HW5

宋子維 B03902042

Problem 1

(1) Induction on n . When $n = 1$, let $w = f(v)$. Since w is α -balanced,

$$S(l(w)) \le \alpha S(w) \text{ and } S(r(w)) \le \alpha S(w)$$

Since $w = f(v)$, $l(w) = v$ or $r(w) = v$. Thus

$$S(v) \le \alpha S(f(v)) \;\Rightarrow\; (1/\alpha)S(v) \le S(f(v))$$

Suppose it holds when $n = k$, that is,

$$(1/\alpha)^k S(v) \le S(f^k(v))$$

Let $w = f(f^k(v)) = f^{k+1}(v)$. Since w is α -balanced,

$$S(l(w)) \le \alpha S(w) \text{ and } S(r(w)) \le \alpha S(w)$$

Since $w = f^{k+1}(v)$, $l(w) = f^k(v)$ or $r(w) = f^k(v)$. Thus

$$S(f^k(v)) \le \alpha S(f^{k+1}(v))$$

By induction hypothesis,

$$(1/\alpha)^k S(v) \le S(f^k(v)) \le \alpha S(f^{k+1}(v)) \;\Rightarrow\; (1/\alpha)^{k+1} S(v) \le S(f^{k+1}(v))$$

By induction, in a α -balance tree, $S(f^n(v)) \ge S(v)(1/\alpha)^n$ for all n .

(2) Let v be the root with level 0 and u be the node with highest level h . Thus, we have

$$v = f^h(u)$$

and the height of α -balanced tree is $h + 1$.
By (1),

$$N = S(v) = S(f^h(u)) \ge S(u)(1/\alpha)^h = (1/\alpha)^h$$

since u is the deepest node.
Take $\log$,

$$\log N \ge h \log(1/\alpha) \;\Rightarrow\; h \le \frac{\log N}{\log(1/\alpha)} = \mathcal{O}(\log N)$$

Then the height of tree is

$$h + 1 = \mathcal{O}(\log N)$$

(3)
Reference:
http://people.scs.carleton.ca/~maheshwa/Honor-Project/scapegoat.pdf
http://www.akira.ruc.dk/~keld/teaching/algoritmedesign_f08/Artikler/03/Galperin93.pdf
http://sharable-stuff.googlecode.com/svn/trunk/Classes/2008-Fall/CMSC651/homework/2008-09-25/homework.pdf
https://carlstrom.com/stanford/cs161/ps5sol.pdf
By definition,

$$S(l(v)) \le \frac{1}{2}S(v) \text{ and } S(r(v)) \le \frac{1}{2}S(v)$$

for all v in $1/2$ -balanced tree. Then we have

$$S(l(r)), S(r(v)) \le \frac{1}{2}S(v) = \frac{1}{2}(S(l(v)) + S(r(v)) + 1)$$

Thus,

$$\begin{aligned} & \frac{1}{2}S(l(v)) \le \frac{1}{2}(S(r(v)) + 1) \text{ and } \frac{1}{2}S(r(v)) \le \frac{1}{2}(S(l(v)) + 1) \\ \Rightarrow\ & S(l(v)) \le (S(r(v)) + 1) \text{ and } S(r(v)) \le (S(l(v)) + 1) \\ \Rightarrow\ & -1 \le S(l(v)) - S(r(v)) \le 1 \\ \Rightarrow\ & \Delta(v) = |S(l(v)) - S(r(v))| \le 1 \end{aligned}$$

Then potential is 0 because $\Delta(v) \le 1$ for all node v in $1/2$ -balanced tree.

Given the subtree T rooted at the highest node u that is not α -balanced and $S(u) = m$. Without loss of generality, suppose

$$S(l(u)) > \alpha m \text{ and } S(r(u)) \le \alpha m$$

Then

$$S(r(u)) = m - S(l(u)) - 1 < (1 - \alpha)m$$

which implies that

$$S(l(u)) - S(r(u)) > (2\alpha - 1)m$$

And we have

$$\Phi(T) = c \sum_{v \in T : \Delta \ge 2} \Delta(v) \ge c(S(l(u)) - S(r(u))) \ge c(2\alpha - 1)m$$

After rebuilding, the potential becomes 0 . Hence the potential difference is at least $c(2\alpha - 1)m$ and it must be the upper bound of βm . Then

$$c(2\alpha - 1)m \geq \beta m \;\; \Rightarrow \;\; c \geq \frac{\beta}{2\alpha - 1}$$

When inserting a node w into α -balanced tree, it only changes $\Delta(i)$ for all i in the path from the root to w since inserting x doesn't change the size of subtrees of other nodes. And $\Delta(i)$ changes at most 1 because one of $S(l(i))$ and $S(r(i))$ increases by one. By (2), the length of this path is at most the height of the tree, so the potential increases by at most $c\mathcal{O}(\log N) = \mathcal{O}(\log N)$.

