"CS 374" Fall 2015 — Homework 6

Due Tuesday, October 20, 2015 at 10am

••• Some important course policies •••

- **You may work in groups of up to three people.** However, each problem should be submitted by exactly one person, and the beginning of the homework should clearly state the names and NetIDs of each person contributing.
- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use. See the academic integrity policies on the course web site for more details.
- **Submit your pdf solutions in Moodle.** See instructions on the course website and submit a separate pdf for each problem. Ideally, your solutions should be typeset in LaTex. If you hand write your homework make sure that the pdf scan is easy to read. Illegible scans will receive no points.
- **Avoid the Three Deadly Sins!** There are a few dangerous writing (and thinking) habits that will trigger an automatic zero on any homework or exam problem. Yes, we are completely serious.
  - Give complete solutions, not just examples.
  - Declare all your variables.
  - Never use weak induction.
- Unlike previous editions of this and other theory courses we are not using the "I don't know" policy.

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

---

1. We have seen some variants of the maximum weight subset problem under constraints during the lecture on greedy algorithms. Consider the following problem which generalizes the two variants we saw. The input is a rooted tree $T = (V, E)$. Each edge $e \in E$ has a non-negative integer capacity $u(e)$. For a given $e \in E$ let T_e be the sub-tree of T under the edge e. The items of interest are the leaves L of T. Each leaf v has a non-negative weight $w(v)$. The goal is to find a maximum weight subset $S \subseteq L$ such that the following holds: for each edge e the number of leaves in S that are in the sub-tree T_e is at most $u(e)$; formally $|S \cap L(T_e)| \leq u(e)$ for each $e \in E$ where $L(T_e)$ is the set of leaves of T_e. Describe a greedy algorithm for this problem and prove its correctness.

2. A long straight country road can be modeled as a line starting at 0. The road has n houses at locations $x_1, x_2, \ldots, x_n$ on the line. The city wants to build fire stations along the road such that every house is within distance D from some fire station. Fire stations cannot be built at arbitrary locations. The city has figured m potential locations on the road at $y_1, y_2, \ldots, y_m$. For simplicity assume that all the x and y values are distinct.

   - Describe a greedy algorithm that finds the minimum number of fire stations that the city can build to satisfy the desired constraint that every house is within distance D from a fire station. Your algorithm has to detect if a feasible solution does not exist. Briefly justify the correctness of your algorithm by aruging why there is an optimum solution that agrees with the first choice of your greedy algorithm.
   - The city has realized subsequently that not all locations are equal in terms of the cost of building a fire station. Asssume that c_j is the cost of building fire station at location y_j. Describe a dynamic programming algorithm that minimizes the total cost of building the fire stations under the same constraint as before. Do you see why a greedy algorithm may not work for this cost version?

3. Graphs are a powerful tool to model many phenomena. The edges of a graph model pairwise relationships. It is natural to consider higher order relationships. Indeed *hypergraphs* provide one such modeling tool. A hypergraph $G = (V, \mathcal{E})$ consists of a finite set of nodes/vertices V and a finite set of hyper-edges $\mathcal{E}$. A hyperedge $e \in \mathcal{E}$ is simply a subset of nodes and the cardinality of the subset can be larger than two. An undirected graph is a hypergraph where each hyper-edge is of size exactly two. Here is an example. $V = \{1, 2, 3, 4, 5\}$ and $\mathcal{E} = \{\{1, 2\}, \{2, 3, 4\}, \{1, 3, 4, 5\}, \{2, 5\}\}$. The representation size of a hypergraph is $|V| + \sum_{e \in \mathcal{E}} |e|$. An alternating sequence of nodes and edges $x_1, e_1, x_2, e_2, \ldots, e_{k-1}, x_k$ where $x_i \in V$ for $1 \leq i \leq k$ and $e_j \in \mathcal{E}$ for $1 \leq j \leq k-1$ is called a path from u to v if (i) $x_1 = u$ and $x_k = v$ and (ii) for $1 \leq j < k$, $x_j \in e_j$ and $x_{j+1} \in e_j$.

   - Given a hypergraph $G = (V, \mathcal{E})$ and two nodes $u, v \in V$ we say that u is connected to v if there is path from u to v. We say that a hypergraph is connected if each pair of nodes u, v in G are connected. Describe an algorithm that given a hypergraph G checks whether G is connected in linear time. In essense describe a reduction of this problem to the standard graph connectivity problem. You need to prove the correctness of your algorithm.

- Suppose we want to quickly spread a message from one person to another person during an emergency on a social network called AppsWhat which is organized as a collection of groups. Messages sent by a group member are broadcast to the entire group. AppsWhat knows the members and the list of groups on its service. The goal is to find the fewest messages that need to be sent such that a person u can reach a person v. Model this problem using hypergraphs and describe a linear-time algorithm for it. You can assume that BFS solves the shortest path problem in a regular unweighted graph in time that is linear in the graph's size. No proof necessary for this part.