Problem 19

In this problem, you will analyze an algorithm to maintain a linked list of $n$ elements, using competitive analysis. Requests to the algorithm are of the form \textsc{Access}(x), where $x$ is an item in the list. In response to this request, the algorithm walks the list starting from the first position until the element $x$ is encountered. The algorithm is also allowed to modify the list as it proceeds. In particular, in servicing an \textsc{Access} request, the accessed element may be moved to any earlier position in the list. We think of the input to the algorithm as the request sequence $\sigma = (\sigma_1, \ldots, \sigma_m)$, where $\sigma_k$ is the element requested by the $k$th \textsc{Access} request.

In order to analyze this problem under the lens of competitive analysis, we need to be precise about the costs of various operations, so here is a model for the costs:

1. \textsc{Access}(x) where $x$ is currently at position $k$ in the list costs $k$. (This models the cost of traversing the linked list until $x$ is encountered). Position numbers start from 1.

2. Moving the accessed element to any earlier position is free.

*Thanks to Tim Roughgarden for permission to include problems from his Winter 2016 edition of CS 261
We will analyze a simple strategy Move-To-Front (MTF): After Access\((x)\), move \(x\) all the way to the front of the list.

It turns out that computing the optimal strategy for List Update is NP-complete. Given that we don’t know how to compute the optimal strategy efficiently, it is quite remarkable that we can compare the performance of MTF to that of the optimal strategy to show that it is 2-competitive. Henceforth, when we refer to the optimal algorithm, we mean some strategy that achieves minimum cost for the request sequence. If there are multiple such strategies, pick one arbitrarily. Note that the optimal strategy is a function of the entire request sequence. Both MTF and the optimal algorithm start with the linked list in the same configuration.

(a) Let \(C_{MTF}(\sigma_k)\) denote the cost of MTF on the \(k\)th request and let \(C_{OPT}(\sigma_k)\) denote the cost of the optimal algorithm on the \(k\)th request. \(C_{MTF}(\sigma) = \sum_k C_{MTF}(\sigma_k)\) and \(C_{OPT}(\sigma) = \sum_k C_{OPT}(\sigma_k)\) One natural approach would be to prove that for every \(k\), \(C_{MTF}(\sigma_k) \leq 2C_{OPT}(\sigma_k)\). Show that such a strong statement is not true.

(b) The analysis involves a potential function argument that is quite commonly used in online algorithms. We imagine running MTF and OPT side by side on the request sequence. We will define a non-negative function \(\Phi\) that we call a potential function. Let \(\Phi_k\) be the value of the potential function after \(k\) requests have been serviced. The initial value is \(\Phi_0 = 0\). We will show that

\[
\underbrace{C_{MTF}(\sigma_k) + \Phi_k - \Phi_{k-1}}_{\text{amortized cost}} \leq 2 \cdot C_{OPT}(\sigma_k) \tag{1}
\]

The LHS of the above inequality is referred to as the amortized cost. The potential function \(\Phi\) helps smooth out the costs of the algorithm over various steps so as to facilitate a comparison with the cost of OPT. Show that (1) implies that \(C_{MTF}(\sigma) \leq 2 \cdot C_{OPT}(\sigma)\).

(c) Consider the following potential function: \(\Phi\) is equal to the number of inverted pairs \(\{x,y\}\) in the two lists currently maintained by MTF and OPT. \(\{x,y\}\) is an inverted pair if \(x\) occurs before \(y\) in one list, but \(y\) appears before \(x\) in the other list. Verify that \(\Phi\) satisfies the conditions \(\Phi \geq 0\), \(\Phi_0 = 0\), and prove that (1) holds. This establishes that MTF is 2-competitive.

*Hint:* Define

\[
S = \{y| y \text{ precedes } \sigma_k \text{ in MTF’s list and in OPT’s list}\}
\]

\[
T = \{y| y \text{ precedes } \sigma_k \text{ in MTF’s list but not in OPT’s list}\}
\]

Now obtain bounds on \(C_{MTF}(\sigma_k)\) and \(C_{OPT}(\sigma_k)\) in terms of \(S\) and \(T\).