Let Φ be the potential difference between two states (before inserting and after rebuilding). If inserting w leads to an unbalanced tree and using the variable c mentioned above so that the potential difference can offset the cost of rebuilding, we have

$$\begin{aligned} & \hat{\mathrm{k}} \\ = \; & \mathrm{k} + \Phi \\ = \; & (\text{insert cost} + \text{rebuild cost}) + \Phi \\ = \; & \text{insert cost} + (\text{rebuild cost} + \Phi) \\ = \; & \mathcal{O}(\log N) + \mathcal{O}(\log N) \\ = \; & \mathcal{O}(\log N) \end{aligned}$$

(insert cost $= \mathcal{O}(\log N)$ since it takes $\mathcal{O}(height)$ to insert a node in BST.)

If inserting w doesn't lead to an unbalanced tree, it is clear that

$$\hat{\mathrm{k}} = \mathcal{O}(\log N)$$

Thus the amortized cost per insertion $\hat{k} = \mathcal{O}(\log N)$.

(4*) First take an inorder traversal of the BST and put a node into the array. It takes $\mathcal{O}(m)$ to do this by solving the recurrence $T(m) = 2T(\frac{m}{2}) + \mathcal{O}(1)$. This would generate a sorted array by the property of BST. Then the algorithm is shown below

```
// sorted array A
Rebuild(A, start, size){
    if size > 0
        root = new Node;
        root.key = A[start + size/2]
        root.left = Rebuild(A, start, size/2)
        root.right = Rebuild(A, start + size/2 + 1, size - size/2 - 1)
        return root
    else
        return  NULL
}
```

The tree constructed by `Rebuild()` is BST since A is sorted. And for each node i , we have

$$S(l(i)) = \lfloor S(i)/2 \rfloor \leq S(i)/2 \;\text{ and }\; S(r(i)) = S(i) - \lfloor S(i)/2 \rfloor - 1 \leq S(i)/2$$

which satisfies the condition of $1/2$ -balanced tree.

The complexity of `Rebuild(A, 0, m)` is \mathcal{O}(m) by solving the recurrence $T(m) = 2T(\frac{m}{2}) + \mathcal{O}(1)$. Thus the total complexity is $\mathcal{O}(m)$.

Problem 2

(1) Induction on k . Suppose there is a 3 -CNF A such that

$$A = C_1 \wedge C_2 \wedge \cdots \wedge C_m, \text{ where } C_i = (x_1 \vee x_2 \vee x_3)$$

When $k = 4$, replace each C_i with $(x_4 \vee C_i) \wedge (\neg x_4 \vee C_i)$. Then we have

$$A' = (x_4 \vee C_1) \wedge (\neg x_4 \vee C_1) \wedge (x_4 \vee C_2) \wedge (\neg x_4 \vee C_2) \cdots \wedge (x_4 \vee C_m) \wedge (\neg x_4 \vee C_m)$$

where A' is a 4 -CNF formula witn $2m$ clauses. A is satisfiable if and only if A' is satisfiable since C_i is satisfiable if and only if $(x_4 \vee C_i) \wedge (\neg x_4 \vee C_i)$ is satisfiable. It takes $\mathcal{O}(m)$ to reduce A to A' . Thus it holds when $k = 4$.

Suppose it holds when $k = j$. That is, A can be reduced to a j -CNF A_j in polynomial time by keeping replacing C_i with $(y \vee C_i) \wedge (\neg y \vee C_i)$, where

$$A_j = B_1 \wedge B_2 \wedge \cdots \wedge B_n, \text{ where } B_i = (x_1 \vee x_2 \vee \cdots \vee x_j)$$

and $n = 2^{j-3}m$. Replace each B_i with $(x_{j+1} \vee B_i) \wedge (\neg x_{j+1} \vee B_i)$. Then we have

$$A_{j+1} = (x_{j+1} \vee B_1) \wedge (\neg x_{j+1} \vee B_1) \wedge (x_{j+1} \vee B_2) \wedge (\neg x_{j+1} \vee B_2) \cdots \wedge (x_{j+1} \vee B_n) \wedge (\neg x_{j+1} \vee B_n)$$

