B03902062 資工二 董文捷

Problem 1

(1) Use mathematical induction to prove

1. For n = 1 $S(v) \leq \alpha \times S(f(v)),\ S(f(v)) \geq S(v) \times (1/\alpha)$

2. Assume for n = k the equation holds, which means $S(f^k(v)) \geq S(v) \times (1/\alpha)^k$

3. For n = k + 1, since $f^k(v)$ is a child of $f^{k+1}(v)$,

$\alpha \times S\left(f^{k+1}(v)\right) \geq S(f^k(v)) \geq S(v) \times (1/\alpha)^k$, so

$S(f^{k+1}(v)) \geqq S(v) \times (1/\alpha)^{k+1}$, the equation also holds

By mathematical induction, the equation holds for all $n \geq 1$

(2) Pick a leaf at the bottom as v, and the root can be represented as $f^{height}(v)$, according to the equation in (1),

$S\left(f^{height}(v)\right) = N \geq S(v) \times (1/\alpha)^{height} = 1 \times (1/\alpha)^{height}$.

Since $(1/\alpha) \geq 1$, height $\leq \log_{\frac{1}{\alpha}} N = \log_{\frac{1}{\alpha}} 10 \times logN$, the height is $O(logN)$

(3)

Problem 2

References :

https://en.wikipedia.org/wiki/Boolean_satisfiability_problem

http://www.cs.princeton.edu/~wayne/cs423/lectures/reductions-poly-4up.pdf

(1) A is satisfiable if and only if (y ∨ A) ^ (¬y ∨ A) is satisfiable since only one of y and ¬y will be 1. With this condition, we expand a clause with k literals to two clauses with k + 1 literals by adding new variable y. Therefore, by repeating this procedure, we can use polynomial time (O(mk)) to expand every 3 literals clause to k literals clause, reduce 3-CNF-SAT to k-CNF-SAT.

(2)

(3) This problem is actually a 3-coloring problem, since a 3-CNF-SAT problem can be reduced to a 3-coloring problem, the 3-coloring problem is a NP-hard problem. Furthermore, it is a NP problem. Therefore, a 3-coloring problem is a NP-complete problem.

To reduce a 3-CNF-SAT problem to a 3-coloring problem

1. Create triangle R(false) G(true) B
2. Create nodes for each literal and connect to B (colored R or G)
3. Create negation for each literal, and connect literal to its negation (colored opposite)
4. For each clause, add “gadget” of 6 new nodes and 13 new edges
If top row has green(true) node, then it is 3-colorable, therefore satisfiable.
The above reduction function can work in polynomial time, so 3-coloring problem is NP-hard.

3-coloring problem is a NP problem because there are at most $\frac{N(N-1)}{2}$ combinations of edges, by checking color of vertices connected by an edge, we can verify the certificate’s correctness, which can be verified in $O(N^2)$.
3-coloring is a NP problem and is NP-hard, so it is a NP-complete problem. (4)

Problem 3

(1)

(a) Simple connected graph isomorphism problem is reducible to graph isomorphism problem because simple connected graph is also one kind of graph, it can be solved as graph isomorphism problem, too.
(b) Use DFS to find connected components of a graph, for each combination of a connected component in G and a connected component in H, call simple connected graph isomorphism problem, if they are isomorphic, connect an edge with capacity 1. Then connect source to components in G, connect sink to components in H. If each simple connected graph in G is isomorphic to a simple connected graph in H, the maximum flow should be #components, otherwise, G and H are not isomorphic. We have to call simple connected graph isomorphism problem at most $O(N^2)$ time (at most N components), extra time of maximum cardinality bipartite matching is $O(N^3)$, which are all polynomial, so graph isomorphism problem is reducible to simple connected graph isomorphism problem.
Simple connected graph isomorphism problem is GI-complete.

(2)

(a) “Determine isomorphism and give the solution” is reducible to graph isomorphism problem. First, call graph isomorphism problem to see whether G and H are isomorphic. A solution can be found by trying

mapping relation one by one. For each node n1 in G, test all unmapped node n2 in H, make a new graph G' by deleting n1 and edges containing it, make a new graph H' by deleting n2 and edges containing it. Call graph isomorphism problem, if G' and H' are isomorphic, then n1->n2 is a possible mapping, delete n1 and n2 on G and H, otherwise, n1->n2 is not a possible mapping, try other n2 to find a mapping for n1. By repeating this process at most $\frac{N(N+1)}{2}$ times, a solution can be found. All extra time including deleting nodes and edges is polynomial. So "Determine isomorphism and give the solution" is reducible to graph isomorphism problem.

(b) Graph isomorphism problem is reducible to "Determine isomorphism and give the solution" because we can just neglect the solution and focus on isomorphism result.