# CS 473: Undergraduate Algorithms, Fall 2013
# Homework 0

Due Tuesday, September 3, 2013 at 12:30pm

---

Quiz 0 (on the course Moodle page)
is also due Tuesday, September 3, 2013 at noon.

---

- **Please carefully read the course policies on the course web site.** These policies may be different than other classes you have taken. (For example: No late anything ever; "I don't know" is worth 25%, but "Repeat this for all $n$" is an automatic zero; ***every*** homework question requires a proof; collaboration is allowed, but you must cite your collaborators.) If you have *any* questions, please ask in lecture, in headbanging, on Piazza, in office hours, or by email.

- Homework 0 and Quiz 0 test your familiarity with prerequisite material—big-Oh notation, elementary algorithms and data structures, recurrences, graphs, and most importantly, induction—to help you identify gaps in your background knowledge. **You are responsible for filling those gaps.** The course web page has pointers to several excellent online resources for prerequisite material. If you need help, please ask in headbanging, on Piazza, in office hours, or by email.

- ***Each student must submit individual solutions for these homework problems.*** You may use any source at your disposal—paper, electronic, or human—but you ***must*** cite ***every*** source that you use. For all future homeworks, groups of up to three students may submit joint solutions.

- **Submit your solutions on standard printer/copier paper, not notebook paper.** If you write your solutions by hand, please use the last three pages of this homework as a template. At the top of each page, please clearly print your name and NetID, and indicate your registered discussion section. Use both sides of the page. If you plan to typeset your homework, you can find a LaTeX template on the course web site; well-typeset homework will get a small amount of extra credit.

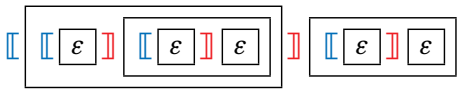
- Submit your solution to each numbered problem (stapled if necessary) in the corresponding drop box outside 1404 Siebel, ***or*** in the corresponding box in Siebel 1404 immediately before/after class. (This is the last homework we'll collect in 1404.) ***Do not staple your entire homework together.***

---

1. Consider the following recursively-defined sets of strings of left brackets ⟦ and right brackets ⟧:

   - A string $x$ is ***balanced*** if it satisfies one of the following conditions:
     - $x$ is the empty string, or
     - $x = ⟦\, y \,⟧\, z$, where $y$ and $z$ are balanced strings.

     For example, the following diagram shows that the string ⟦ ⟦ ⟧ ⟦ ⟧ ⟧ ⟦ ⟧ is balanced. Each boxed substring is balanced, and $\varepsilon$ is the empty string.

   - A string $x$ is ***erasable*** if it satisfies one of two conditions:
     - $x$ is the empty string, or
     - $x = y ⟦\, ⟧\, z$, where $yz$ is an erasable string.

     For example, we can prove that the string ⟦ ⟦ ⟧ ⟦ ⟧ ⟧ ⟦ ⟧ is erasable as follows:

     $$⟦\,⟦\,⟧\,\cancel{⟦\,⟧}\,⟧\,⟦\,⟧ \;\to\; ⟦\,\cancel{⟦\,⟧}\,⟧\,⟦\,⟧ \;\to\; ⟦\,⟧\,\cancel{⟦\,⟧} \;\to\; \cancel{⟦\,⟧} \;\to\; \varepsilon$$

   Your task is to prove that these two definitions are equivalent.

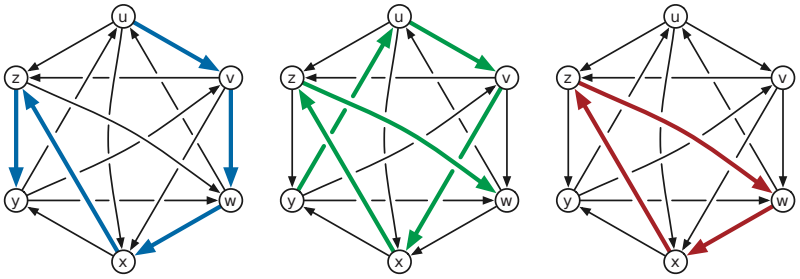
   (a) Prove that every balanced string is erasable.

   (b) Prove that every erasable string is balanced.

2. A ***tournament*** is a directed graph with exactly one directed edge between each pair of vertices. That is, for any vertices $v$ and $w$, a tournament contains either an edge $v \to w$ or an edge $w \to v$, but not both. A ***Hamiltonian path*** in a directed graph $G$ is a directed path that visits every vertex of $G$ exactly once.

   (a) Prove that every tournament contains a Hamiltonian path.

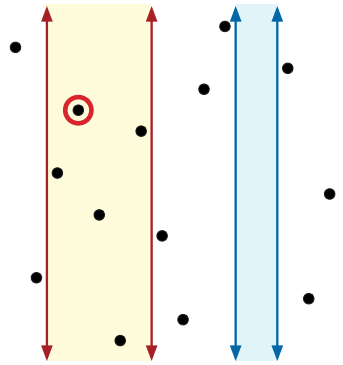
   (b) Prove that every tournament contains either *exactly one* Hamiltonian path or a directed cycle of length three.




A tournament with two Hamiltonian paths $u \to v \to w \to x \to z \to y$ and $y \to u \to v \to x \to z \to w$ and a directed triangle $w \to x \to z \to w$.

3. Suppose you are given a set $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ of $n$ points in the plane with distinct $x$- and $y$-coordinates. Describe a data structure that can answer the following query as quickly as possible:

   > Given two numbers $l$ and $r$, find the highest point in $P$ inside the vertical slab $l < x < r$. More formally, find the point $(x_i, y_i) \in P$ such that $l < x_i < r$ and $y_i$ is as large as possible. Return NONE if the slab does not contain any points in $P$.



A query with the left slab returns the indicated point.
A query with the right slab returns NONE.

To receive full credit, your solution must include (a) a concise description of your data structure, (b) a concise description of your query algorithm, (c) a proof that your query algorithm is correct, (d) a bound on the size of your data structure, and (e) a bound on the running time of your query algorithm. You do ***not*** need to describe or analyze an algorithm to construct your data structure.

Smaller data structures and faster query times are worth more points.