where A_{j+1} is a $(j+1)$ -CNF formula with $2n$ clauses. A_j is satisfiable if and only if A_{j+1} is satisfiable since B_i is satisfiable if and only if $(x_{j+1} \vee B_i) \wedge (\neg x_{j+1} \vee B_i)$ is satisfiable. It takes $\mathcal{O}(2n) = \mathcal{O}(2^{j-2}m) = \mathcal{O}(m)$ to run it since j is a constant. By induction hypothesis, A can be reduced to A_j in polynomial time and since A_j can be reduced to A_{j+1} in polynomial time, then A can be reduced to A_{j+1} in polynomial time since polynomial is closed under addition. Thus the problem is proved.

(2) Let A_k be k -CNF with

$$A_k = C_1 \wedge C_2 \wedge \cdots \wedge C_m, \text{ where } C_i = (x_1 \vee x_2 \vee \cdots \vee x_k)$$

Then partition $x_1 \vee x_2 \vee \cdots \vee x_k$ in each C_i into

$$(T_1 \vee T_2) \vee (T_3 \vee T_4) \cdots \vee (T_{\frac{k}{4}-3} \vee T_{\frac{k}{4}-2}) \vee (T_{\frac{k}{4}-1} \vee T_{\frac{k}{4}-2}) \vee S_1 \vee S_2 \cdots \vee S_{k \ mod \ 4}$$

where $T_j = (x_{2j-1} \vee x_{2j})$ and $S_j = x_{k-k \ mod \ 4+j}$. By hints,

$$T_{2j-1} \vee T_{2j} \Leftrightarrow (p_j \vee T_{2j-1}) \wedge (\neg p_j \vee T_{2j}) \quad and$$
$$S_j \Leftrightarrow (q_j \vee S_j) \wedge (\neg q_j \vee S_j) \Leftrightarrow (r_j \vee q_j \vee S_j) \wedge (\neg r_j \vee q_j \vee S_j) \wedge (w_j \vee \neg q_j \vee S_j) \wedge (\neg w_j \vee \neg q_j \vee S_j)$$

Replace each T_j and S_j with symbols mentioned above. By hints, the old one is satisfiable if and only if the new one is satisfiable. Since there are three literals in each pair of clauses and it takes $\mathcal{O}(m)$ to run it, the problem is proved.

(3)
Reference:
http://cse.vnit.ac.in/people/narendraschaudhari/wp-content/uploads/sites/15/2014/10/039-3-SAT-to-K-Colorability-ooc1005.pdf
http://cgi.csc.liv.ac.uk/~igor/COMP309/3CP.pdf
Let $G = \{V, E\}$ where

$$V = \{v_1, v_2, v_3 \dots v_n\}$$

where v_i is an island and

$$E = \{(v_i, v_j) : \text{the bridge between island } i \text{ and island } j\}$$

Suppose there are three colors red , blue and green and each color represents exactly one son. The island v_i painted with the color means that it is owned by the son corresponding to this color. Since the two islands v_i, v_j connected by each bridge (v_i, v_j) cannot be assigned to the same son, v_i and v_j cannot be colored with the same color.
Hence, it is now a graph 3 -colorability problem. To show the problem is NPC, first I prove that it is NP and then prove it is NP-hard.
Prove it is NP:
Given a graph $G = \{V, E\}$ and a coloring assignment of the vertices. Traverse the graph and check whether all adjacent vertices have a different color and check that there are only there colors. Since the father want to “evenly” split the islands to each son, it needs to make sure that each color appears the same times. Totally, it takes $\mathcal{O}(|V| + |E|)$ implying graph 3 -colorability is NP.
Prove it is NP-hard:
To prove it, I will show that 3 -CNF can be reduced to graph 3 -colorability.
Given 3 -CNF formula F such that

$$F = C_1 \wedge C_2 \wedge \dots C_m, \text{ and } C_i = x_1 \vee x_2 \vee x_3$$

