** $\subseteq$ **NL**. It is also clear to see that if a problem is solvable in logarithmic space on a non-deterministic machine then it is solvable in polynomial space on a deterministic machine by Savitch's Theorem $\Rightarrow$ **NL** $\subseteq$ **PSPACE**.

If **L** $\subseteq$ **NL** and **NL** $\subseteq$ **PSPACE**, then the above sequence of set containments holds true.
We now attempt to derive a contradiction: assume that TWO CLIQUES is **PSPACE**-complete. This would mean that **L** =

PSPACE $\Rightarrow$ **PSPACE** $\subseteq$ **NL**. We already have that **NL** $\subseteq$ **PSPACE**, so combining these we get that **NL** = **PSPACE** which we can prove not to be true.

Proof of NL $\neq$ PSPACE

First, we need to define space constructible functions. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $f(n) \geq log(n)$ is called space constructible if the function that maps 1^n to the binary representation of $f(n)$ is computable in space $O(f(n))$ From this it is clear that if $f(n)$ and $g(n)$ are space constructible functions, and $f(n) = o(g(n))$ then $SPACE[f] \subset SPACE[g]$. By Savitch's Theorem, **NL** $\subseteq SPACE[log^2(n)]$. Since $log^2(n) = o(n)$, we have $SPACE[log^2(n)] \subset SPACE[n]$ $\therefore$ **NL** $\neq$ **PSPACE** $\square$

Continuing from before proof...

This is a contradiction $\therefore$ we can say (unconditionally) that TWO CLIQUES is not **PSPACE**-complete $\square$

2 Algorithmic Game Theory

3. **Question 3**

   (a) Find all the pure strategy equilibria.
   There are $\binom{n}{2}$ pure strategy equilibria in this game; where exactly two people are helping. This is the case because in these circumstances if one of the players were to stop helping then they would decrease their score from 9 to 8 $\Rightarrow$ this would no longer be an equilibrium. Also, if one of the players who was currently not volunteering changed to volunteering then they would decrease their score from 10 to 9 $\Rightarrow$ this would no longer be an equilibrium.

   (b) Find all the symmetric mixed strategy equilibria.
   Note: $P(n)$ is the probability function describing the probability of n players volunteering.

   Score for volunteering: 9
   Score for not volunteering:
   $10(P(\geq 2)) + 8(P(< 2)) = 10(P(\geq 2)) + 8(1 - P(\geq 2))$
   $10P(\geq 2) + 8(1 - P(\geq 2)) = 9$
   $8(1 - P(0) - P(1)) + 10(P(0) + P(1)) = 9$

$10P(0) - 8P(0) + 10P(1) - 8P(1) + 8 = 9$
$2(P(0) + P(1)) = 1$
$P(0) + P(1) = \frac{1}{2}$
Now using the binomial distribution ($\binom{n}{r}q^r(1-q)^{n-r}$) we can see that the probability of $P(0)$ and $P(1)$ are as follows:
$P(0) = p^{n-1}$, $P(1) = (n-1)(1-p)p^{n-2}$
Using these we can create the following polynomial to attempt to solve for p which is the probability of **not** volunteering:
$p^{n-1} + (n-1)(1-p)p^{n-2} - \frac{1}{2} = 0$
$(4-2n)p^{n-1} + 2(n-1)p^{n-2} - 1 = 0$
$2(n-2)p^{n-1} + 2(1-n)p^{n-2} + 1 = 0$
Intermediate Value Theorem can now be used to show that a solution to this polynomial exists:
$F(p) = 2(n-2)p^{n-1} + 2(1-n)p^{n-2} + 1$
$F(0) = 1$
$F(1) = 2(n-2) + 2(1-n) + 1 = -4 + 2 + 1 = -1$
$\therefore\ y = F(p)$ is such that $F(1) < F(p) < F(0)$ meaning there is some c between $0 < c < 1$ for which c is a solution of $F(p)$ and since $F(p) = 0$ above must be solution to $F(p) = 0$ for our symmetric mixed strategy; our probability $\square$

4. **Question 4**
Show that a Boolean function $f(p_1, p_2)$ can be computed by a three-player game. That is, each player has two pure strategies that are the Boolean values T and F, and there's a pure-strategy Nash equilibrium if and only if the strategy of player three, $p_3 = f(p_1, p_2)$ where p_1 and p_2 are the strategies of players one and two, respectively.

The construction of the game is as follows:
We have three players 1,2,and 3. Each player has two pure strategies T and F. The players receive a score of 1 if the strategies p_1, p_2, p_3 are such that $p_3 = f(p_1, p_2)$, and a score of 0 otherwise.
Using the **above** construction we want to prove that there is a pure strategy Nash Equilibrium $\iff$ the strategy of player 3, p_3, is such that $p_3 = f(p_1, p_2)$.
$(\Rightarrow)$
Pure Strategy Nash Equilibrium (NE) $\Rightarrow p_3 = f(p_1, p_2)$
Suppose that there exists some pure strategy NE such that $p_3 \neq f(p_1, p_2)$.

This would mean that the score for the players would be 0 as such there would be a unilateral change in strategy that results in a higher score $\Rightarrow$ this is not a pure strategy NE. $\therefore$ we have a contradiction which means that Pure Strategy NE $\Rightarrow p_3 = f(p_1, p_2)$.
$(\Leftarrow)$
$p_3 = f(p_1, p_2) \Rightarrow$ Pure Strategy Nash Equilibrium
If $p_3 = f(p_1, p_2)$ then the players would get a score of 1 which is the maximum score in the game, this means that any change in strategy would result in a lower or equal score $\Rightarrow$ this is a pure strategy equilibrium. $\therefore p_3 = f(p_1, p_2) \Rightarrow$ Pure Strategy NE.
As demonstrated we have proven the correspondence of $p_3 = f(p_1, p_2)$ and a pure strategy NE, and in turn we have shown that a Boolean function $f(p_1, p_2)$ can be computed by a three-player game $\square$