CS 255: Practice Final

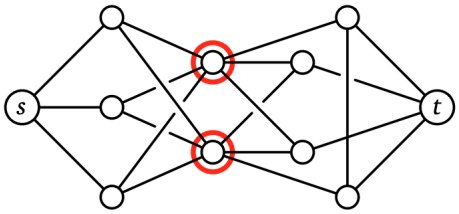
The final exam will be cumulative, but will focus more on the material after the midterm. Some details:

- The exam is **Tuesday May 21** from 2:45-5pm.
- You will have the complete **135 minutes**.
- You are allowed to bring **two** handwritten double-sided 8.5×11 cheat sheets.
- No phones, laptops, tablets, books, or resources are allowed.
- You will not need a calculator.
- I strongly recommend you read each problem before starting.
- We will discuss solutions to these problems in an optional **review session** on **Thursday May 9** (in class).

The actual exam will have about 10 True/False questions similar to the midterm

1. The Island of Sodor is home to a large number of towns and villages, connected by an extensive rail network. Recently, several cases of a deadly contagious disease have been reported in the village of Skarloey. The controller of the Sodor railway plans to close down certain railway stations to prevent the disease from spreading to Tidmouth, his home town. No trains can pass through a closed station. To minimize expense (and public notice), he wants to close down as few stations as possible. However, we cannot close the Skarloey station, because that would expose him to the disease, and head cannot close the Tidmouth station, because then he couldn't visit his favorite pub. Describe and analyze an algorithm to find the minimum number of stations that must be closed to block all the rail travel from Skarloey to Tidmouth. The Sodor rail network is represented by an undirected graph, with a vertex for each station and an edge for each rail connection between two stations. Two special vertices s and t represent the station in Skarloey and Tidmouth.

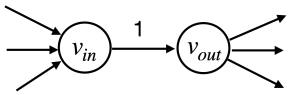
For example, in the following graph your algorithm should return the integer 2.



Solution. We want something like a min-cut to disconnect s from t, but we can't delete edges of the graph. We model closing stations by giving each vertex (station) a capacity of 1, and all edges (tracks) capacity ∞. This ensures that the vertex capacities create the bottleneck limiting the max-flow from s to t. A min-cut in the modified graph will allow us to identify the stations we need to close.

First, we replace each undirected edge $(u, v) \in E$, with two directed edges $u \to v$ and $v \to u$. We give all of these edges infinite capacity. Next, we give each vertex a capacity as follows. We replace each vertex $v \neq s, t$ with two vertices v_{in} and v_{out}. For each directed edge into v, $u \to v$, connects to v_{in}. Each edge out of v, $v \to u$, leaves from v_{out}. We make a single directed edge $v_{in} \to v_{out}$ with capacity 1. This ensures that at most one s-t path can pass through the vertex v.

(a) Replacing undirected edges with two directed edges.



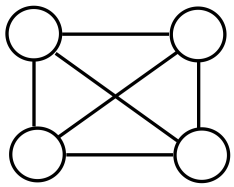
(b) Creating vertex capacities.

We compute a min-cut in the graph in the modified graph. The min-cut can only contain edges $v_{in} \to v_{out}$ with capacity 1, since all others have capacity ∞. If we remove these edges (close the corresponding stations), then we disconnect s from t, as desired.

We construct the modified graph by brute force in $O(V + E)$ time. The graph contains $|V'| = 2|V|$ vertices and $|E'| = 2|E|+|V| = O(E)$ edges. Observe that the max-flow value f in H is no more than $|V|$ due to the vertex capacities of 1. Therefore, Ford-Fulkerson runs in $O(E \cdot f) = O(E \cdot V)$ time. Computing the min-cut takes an additional $O(V + E)$ time with BFS/DFS. Therefore, the runtime is dominated by the max-flow computation $O(E \cdot V)$.

2. A *relaxed 3 coloring* of a graph G assigns each vertex one of the three colors such that *at most one* edge in G has both endpoints the same color.