with the set of variables $X = \{x_1, x_2 \dots x_n\}$ and m pairs of clauses. Construct a graph G as follows.
The graph $G = \{V, E\}$ consists of two triangles, one is the triangle for each variable and the other is the triangle for each clause in F .
The triangle for each variable consists of x_i , $\neg x_i$ and Base vertex B where $x_i \in X$. Since each $\{x_i, \neg x_i\}$ shares the same Base vertex B , it can have two truth assignment, which are $\{true, false\}$ and $\{false, true\}$ so that the triangle is colorable.
As for the triangle for each clause, connect each of vertex in it to the corresponding literal vertex. Thus, such edge forces its two endpoints to be colored with different colors.
Correctness:
The triangle for each clause can is colorable if and only if not all its corresponding vertices are colored with the same color, that is, not all literals in the clause are assigned the same boolean value.
The reduction algorithm lead to graph $G = \{V, E\}$ such that

$$|V| = 2n + 3m + 1 \text{ and } |E| = 3n + 6m$$

implying the algorithm yields to polynomial running time. Thus 3 -CNF SAT $\leq_P$ graph 3 -colorability.
Since 3 -CNF SAT is in NPC and graph 3 -colorability is in NP, 3 -colorability is in NPC, and so is this problem.

(4)
Reference: https://en.wikipedia.org/wiki/Clique_cover_problem
Since the islands belonging to the same son should all be connected by one bridge to one another, they forms a clique,

that is, every two of the islands owned by the same son are adjacent. Then the problem is a 3 clique cover problem which means that whether all islands can be partitioned into 3 cliques. To prove 3 clique cover problem is in NPC, I will show that 3 clique cover problem is NP and graph 3 -colorability, which is in NPC, can reduce to 3 clique cover problem.

Prove 3 clique cover problem is in NP:

Once given a partition of the vertices into 3 sets, it takes polynomial time to check whether each set forms a clique. Hence, it is in NP.

Prove graph 3 -colorability can reduce to 3 clique cover problem

More definitions:

Definition: A clique in a graph is a set of vertices every two of which are adjacent.

Definition: An independent set in a graph is a set of vertices no two of which are adjacent.

Lemma 1: Suppose that $G = \{V, E\}$ is an undirected graph and V' is a subset of V . Then

$$V' \text{ is a clique of } G \text{ if and only of } V' \text{ is an independent set of } \bar{G}$$

where $\bar{G}$ is the complement of G .

Prove:

$$V' \text{ is a clique of } G \Leftrightarrow (x, y) \in G \Leftrightarrow (x, y) \notin \bar{G} \Leftrightarrow V' \text{ is an independent set of } \bar{G}$$

where x, y are any two vertices in V' .

Now given the instance $G = \{V, E\}$ of graph 3 -colorability. First transform G into its complement $\bar{G}$. By Lemma 1, a partition of $\bar{G}$ into 3 cliques is equivalent to a partition of G into 3 independent sets. Then color each set with corresponding color which yields to graph 3 -colorability. Thus 3 -colorability of G could be solved by a single call to 3 clique cover of $\bar{G}$ in addition to polynomial time of constructing complement graph which implying 3 clique cover could be reduced from 3 -colorability.

Since 3 clique cover is in NP and 3 -colorability $\leq_P$ 3 clique cover, 3 clique cover is in NPC, and so is this problem.

Problem 3

(1)

Reference:

http://arxiv.org/pdf/1208.0142.pdf

http://www.cs.uu.nl/docs/vakken/an/graphisomorphism_handout.pdf

http://www.math.ucsd.edu/~minnes/archive/fall2012/154/files/154fa12_hw1_sol.pdf

http://web.stanford.edu/class/cs103x/2007/solutions8.pdf

Prove that determining two simple connected graphs are isomorphic is in GI-complete

Let this problem be X .

(a) **X is reducible to graph isomorphism problem**

It holds trivially since in the original graph isomorphism problem, the graph need not to be connected.

(b) **Graph isomorphism problem is reduciable to X**

First prove two lemmas.

Lemma 1: Two graphs G and H are isomorphic if and only if their complement graphs $\bar{G}$ and $\bar{H}$ are isomorphic.

Prove: Given G and H are two graphs. G and H are isomorphic if and only if is a bijective function $f : V(G) \to V(H)$ such that if $(u, v) \in E(G)$, then $(f(u), f(v)) \in E(H)$. Recall complement graph consists of the same verices of the original graph but consists of all edges that are not in the original graph. Thus

$$(x, y) \in E(\bar{G}) \quad \Leftrightarrow \quad (x, y) \notin E(G) \quad \Leftrightarrow \quad (f(x), f(y)) \notin E(H) \quad \Leftrightarrow \quad (f(x), f(y)) \in E(\bar{H})$$

implying G and H are isomorphic if and only if $\bar{G}$ and $\bar{H}$ are isomorphic.

