CS 473 ✧ Fall 2022

"Homework" 10

Practice only — Solutions will be released on Monday, December 5, 2022

---

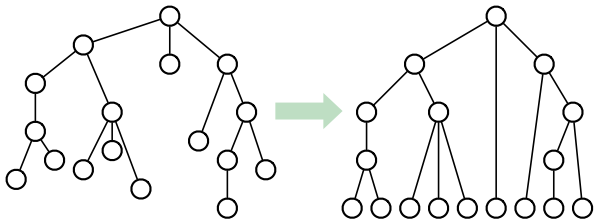
1. Alex and Bo are playing another game with even more complex rules. Each player independently chooses an integer between 0 and n, then both players simultaneously reveal their choices, and finally they get points based on those choices.

   Chris and Dylan are watching the game, but they don't really understand the scoring rules, so instead, they decide to place bets on the *sum* of Alex and Bo's choices. They both somehow know the probabilities that Alex and Bo use, and they want to figure out the probability of each possible sum.

   Suppose Chris and Dylan are given a pair of arrays $A[0..n]$ and $B[0..n]$, where $A[i]$ is the probability that Alex chooses i, and $B[j]$ is the probability that Bo chooses j. Describe and analyze an algorithm that computes an array $P[0..2n]$, where $P[k]$ is the probability that the sum of Alex and Bo's choices is equal to k.

2. Suppose you are given a rooted tree T, where every edge e has two associated values: a non-negative *length* $\ell(e)$ and a *cost* $\$(e)$ (which could be positive, negative, or zero). Your goal is to add a non-negative *stretch* $s(e) \geq 0$ to the length of every edge e in T, subject to the following conditions:

   - Every root-to-leaf path π in T has the same total stretched length $\sum_{e\in\pi}(\ell(e)+s(e))$
   - The total *weighted* stretch $\sum_e s(e)\cdot\$(e)$ is as small as possible.

   (a) Describe an instance of this problem with no optimal solution.

   (b) Give a concise linear programming formulation of this problem.

   (c) Suppose that for the given tree T and the given lengths and costs, the optimal solution to this problem is unique. Prove that in the optimal solution, $s(e) = 0$ for every edge on some longest root-to-leaf path in T. In other words, prove that the optimally stretched tree has the same depth as the input tree. *[Hint: What is a basis in your linear program? When is a basis feasible?]*

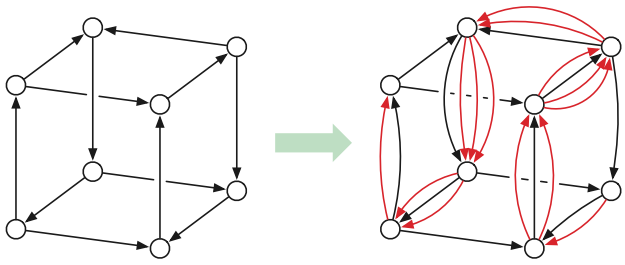
   (d) Describe and analyze a self-contained algorithm that solves this problem in (explicit) polynomial time. Your algorithm should either compute the minimum total weighted stretch, or report correctly that the total weighted stretch can be made arbitrarily negative.

3. An *Euler tour* in a directed graph G is a closed walk (starting and ending at the same vertex) that traverses every edge in G exactly once; a directed graph is *Eulerian* if it has an Euler tour. Euler tours are named after Leonhard Euler, who was the first person to systematically study them, starting with the Bridges of Königsberg puzzle.

   (a) Prove that a directed graph G with no isolated vertices is Eulerian if and only if (1) G is strongly connected—for any two vertices u and v, there is a directed walk in G from u to v and a directed walk in G from v to u—and (2) the in-degree of each vertex of G is equal to its out-degree. *[Hint: Flow decomposition!]*

   (b) Suppose that we are given a strongly connected directed graph G with no isolated vertices that is *not* Eulerian, and we want to make G Eulerian by duplicating existing edges. Each edge e has a duplication cost $€(e) \geq 0$. We are allowed to add as many copies of an existing edge e as we like, but we must pay $€(e)$ for each new copy. On the other hand, if G does not already have an edge from vertex u to vertex v, we cannot add a new edge from u to v.