(d) Let \(A\) be any deterministic algorithm for the problem. For any \(\epsilon > 0\), show that the competitive ratio of \(A\) is at least \(\frac{2n}{n+1} - \epsilon\). *Hint:* Construct a long request sequence that makes \(A\) incur very high cost. While computing the best offline strategy is NP-complete, a good low cost heuristic is enough to complete the proof.

**Problem 20**

A set function \(f : 2^U \to \mathbb{R}_+\) is monotone if \(f(S) \leq f(T)\) whenever \(S \subseteq T \subseteq U\). Such a function is submodular if it has diminishing returns: whenever \(S \subseteq T \subseteq U\) and \(i \notin T\), then

\[
f(T \cup \{i\}) - f(T) \leq f(S \cup \{i\}) - f(S). \tag{2}
\]

We consider the problem of, given a function \(f\) and a budget \(k\), computing\(^1\)

\[
\max_{S \subseteq U:|S|=k} f(S). \tag{3}
\]

(a) Prove that the set coverage problem is a special case of this problem.

\(^1\)Don’t worry about how \(f\) is represented in the input. We assume that it is possible to compute \(f(S)\) from \(S\) in a reasonable amount of time.
Let $G = (V, E)$ be a directed graph and $p \in [0, 1]$ a parameter. Recall the cascade model (Lecture #15 notes):

- Initially the vertices in some set $S$ are “active,” all other vertices are “inactive.” Every edge is initially “undetermined.”
- While there is an active vertex $v$ and an undetermined edge $(v, w)$:
  - with probability $p$, edge $(v, w)$ is marked “active,” otherwise it is marked “inactive;”
  - if $(v, w)$ is active and $w$ is inactive, then mark $w$ as active.

Let $f(S)$ denote the expected number of active vertices at the conclusion of the cascade, given that the vertices of $S$ are active at the beginning. (The expectation is over the coin flips made for the edges.)

Prove that $f$ is monotone and submodular.

[Hint: prove that the condition (2) is preserved under convex combinations.]

(c) Let $f$ be a monotone submodular function. Define the greedy algorithm in the obvious way — at each of $k$ iterations, add to $S$ the element that increases $f$ the most. Suppose at some iteration the current greedy solution is $S$ and it decides to add $i$ to $S$. Prove that

$$f(S \cup \{i\}) - f(S) \geq \frac{1}{k} (OPT - f(S)),$$

where $OPT$ is the optimal value in (3).

[Hint: If you added every element in the optimal solution to $S$, where would you end up? Then use submodularity.]

(d) Prove that for every monotone submodular function $f$, the greedy algorithm is a $(1 - \frac{1}{e})$-approximation algorithm.

Problem 21

This problem considers the “[1, 2]” special case of the asymmetric traveling salesman problem (ATSP). The input is a complete directed graph $G = (V, E)$, with all $n(n - 1)$ directed edges present, where each edge $e$ has a cost $c_e$ that is either 1 or 2. Note that the triangle inequality holds in every such graph.

(a) This part considers a useful relaxation of the ATSP problem. A cycle cover of a directed graph $G = (V, E)$ is a collection $C_1, \ldots, C_k$ of simple directed cycles, each with at least two edges, such that every vertex of $G$ belongs to exactly one of the cycles. (A traveling salesman tour is the special case where $k = 1$.) Prove that given a directed graph with edge costs, a cycle cover with minimum total cost can be computed in polynomial time.

[Hint: bipartite matching.]

(b) Using (a) as a subroutine, give a $\frac{3}{2}$-approximation algorithm for the {1, 2} special case of the ATSP problem.

Problem 22

This problem considers a natural clustering problem, where it’s relatively easy to obtain a good approximation algorithm and a matching hardness of approximation bound.

The input to the metric $k$-center problem is the same as that in the metric TSP problem — a complete undirected graph $G = (V, E)$ where each edge $e$ has a nonnegative cost $c_e$, and the edge costs satisfy the triangle inequality ($c_{uv} + c_{uw} \geq c_{uw}$ for all $u, v, w \in V$). Also given is a parameter $k$. Feasible solutions correspond to choices of $k$ centers, meaning subsets $S \subseteq V$ of size $k$. 