Lemma 2: Either G or its complement graph $\bar{G}$ is connected.

Prove: If G is connected, it holds trivially. Suppose G is disconnected. Let u and v are two vertices in G . If u and v are in different connected components of G , the there is no edge between u and v . Thus the edge (u, v) in $\bar{G}$. If u and v are in the same connected component of G , since G is disconnected, there exists a vertex w in another different connected component. By the first condiction, the edges (u, w) and (v, w) are both in $\bar{G}$. Then the path $u \to w \to v$ exists in $\bar{G}$ implying u and v are connected in $\bar{G}$. Hence any two vertices in $\bar{G}$ are connected and $\bar{G}$ is connected.
Now back to original problem. For a simple graph, it takes polynomial time to construct its complement graph. Given two graphs G and H . If both G and H is connected, it holds trivially since it is exactly problem X . If both G and H are disconnected, first take polynomial time to find complement graphs $\bar{G}$ and $\bar{H}$. By Lemma 2, both $\bar{G}$ and $\bar{H}$ are connected. Apply the algorithm solving X to solve whether $\bar{G}$ and $\bar{H}$ are isomorphic. By Lemma 2, G and H are isomorphic if and only if $\bar{G}$ and $\bar{H}$ are isomorphic. Hence, graph isomorphism problem is reduciable to X .
By (a) and (b), determining two simple connected graphs are isomorphic is in GI-complete.

(2)
Reference: https://cs.uwaterloo.ca/research/tr/1977/CS-77-04.pdf
Prove that giving a bijective f if the two graphs are isomorphic is in GI-complete
Let this problem be X .
(a) **Graph isomorphism problem is reduciable to X**
It's clear that graph isomorphic problem can be solved by one call to X . Thus, Graph isomorphism problem is reduciable to X
(b) **X is reduciable to graph isomorphism problem**
Given two graphs G and H , and both of them with n vertices. If they are not isomorphic, there is not such bijective f . If they are isomorphic, select a vertex $x \in G$ and label v uniquely. The graph after selecting x is denoted as G_x . Search a vertex $a \in H$ such that G_x and H_a are isomorphic. This vertex exists certainly since G and H are isomorphic. Then select a new vertex $y \in G$ and search a vertex $b \in H$ such that $G_{x,y}$ and $H_{a,b}$ are isomorphic. Iteratively select and search until all vertices are labeled. Then f can be found since it maps the $i-th$ labeled vertex in G to the $i-th$ labeled vertex in H . This method calls $\mathcal{O}(n^2)$ times graph isomorphism algorithm to solve X and some additional polynomial time is needed to search a corresponding vertex in H . Thus, X is turing reduciable to graph isomorphism problem.
By (a) and (b), giving a bijective f if the two graphs are isomorphic is in GI-complete.

(3)
Reference: https://cs.uwaterloo.ca/research/tr/1977/CS-77-04.pdf
Prove that to determine two regular graphs are isomorphic is in GI-complete
Let this problem be X .
(a) **X is reduciable to graph isomorphism problem**
It holds trivially since in the original graph isomorphism problem, the graph need not to be regular.

(b) **Graph isomorphism problem is reduciable to X**
Define a graph $R(d)$ such that $R(d)$ has all vertices with degree d except two vertices s and t with degree $d-1$. s and t must have the same adjacent vertices.
I will show how to construct a regular graph which is unique to the original graph.
Given a graph G with maximum degree d . Take two copies of G , denoted by G_1 and G_2 . Each vertex $v \in G$ corresponds to a vertex $v_1 \in G_1$ and a vertex $v_2 \in G_2$. For each vertex $v \in G$, take $d - deg(v)$ copies of $R(d)$ and build the egdes (v_1, s) and (v_2, t) in each copy. The result graph G' is a d -regular graph and is uniquely decided by G .
Lemma 1: Given two graphs G and H . G and H are isomorphic if and only if G' and H' are isomorphic.
Prove: Straightforward since G' (H') is uniquely decided by G (H).
Given two graphs G and H . If G and H are both regular graphs, it holds trivially. If not, take polynomial time to construct G' and H' . By Lemma 1, call X to solve whether G' and H' are isomorphic and the answer is the same to whether G and H are isomorphic. Thus, graph isomorphism problem is reduciable to X .
By (a) and (b), to determine two regular graphs are isomorphic is in GI-complete.