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.

   Solution: For each index k, we have

$$P[k] = \sum_{i+j=k} A[i] \cdot B[j].$$

   In other words, P is the convolution of A and B. So we can compute P in $\boldsymbol{O(n \log n)}$ ***time*** using Fast Fourier transforms. ■

2. 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* $\text{dist}(v)$ *denote the length of every walk from s to v.]*

Solution: Our linear program is based on the following observation, suggested by the hint: **All cycles in G have length 0 if and only if, for all vertices s and t of G, all walks in G from s to t have the same length.**

$\Rightarrow$ Suppose all cycles in G have length 0. Fix two vertices s and t. Let α and β be any two walks from s to t, and let γ be any walk from t to s (which must exist because G is strongly connected). The closed walks $\alpha \cdot \gamma$ and $\beta \cdot \gamma$ are composed of cycles and therefore have length zero. Thus α and β have the same length, namely the negation of the length of γ. We conclude that all walks from s to t have the same length.

$\Leftarrow$ Suppose for all vertices s and t, all walks from s to t have equal length. Then in particular, for every vertex s, every closed walk through s has the same length, including the trivial walk with no edges. Thus, every closed walk in G (and in particular every cycle in G) has length zero.

Fix an arbitrary vertex s. The following linear program has a variable $dist(v)$ for every vertex v, which represents the length of *every* walk from s to v *with respect to the new edge lengths*.

$$\begin{array}{lrcll} \text{maximize} & & 0 & & \\ \text{subject to} & dist(v) - dist(u) & \geq & \ell(u \to v) & \text{for every red edge } u \to v \\ & dist(v) - dist(u) & = & \ell(u \to v) & \text{for every black edge } u \to v \\ & dist(v) - dist(u) & \leq & \ell(u \to v) & \text{for every blue edge } u \to v \\ & dist(s) & = & 0 & \end{array}$$

Because we only care about feasibility, the objective function doesn't actually matter here; the objective function 0 is convenient for part (b). (For the same reason, the last constraint $dist(s) = 0$ is actually redundant.) ■

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

Solution: We have a dual variable $f(u{\to}v)$ for each edge $u{\to}v$, corresponding to the primal constraints.

$$\begin{aligned}
\text{minimize} \quad & \sum_{u\to v} f(u{\to}v)\cdot \ell(u{\to}v) \\
\text{subject to} \quad & \sum_{u\to v} f(u{\to}v) - \sum_{v\to w} f(v{\to}w) \;=\; 0 \quad \text{for every vertex } v \neq s \\
& f(u{\to}v) \;\le\; 0 \quad \text{for every red edge } u{\to}v \\
& f(u{\to}v) \;\ge\; 0 \quad \text{for every blue edge } u{\to}v
\end{aligned}$$

I called the dual variable f because the vertex constraints look like flow conservation; that's also why I chose the primal objective vector 0.

(If we omit the redundant constraint $dist(s) = 0$ from the primal LP in part (a), the dual LP includes a redundant conservation constraint at s.) ■

(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.

Solution: Let H be the graph obtained from G by inserting the reversal $v{\to}u$ of every black or red edge $u{\to}v$, defining $\ell(v{\to}u) = -\ell(u{\to}v)$ for each reversed edge, and then deleting every original red edge. The dual LP describes an uncapacitated minimum-cost flow problem in H.

I claim that ***all cycles in G can be nullified if and only if H does not contain a negative cycle.*** This claim follows immediately from the observation that our linear program from part (a) is infeasible if and only if its dual linear program from part (b) is unbounded.

We can also give a self-contained proof as follows. As usual, the proof has two parts.

⇒ Suppose all cycles in G can be nullified. Let $\ell' : E \to \mathbb{R}$ be any new length function such that all cycles in G have length 0. Our proof in part (a) implies that for every pair of vertices s and t, all walks from s to t have the same length. Fix an arbitrary vertex s in G, and then for each vertex v, let $dist'(v)$ denote the common length of every walk from s to v in G with respect to the new edge lengths ℓ'. The distances $dist'(v)$ are the components of the optimal solution to our LP from part (a). Think of each $dist'(v)$ as an estimated shortest-path distance in H;

To prove that H has no negative cycles (with respect to the original edge lengths ℓ), it suffices to show that no edge in H is tense (with respect to the distances $dist'$). Let $u{\to}v$ be an arbitrary edge in H; there are two cases to consider:

– If $u \to v$ is a (blue or black) edge in G, then

$$dist'(v) - dist'(u) = \ell'(u \to v) \leq \ell(u \to v),$$

which means $u \to v$ is not tense in H.

– If $v \to u$ is a (red or black) edge in G, then

$$dist'(u) - dist'(v) = \ell'(v \to u) \geq \ell(v \to u) = -\ell(u \to v)$$

and thus $dist'(v) - dist'(u) \leq \ell(u \to v)$, which means $u \to v$ is not tense in H.

$\Leftarrow$ Now suppose H does not contain a negative cycle. Then shortest-path distances in H are well-defined. Add a new vertex $\hat{s}$ with zero-length edges to every vertex in H, and then for each vertex v, let $dist(v)$ denote the shortest-path distance from $\hat{s}$ to v in H. (We need the extra vertex $\hat{s}$ because there might be no vertex that can reach every other vertex in H, even though the original graph G is strongly connected.) Finally, for every edge $u \to v$ in G, define $\ell'(u \to v) := dist(v) - dist(u)$.

