Problem Set No. 4 – Re-Corrected
Collaboration allowed on all problems

1. A strong red-black tree is a red-block tree such that each red node not only has a black parent but also a black sibling. Describe in detail insertion and deletion algorithms for strong red-black trees that run in $O(\log n)$ time on an n-node tree.

2. Consider the following three stack operations: push, which pushes symbol $\alpha$ on the stack, top, which writes the top symbol on the stack, and pop, which pops the top element on the stack. Given a sequence of symbols $\alpha_1, \alpha_2, \ldots, \alpha_n$, we will determine a minimum number of stack operations that wish to write $\alpha_1, \alpha_2, \ldots, \alpha_n$ (in order), starting and ending with an empty stack.

   (a) Prove that the minimum number of stack operations needed is $C(1,n)$, where $C(i,j)$ for $1 \leq i \leq j \leq n$ is defined by the following recurrence:

   $$C(i,j) = \begin{cases} 
   3 & \text{if } i=j \\
   \min \{ \text{if } \alpha_i=\alpha_j \text{ then } 1+C(i+1,j) \text{ else } 3+C(i+1,j), \\
   \min \{ 1+C(i+1,k)+C(k+1,j) \mid i<k<j \text{ and } \alpha_k=\alpha_k \} \} 
   \end{cases}$$

   (b) Show how to compute $C(1,n)$ in $O(n^3)$ time.

3(a) Show that every strongly connected graph $G$ is either a single vertex or can be represented, for some $k \geq 1$, as a sequence of graphs $G_1, G_2, \ldots, G_k$ and a sequence of arcs $(v_{10}, v_{2i}), (v_{20}, v_{3i}), \ldots, (v_{ki}, v_{1i})$ such that:

   (i) the vertices of $G_1, \ldots, G_k$ partition the vertices of $G$;
   (ii) the union of the arcs of $G_1, \ldots, G_k$ and $(v_{10}, v_{2i}), \ldots, (v_{ki}, v_{1i})$ is the set of arcs of $G$;
   (iii) each arc $(v_{ai}, v_{bi})$ has $v_{ai}$ a vertex in $G_a$ and $v_{bi}$ a vertex in $G_b$;
   (iv) each $G_i$ is either strongly connected or consists of just a single vertex.

   Note: for $k = 1$, there is one arc $(v_{10}, v_{1i})$. Both $v_{ai}$ and $v_{ao}$ belong to graph $G_a$, they may or may not be different..

   (b) By applying the decomposition in part (a) recursively, we can break any strongly connected graph up into a nested decomposition of “cycles inside cycles”. Design an efficient algorithm (one as fast as possible) to compute such a nested decomposition. Include a description of the data structure your algorithm uses to
represent the decomposition. What is the worst-case running time of your algorithm?