# ☙ New CS 473: Algorithms, Spring 2015 ❧
# Homework 8

Due Tuesday, April 6, 2015 at 5pm

---

All homework must be submitted electronically via Moodle as separate PDF files, one for each numbered problem. Please see the course web site for more information.

---

1. Suppose you are given a directed graph $G = (V, E)$, two vertices $s$ and $t$, a capacity function $c: E \to \mathbb{R}^+$, and a second function $f: E \to \mathbb{R}$.

   (a) Describe and analyze an efficient algorithm to determine whether $f$ is a maximum $(s, t)$-flow in $G$.

   (b) Describe and analyze an efficient algorithm to determine whether $f$ is the *unique* maximum $(s, t)$-flow in $G$.

   Do not assume anything about the function $f$.

2. A new assistant professor, teaching maximum flows for the first time, suggested the following greedy modification to the generic Ford-Fulkerson augmenting path algorithm. Instead of maintaining a residual graph, the greedy algorithm just reduces the capacity of edges along the augmenting path. In particular, whenever the algorithm saturates an edge, that edge is simply removed from the graph.

```
GreedyFlow(G, c, s, t):
  for every edge e in G
      f(e) ← 0

  while there is a path from s to t in G
      π ← arbitrary path from s to t in G
      F ← minimum capacity of any edge in π
      for every edge e in π
          f(e) ← f(e) + F
          if c(e) = F
              remove e from G
          else
              c(e) ← c(e) − F

  return f
```

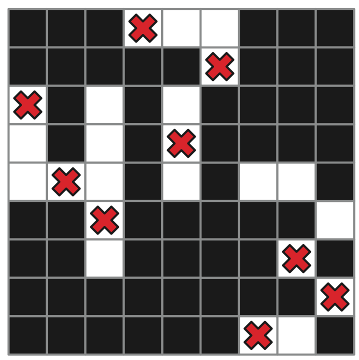
   (a) Prove that GreedyFlow does not always compute a maximum flow.

   (b) Prove that GreedyFlow is not even guaranteed to compute a good approximation to the maximum flow. That is, for any constant $\alpha > 1$, describe a flow network $G$ such that the value of the maximum flow is more than $\alpha$ times the value of the flow computed by GreedyFlow. *[Hint: Assume that GreedyFlow chooses the worst possible path $\pi$ at each iteration.]*

(c) Prove that for any flow network, if the Greedy Path Fairy tells you precisely which path $\pi$ to use at each iteration, then GreedyFlow does compute a maximum flow. (Sadly, the Greedy Path Fairy does not actually exist.)

3. Suppose we are given an $n \times n$ square grid, some of whose squares are colored black and the rest white. Describe and analyze an algorithm to determine whether tokens can be placed on the grid so that

   - every token is on a white square;
   - every row of the grid contains exactly one token; and
   - every column of the grid contains exactly one token.



Your input is a two dimensional array $IsWhite[1..n, 1..n]$ of booleans, indicating which squares are white. Your output is a single boolean. For example, given the grid above as input, your algorithm should return True.

# New CS 473 Spring 2015 — Homework 8 Problem 1

| Name: | NetID: |
|---|---|
| Name: | NetID: |
| Name: | NetID: |

---

(a) Describe and analyze an efficient algorithm to determine whether a given function $f$ is a maximum $(s,t)$-flow in a given graph $G$.

(b) Describe and analyze an efficient algorithm to determine whether a given function $f$ is the *unique* maximum $(s,t)$-flow in a given network $G$.

Do not assume anything about the function $f$.

---

# New CS 473 Spring 2015 — Homework 8 Problem 2

| Name: | NetID: |
|---|---|
| Name: | NetID: |
| Name: | NetID: |

---

(a) Prove that GreedyFlow does not always compute a maximum flow.

(b) Prove that GreedyFlow is not even guaranteed to compute a good approximation to the maximum flow. That is, for any constant $\alpha > 1$, describe a flow network $G$ such that the value of the maximum flow is more than $\alpha$ times the value of the flow computed by GreedyFlow. *[Hint: Assume that GreedyFlow chooses the worst possible path $\pi$ at each iteration.]*

(c) Prove that for any flow network, if the Greedy Path Fairy tells you precisely which path $\pi$ to use at each iteration, then GreedyFlow does compute a maximum flow. (Sadly, the Greedy Path Fairy does not actually exist.)

---

# New CS 473 Spring 2015 — Homework 8 Problem 3

| | |
|---|---|
| Name: | NetID: |
| Name: | NetID: |
| Name: | NetID: |

---

Describe and analyze an algorithm to determine whether $n$ tokens can be placed on a given $n \times n$ grid, in which some squares are white and the rest black, so that

- every token is on a white square;
- every row of the grid contains exactly one token; and
- every column of the grid contains exactly one token.

---