   Describe an algorithm that computes the minimum-cost set of edge-duplications that makes G Eulerian.



Making a directed cube graph Eulerian.

4. *Reservoir sampling* is a method for choosing an item uniformly at random from an arbitrarily long stream of data; for example, the sequence of packets that pass through a router, or the sequence of IP addresses that access a given web page. Like all data stream algorithms, this algorithm must process each item in the stream quickly, using very little memory.

```
GetOneSample(stream S):
  ℓ ← 0
  while S is not done
      x ← next item in S
      ℓ ← ℓ + 1
      if Random(ℓ) = 1
          sample ← x        (⋆)
  return sample
```

   At the end of the algorithm, the variable ℓ stores the length of the input stream S; this number is *not* known to the algorithm in advance. If S is empty, the output of the algorithm is (correctly!) undefined.

   In the following problems, S denotes a stream of (unknown) length n.

(a) Prove that the item returned by GETONESAMPLE(S) is chosen uniformly at random from S.

(b) What is the *exact* expected number of times that GETONESAMPLE(S) executes line ($\star$)?

(c) What is the *exact* expected value of ℓ when GETONESAMPLE(S) executes line ($\star$) for the *last* time?

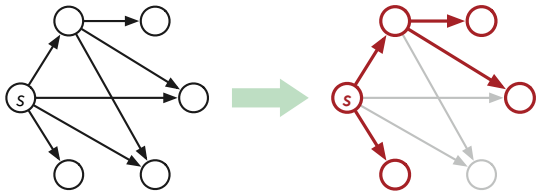
(d) What is the *exact* expected value of ℓ when either GETONESAMPLE(S) executes line ($\star$) for the *second* time (or the algorithm ends, whichever happens first)?

(e) Describe and analyze an algorithm that returns a subset of k distinct items chosen uniformly at random from a data stream of length at least k. The integer k is given as part of the input to your algorithm. Prove that your algorithm is correct.

For example, if $k = 2$ and the stream contains the sequence $\langle ♠, ♥, ♦, ♣ \rangle$, the algorithm would return the subset $\{♦, ♠\}$ with probability $1/6$.

5. Suppose you are given a directed acyclic graph G with a single source vertex s. Describe an algorithm to determine whether G contains a ***spanning binary tree***. Your algorithm is looking for a spanning tree T of G, such that every vertex in G has at most two outgoing edges in T and every vertex of G *except* s has exactly one incoming edge in T.

For example, given the dag on the left below as input, your algorithm should FALSE, because the largest binary subtree excludes one of the vertices.



6. Suppose you are given an arbitrary directed graph $G = (V, E)$ with arbitrary edge weights $\ell : E \to \mathbb{R}$. Each edge in G is colored either red, white, or blue to indicate how you are permitted to modify its weight:

- You may increase, but not decrease, the length of any red edge.
- You may decrease, but not increase, the length of any blue edge.
- You may not change the length of any black edge.

The *cycle nullification* problem asks whether it is possible to modify the edge weights—subject to these color constraints—so that *every cycle in G has length* 0. Both the given weights and the new weights of the individual edges can be positive, negative, or zero. To keep the following problems simple, assume that G is strongly connected.

(a) Describe a linear program that is feasible if and only if it is possible to make every cycle in G have length 0. *[Hint: Pick an arbitrary vertex s, and let* dist(v) *denote the length of every walk from s to v.]*

(b) Construct the dual of the linear program from part (a). *[Hint: Choose a convenient objective function for your primal LP.]*

(c) Give a self-contained description of the combinatorial problem encoded by the dual linear program from part (b). Do not use the words "linear", "program", or "dual". Yes, you have seen this problem before.

(d) Describe and analyze a self-contained algorithm to determine *in* $O(EV)$ *time* whether it is possible to make every cycle in G have length 0, using your dual formulation from part (c). Do not use the words "linear", "program", or "dual".