(a) Give an example of a graph that has a relaxed 3 coloring, but does not have a proper 3 coloring.



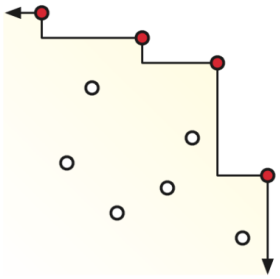
(b) Prove that it is NP-hard to determine whether a given graph has a relaxed 3 coloring.

Solution. We give a reduction from 3-Color. Call the example from part (a) the gadget. Given a 3-Color instance G, we create an new instance $H = G \cup$ gadget. That is, just G and the gadget disconnected from each other. This reduction clearly takes polynomial-time. We claim G is 3 colorable if and only if H has a relaxed 3 coloring.

$\implies$ Any valid 3 coloring of G, along with any relaxed 3 coloring of the gadget gives a valid relaxed 3 coloring of H.

$\impliedby$ If H has a relaxed 3 coloring, then the gadget must contain one edge with both endpoints of the same color. Since a relaxed 3 coloring allows for *at most one* such edge, then the remaining portion of H (that is G) must have a valid 3 coloring.

3. Let S be an arbitrary set of n points in the plane with distinct x and y coordinates. A point p is Pareto-optimal if no other point in S is both above and to the right of p. Intuitively, the Pareto-optimal points define a 'staircase' with all the other points of S below them.



(a) Describe and analyze an algorithm that computes the Pareto-optimal points of S in $O(n \log n)$ time.

Solution. Observe that the right most point is Pareto-optimal since no other point is to its right. If we sort the points by their x coordinates and scan from right to left, then any point with y coordinate greater than all points to its right is Pareto-optimal. A pseudocode implementation is given below.

```
GetParetoOptimal(S):
    sort and relabel points in increasing order of their x-coordinates.
    count ← 0
    y_max ← −∞
    for i ← n to 1 do
        if S[i].y > y_max then
            count ← count + 1
            y_max ← S[i].y
    return count
```

This algorithm runs in $O(n \log n)$ time.

(b) Suppose each point in S is chosen independently and uniformly at random from the unit square $[0, 1] \times [0, 1]$. What is the exact expected number of Pareto-optimal points? [Hint: What is the probability that the left most point is Pareto-optimal?]

Solution. The left most point is Pareto-optimal if it has the the largest y coordinate. Since the y-coordinates are generated uniformly at random, then the probability that any one of them is largest is $1/n$. On the other hand, the right most point is guaranteed to be Pareto-optimal. In general, the kth point from the right is Pareto-optimal if and only if it has the largest y coordinate out the the k right most points. This happens with probability $1/k$, since the points are generated uniformly at random.

Define the indicator random variables $X_k = 1$ if the kth point from the right is P.O., 0 otherwise. As discussed above

$$E[X_k] = P(X_k = 1) = \frac{1}{k}.$$

Further, $X = \sum_{k=1}^{n} X_k$, gives the total number of Pareto-optimal points. By linearity of expectation

$$E[X] = \sum_{k=1}^{n} E[X_k] = \sum_{k=1}^{n} P(X_k = 1) = \sum_{k=1}^{n} \frac{1}{k} = H_n,$$

where H_n is the nth Harmonic number.

4. Consider the following multithreaded pseudocode for transposing an $n \times n$ matrix A in place:

```
P-Transpose(A):
    n ← A.rows
    parallel for j ← 2 to n
        for i ← 1 to j − 1
            exchange a_ij with a_ji
```

What is the work, span, and parallelism of this algorithm?

Solution.

- Work T_1 is runtime on a single processor. Just ignore all parallel keywords and find the runtime, which gives $\Theta(n^2)$.
- Span T_∞ is runtime with infinitely man processors. Let $iter_\infty(j)$ be the span of iteration j. Then
$$T_\infty(n) = \log n + \max_{j=2}^{n} iter_\infty(j) = \log n + \Theta(n) = \Theta(n).$$
- The parallelism is $T_1/T_\infty = \Theta(n)$.