For \( v \in V \), subset \( S \subseteq V \), define \( d(v, S) = \min_{s \in S} \{ c_{sv} \} \), i.e. the distance of \( v \) to the closest point in \( S \).
The goal of the metric \( k \)-center problem is to pick a set of \( k \) centers from \( V \) so as to minimize the furthest distance of a point to its nearest center:

\[
\min_{S \subseteq V : |S| = k} \max_{v \in V} d(v, S).
\]  

(a) Consider the following algorithm:

1. Pick any \( z \in V \) and set \( S = \{ z \} \)
2. while \( |S| < k \)
   \[ z = \arg\max_{v \in V} \{ d(v, S) \} \]
   \[ S = S \cup \{ z \} \]

Let \( S \) be the set of \( k \) points returned by the algorithm above. Let \( r = \max_{v \in V} d(v, S) \), \( x = \arg\max_{v \in V} d(v, S) \). Then show that the set of \( k + 1 \) points \( S \cup \{ x \} \) have minimum pairwise distance \( r \).

(b) Argue that the algorithm from part (a) is a 2-approximation algorithm for metric \( k \)-center.

(c) Recall the well-known \( NP \)-complete Dominating Set problem, where given an undirected graph \( G \) and a parameter \( k \), the goal is to decide whether or not \( G \) has a dominating set of size at most \( k \).

Using a reduction from the Dominating Set problem, prove that for every \( \epsilon > 0 \), there is no \((2-\epsilon)\)-approximation algorithm for the metric \( k \)-center problem, unless \( P = NP \).

[Hint: look to our reduction to TSP (Lecture #16 notes) for inspiration.]

Problem 23

In this problem we revisit the min cut problem (yet again). This time, we will show how you can obtain a minimum cut by rounding a fractional LP solution. This kind of rounding method is frequently used in the design of approximation algorithms for graph partitioning problems. Here, we use it to solve min cut exactly.

(a) Write down the dual for this LP formulation of max flow on directed graph \( G(V,E) \) with source \( s \), sink \( t \) and edge capacities \( u_e > 0 \) for \( e \in E \):

(Use the dual variable names indicated in parentheses after each constraint)

\[
\max \sum_{e \in \delta^-(t)} f_e
\]

subject to

\[
\sum_{e \in \delta^+(v)} f_e - \sum_{e \in \delta^-(v)} f_e \leq 0 \quad \text{for all } v \neq s, t \quad (z_v)
\]

\[
f_e \leq u_e \quad \text{for all } e \in E \quad (\ell_e)
\]

\[
f_e \geq 0 \quad \text{for all } e \in E.
\]

Notice that the usual flow conservation equality constraints have been replaced by inequalities. Argue that in an optimal solution to the primal, these flow conservation inequalities are tight.

(b) Let \((z^*, \ell^*)\) be an optimal dual solution. Define \( z^*_v = 0 \) and \( z^*_v = 1 \). Show that for all edges \( e = (u,v) \), \( \ell^*_e = \max(z^*_v - z^*_u, 0) \).

(c) We obtain an \((s,t)\) cut from the optimal dual solution by applying the following randomized algorithm:

Pick \( r \) uniformly and at random in \([0,1)\). Define \( S(r) = \{ v \in V | z^*_v \leq r \} \). Show that \((S(r), V \setminus S(r))\) is an \((s,t)\) cut.

A dominating set is a subset \( S \subseteq V \) of vertices such that every vertex \( v \in V \) either belongs to \( S \) or has a neighbor in \( S \).
(d) The random choice of $r$ gives rise to a distribution over cuts $(S(r), V \setminus S(r))$. Show that, for any edge $e$, the probability that $e$ contributes to the capacity of the cut $(S(r), V \setminus S(r))$ is exactly $\ell_e^*$. 

(e) Show that the expected capacity of the cut $(S(r), V \setminus S(r))$ is equal to the dual LP objective.

(f) Show that $(S(r), V \setminus S(r))$ is a min cut for every $r \in [0, 1)$. This means that we didn’t need randomization. For any choice of $r \in [0, 1)$, the algorithm returns a min cut!