**cs473 Algorithms**

# Problem Set #5

Prof. Michael A. Forbes — Due: *Mon., 2026-03-30 17:00*

Submissions must obey the following guidelines when using *reductions* to solve problems, in particular when using algorithms for computing maximum flows as a black-box.

- *Construction:* Submissions must completely describe the relevant capacitated graph by describing the vertices, the edges (with direction), and capacities. The source $s$ and sink $t$ must also be identified.
- *Forward reduction:* Submissions must describe an algorithm to, given the original input, construct the above mentioned capacitated graph. This includes a correctness and complexity analysis.
- *Backwards reduction:* Submissions must describe an algorithm to, given a maximum flow in the constructed capacitated graph, efficiently solve the original problem. This includes a correctness and complexity analysis. An argument for correctness will typically require two directions. First, if the resulting maximum flow is *large* in value, then the produced solution to the original problem is correct. Second, if the resulting maximum flow is *small* in value, then the produced solution to the original problem is correct.
- *Complexity:* The overall complexity of the entire algorithm must be specified as a function of the *original* input (**not** as a function of the constructed capacitated graph).
- Submissions should assume that maximum flows can be computed in $O(nm)$ time on graphs with $n$ vertices and $m$ edges, and in particular should **not** reproduce a maximum flow algorithm unless required.

Analogous guidelines exist for reductions to other problems whose algorithms were presented in lecture or auxiliary reading, such as shortest paths with negative edge lengths, or maximum bipartite matching.

All problems are of equal value.

1. Let $G = (V, E)$ be a (directed) capacitated graph with non-negative integral capacities $(c_e)_{e \in E}$. Let $k$ be an integer.
   (a) Design an efficient algorithm to decide if $G$ has a flow $f$ of value $k$, where all edge flows $f_e$ are *even*.
   (b) Design an efficient algorithm to decide if $G$ has a flow $f$ of value $k$, where all edge flows $f_e$ are *odd*.
2. Overlay Networks. Kleinberg-Tardos Chapter 13, Problem #4.
3. In this problem we improve the analyses from lecture.
   (a) Prove that in the contention resolution process on $n$ players, the time until all players successfully communicate is $O(n \lg n)$.
   *Hint:* Use a formula we derived for the expectation of a random variable.

(b) Consider the following algorithm for picking a splitter, which we used to develop our randomized selection algorithm. Let $0 < \epsilon < 1$ be a parameter.

```
pick-splitter_ε(a = (a_1, ..., a_n)):
    while
        randomly pick i ∈ {1, ..., n}
        partition a into b, a_i, c with b_j < a_i < c_k for any j, k
        let r = |b| + 1
        if εn ≤ r ≤ (1 − ε)n, return a_i
```

In lecture, we considered the above algorithm when $\epsilon = 1/4$.

i. For each value of $0 < \epsilon < 1$, give an analysis of the expected number of comparisons used for our **selection** algorithm, when using the above **pick-splitter**$_\epsilon$ algorithm.

ii. Choose a value of $\epsilon$ to minimize the number of comparisons from part (i).