Algorithms – CMSC-27200
http://alg15.cs.uchicago.edu
Homework set #4
Probems due February 4 except where stated otherwise

Read the homework instructions on the website. The instructions that follow here are only an incomplete summary.

Hand in your solutions to problems marked "HW." Do not hand in problems marked "DO." If you hand in homework marked **"XC" (extra credit)**, do so on **separate and separately stapled sheets, please**. PRINT YOUR NAME and SECTION NUMBER ON EVERY SHEET you submit. We ask you to **use LaTeX to typeset your solutions**. Because of the late posting, this is not mandatory (but preferred) for this homework. It will again be required starting with the next homework (due Feb 11). Hand in your solutions on paper, do not email.

When writing pseudocode, **explain the meaning of your variables.** Use comments to explain what is happening in each line. Also, give a short explanation of the idea behind the algorithm. **Describe all algorithms in this problem set in pseudocode. Elegance of your code matters.**

Carefully study the policy (stated on the website) on collaboration, internet use, and academic integrity.

In this problem set, every digraph is given in an *adjacency-list* representation, unless expressly stated otherwise. Recall that an adjacency-list representation consists of an array of vertices, where vertex u is the start of the linked list Adj$[u]$ that lists all the out-neighbors of u in some order. An algorithm on a digraph $G = (V, E)$ is said to run in ***linear time*** if the number of steps is $O(|V| + |E|)$.
By "graph" we mean an undirected graph without self-loops and without parallel edges. A graph is a digraph satisfying the condition that for all $u, v \in V$, if $(u, v) \in E$ then $(v, u) \in E$ and $(u, u) \notin E$.

4.1 **(Little-oh notation)** Let a_n, b_n be sequences of real numbers, $b_n \neq 0$. Recall that we say that $a_n = o(b_n)$ ("a_n is little-oh b_n") if $\lim_{n\to\infty} a_n/b_n = 0$. In this case we also say that a_n is of *lower order of magnitude* than b_n.

(a) DO: Consider the following statements:
(A) $a_n = o(b_n)$ and (B) $a_n = O(b_n)$.
Prove: (a1) (A) $\Longrightarrow$ (B) but (a2) (B) $\not\Longrightarrow$ (A).

(b) DO: Prove: if $a_n = o(b_n)$ then eventually (i. e., for all sufficiently large n), $|a_n| < |b_n|$.

(c) **HW (4+4 points)** Assume $a_n, b_n > 1$. Consider the following two statements.
(C) $a_n = o(b_n)$ and (D) $\ln a_n = o(\ln b_n)$.
Prove: (c1) (C) $\not\Rightarrow$ (D) and (c2) (D) $\not\Rightarrow$ (C).

4.2 **HW (5 points)** Part (c) of the previous exercise is a strong caveat if trying to use logarithms to establish little-oh relations. Nevertheless, with some care, logarithms can sometimes be used to establish little-oh relations. Specifically, prove that

$$2^{(\lg n)^{1/2}} = o(n^{1/10}). \tag{1}$$

Hint. Use logarithms to prove that $2^{(\lg n)^{1/2}}$ is eventually less than $n^{1/11}$. (Use part (b) of the preceding exercise.) Then observe that $n^{1/11} = o(n^{1/10})$ and put these two facts together.

4.3 DO: Prove the following asymptotic equality:

$$\frac{4^n}{\binom{2n}{n}} \sim c\sqrt{n} \tag{2}$$

for some constant c. Determine c.

4.4 **HW, due February 11 (20 points)** Let $G = (V, E, w)$ be an edge-weighted DAG. Solve the single-source min-cost path problem in linear time. (What you need to do is, given a source vertex $s \in V$, determine for each $v \in V$ the cost of a min-cost path from s to t and find a tree of such min-cost paths, represented by parent links.) Note: you cannot use priority queues because we don't know how to implement them in linear time. In Dijkstra's algorithm, a priority queue is used to determine the order in which the vertices are finalized (colored "black"). What alternative ordering tool do we have in the case of DAGs? Prove that your algorithm is (a) correct and (b) runs in linear time. ("Linear time" means total cost $(|V| + |E|)$, where arithmetic with reals and bookkeeping operations are performed at unit cost.) Note: negative weights ARE allowed in this problem.

Your algorithm will run in two phases. The first phase will sort the vertices. This will be done by a routine we have studied; just name the routine. The second phase is a modification of Dijkstra's algorithm;

describe this phase in **pseudocode**. Explain your variables. State the loop-invariant needed to prove correctness. (Refer to the handout on loop-invariants to be posted later this weekend.) Indicate why this second phase completes in linear time; state the number of arithmetic and comparison operations for real numbers will be performed.

4.5 **HW (6 points)** Solve the single-source **max-cost** path problem for edge-weighted DAGs in linear time. (This is a problem of practical significance since DAGs model dependencies among subtasks in complex projects.) State the computational task; be accurate in describing the input and the output. You may refer to the preceding exercise without proof; no pseudocode needed.

4.6 DO: The midterm (posted) states the "Uphill-downhill Dijkstra problem" and asks to solve it in "Dijkstra time." Show you can do better: solve this problem in linear time.

4.7 DO: A "partial permutation" of k distinct items is a permutation of a subset of those items. Prove: the number of partial permutations of k distinct items is $O(k!)$. In fact, show that this number is $< \mathrm{e}k!$.

4.8 DO **(Dijkstra with some negative edges)** Let $G = (V, E, w)$ be an edge-weighted digraph with source vertex s. By "Dijkstra time" we mean our best upper bound on the running time of Dijkstra's algorithm in terms of $|V|$ and $|E|$. Solve the single-source min-cost path problem for (G, s)

   (a) in big-Oh of Dijkstra time assuming there is at most one negative edge

   (b) in $k!$ times big-Oh of Dijkstra time assuming there are at most k negative edges and no negative cycles.