CS 573: Algorithms, Fall 2014
Homework 3, due Monday, November 3, 23:59:59, 2014

Version 1.01

For submission guidelines see previous homeworks.

At other times you seemed to me either pitiable or contemptible, eunuchs, artificially confined to an eternal childhood, childlike and childish in your cool, tightly fenced, neatly tidied playground and kindergarten, where every nose is carefully wiped and every troublesome emotion is soothed, every dangerous thought repressed, where everyone plays nice, safe, bloodless games for a lifetime and every jagged stirring of life, every strong feeling, every genuine passion, every rapture is promptly checked, deflected and neutralized by meditation therapy.
– The Glass Bead Game, Hermann Hesse

Required Problems

1. Euclidean three dimensional matching. (20 PTS.)
You are given three disjoint sets R, G, B of n points in the plane. Our purposes is to output a set $\mathcal{T}$ of n disjoint triplets of points $(r_i, g_i, b_i) \in R \times G \times B$, such that all the points in $R \cup G \cup B$ are included in exactly one such triplet, and furthermore, the price function

$$f(\mathcal{T}) = \max_{i=1}^{n} \Big(\|r_i - g_i\| + \|g_i - b_i\| + \|b_i - r_i\| \Big)$$

is minimized (i.e., you are minimizing the maximum perimeter of the triangles you choose), where $\|p - q\|$ denotes the Euclidean distance between p and q. Provide a polynomial time constant approximation algorithm for this problem. **Prove** the correctness and the quality of approximation of your algorithm. The better the constant of approximation in your algorithm, the better your solution is.
(Hint: You would need to use network flow somewhere.)

2. Can you hear me? (20 PTS.)
You are given a set P of n points in the plane (i.e., location of n clients with phones), and a set Q of m points (i.e., base stations). The ith base station b_i, can serve at most α_i clients, and each client has to be in distance at most r_i from it, for $i = 1, \ldots, m$. Describe a polynomial time algorithm that computes for each client which base station it should use, and no base station is assigned more clients that it can use. What is the running time of your algorithm?

3. Unique Cut. (20 PTS.)

(A) (10 PTS.) Let $G = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative edge capacities $\{c_e\}$. Give a polynomial-time algorithm to decide whether G has a *unique* minimum s-t cut (i.e., an s-t of capacity strictly less than that of all other s-t cuts).

(B) (10 PTS.) THE GOOD, THE BAD, AND THE MIDDLE.
Suppose you're looking at a flow network G with source s and sink t, and you want to be able to express something like the following intuitive notion: Some nodes are clearly on the "source side" of the main bottlenecks; some nodes are clearly on the "sink side" of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we try making this idea precise.
Here's one way to divide the nodes of G into three categories of this sort.

- We say a node v is ***upstream*** if, for all minimum s-t cuts (A, B), we have $v \in A$ – that is, v lies on the source side of every minimum cut.
- We say a node v is ***downstream*** if, for all minimum s-t cuts (A, B), we have $v \in B$ – that is, v lies on the sink side of every minimum cut.

- We say a node v is ***central*** if it is neither upstream nor downstream; there is at least one minimum s-t cut (A, B) for which $v \in A$, and at least one minimum s-t cut (A', B') for which $v \in B'$.

Give an algorithm that takes a flow network G and classifies each of its nodes as being upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a *single* maximum flow.

4. Prove infeasibility. (20 PTS.)

You are trying to solve a circulation problem, but it is not feasible. The problem has demands, but no capacity limits on the edges. More formally, there is a graph $G = (V, E)$, and demands d_v for each node v (satisfying $\sum_{v \in V} d_v = 0$), and the problem is to decide if there is a flow f such that $f(e) \geq 0$ and $f^{in}(v) - f^{out}(v) = d_v$ for all nodes $v \in V$. Note that this problem can be solved via the circulation algorithm by setting $c_e = +\infty$ for all edges $e \in E$. (Alternately, it is enough to set c_e to be an extremely large number for each edge – say, larger than the total of all positive demands d_v in the graph.)

You want to fix up the graph to make the problem feasible, so it would be very useful to know why the problem is not feasible as it stands now. On a closer look, you see that there is a subset U of nodes such that there is no edge into U, and yet $\sum_{v \in U} d_v > 0$. You quickly realize that the existence of such a set immediately implies that the flow cannot exist: The set U has a positive total demand, and so needs incoming flow, and yet U has no edges into it. In trying to evaluate how far the problem is from being solvable, you wonder how big the demand of a set with no incoming edges can be.

Give a polynomial-time algorithm to find a subset $S \subset V$ of nodes such that there is no edge into S and for which $\sum_{v \in S} d_v$ is as large as possible subject to this condition.

(Hint: Think about strong connected components, and ho to use them in this case.)

5. Maximum Flow By Scaling (20 PTS.)

Let $G = (V, E)$ be a flow network with source s, sink t, and an integer capacity $c(u, v)$ on each edge $(u, v) \in E$. Let $C = max_{(u,v) \in E} c(u, v)$.

(A) (3 PTS.) Argue that a minimum cut of G has capacity at most $C|E|$.

(B) (3 PTS.) For a given number K, show that an augmenting path of capacity at least K can be found in $O(E)$ time, if such a path exists.

The following modification of FORD-FULKERSON-METHOD can be used to compute a maximum flow in G.

```
maxFlow-By-Scaling($G, s, t$)
1   $C \leftarrow max_{(u,v) \in E} c(u, v)$
2   initialize flow $f$ to 0
3   $K \leftarrow 2^{\lfloor \lg C \rfloor}$
4   while $K \geq 1$ do {
5       while (there exists an augmenting path $p$ of
                  capacity at least $K$) do {
6           augment flow $f$ along $p$
        }
7       $K \leftarrow K/2$
    }

8   return $f$
```

(C) (3 PTS.) Argue that **maxFlow-By-Scaling** returns a maximum flow.

(D) (3 PTS.) Show that the capacity of a minimum cut of the residual graph G_f is at most $2K|E|$ each time line 4 is executed.

(E) (4 PTS.) Argue that the inner **while** loop of lines 5-6 is executed $O(|E|)$ times for each value of K.

(F) (4 PTS.) Conclude that **maxFlow-By-Scaling** can be implemented so that it runs in $O(E^2 \lg C)$ time.