Let $u \to v$ be an arbitrary edge in G. We need to verify that $\ell'(u \to v)$ is at least, at most, or equal to $\ell(u \to v)$, depending on the color of $u \to v$. There are three cases to consider.

– If $u \to v$ is blue or black, then $\ell'(u \to v) = dist(v) - dist(u) \leq \ell(u \to v)$ because $u \to v$ is not tense in H.

– Symmetrically, if $u \to v$ is red or black, then $\ell'(u \to v) = dist(v) - dist(u) \geq -\ell(v \to u) = \ell(u \to v)$ because $v \to u$ is not tense in H.

– Thus, if $u \to v$ is black, then $\ell'(u \to v) = \ell(u \to v)$.

We conclude that all new edge lengths are consistent with the edge colors.

Finally, every cycle $v_0 \to v_1 \to \ldots \to v_{k-1} \to v_0$ in G has length zero, because

$$\sum_{i=0}^{k-1} \ell'(v_i \to v_{i+1 \bmod k}) = \sum_{i=0}^{k-1} \big(dist(v_{i+1 \bmod k}) - dist(v_i)\big) = 0.$$

(Each term $dist(v_i)$ appears once positively and once negatively in the second sum.) ■

(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".

Solution: We can construct the graph H in $O(V+E)$ time, and then find negative cycles in H using a modification of the Bellman-Ford shortest-path algorithm, as described in the textbook, in $O(VE)$ time. ■

3. Your eight-year-old cousin Elmo decides to teach his favorite new card game to his baby sister Daisy. At the beginning of the game, n cards are dealt face up in a long row. Each card is worth some number of points, which may be positive, negative, or zero. Then Elmo and Daisy take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, each player can decide which of the two cards to take. When the game ends, the player that has collected the most points wins.

   Daisy isn't old enough to get this whole "strategy" thing; she's just happy to play with her big brother. When it's her turn, she takes the either leftmost card or the rightmost card, each with probability $1/2$.

   Elmo, on the other hand, *really* wants to win. Having never taken an algorithms class, he follows the obvious greedy strategy—when it's his turn, Elmo *always* takes the card with the higher point value.

   Describe and analyze an algorithm to determine Elmo's expected score, given the initial sequence of n cards as input. Assume Elmo moves first, and that no two cards have the same value.

Solution: Let $C[1..n]$ be the input card values. Assume all card values are distinct (since otherwise we don't know how Elmo plays).

For any indices i and j, let $EES(i,j)$ denote Elmo's Expected Score if Elmo plays first, starting with the cards $C[i..j]$. We need to compute $EES(1,n)$. This function obeys the recurrence

$$EES(i,j) = \begin{cases} 0 & \text{if } i > j \\ C[i] & \text{if } i = j \\ C[i] + \dfrac{EES(i+1,j-1) + EES(i+2,j)}{2} & \text{if } i < j \text{ and } C[i] > C[j] \\ C[j] + \dfrac{EES(i+1,j-1) + EES(i,j-2)}{2} & \text{if } i < j \text{ and } C[i] < C[j] \end{cases}$$

This recurrence can be memoized into a two-dimensional array $EES[1..n, 1..n]$, which we index by i and j. We can fill the array by decreasing i in the outer loop and increasing j in the inner loop, in $\boldsymbol{O(n^2)}$ ***time***. ■

4. You are attending a gala on the planet Krypton which k people are attending. There are n days in a Kryptonian year. Assume that the birthday of each person is on a uniformly random day in the year (and birthdays of different people are independent). We say that a triple-collision occurs whenever *three* people have the same birthday.

   Find a threshold value k^* for triple collisions. In other words, (i) if $k = o(k^*)$m the probability of having any triple collision is $o(1)$, and (ii) if $k = \omega(k^*)$, the probability of having a triple collision is $1 - o(1)$.[1]

   (a) Guess a threshold value k^*. You will prove its validity in parts (b) and (c) below.

   Solution: For each triple p, let $X_p = 1$ if all three people in p have the same birthday. We immediately have

   $$\mathrm{E}[X_p] = n \cdot \frac{1}{n^3} = \frac{1}{n^2}.$$

   Let $X = \sum_p X_p$ denote the total number of triple collisions. Linearity of expectation implies

   $$\mathrm{E}[X] = \sum_p \mathrm{E}[X_p] = \binom{k}{3}\frac{1}{n^2} \leq \frac{k^3}{n^2}.$$

   Based on this we should guess $\boxed{\boldsymbol{k^* = n^{2/3}}}$. Then if $k = o(k^*)$, we have

   $$\mathrm{E}[X] \leq \frac{k^3}{n^2} = \frac{o((k^*)^3)}{n^2} = \frac{o((n^{2/3})^3)}{n^2} = \frac{o(n^2)}{n^2} = o(1),$$

   and symmetrically if $k = \omega(k^*)$, we have $\mathrm{E}[X] = \omega(1)$. We will prove that this intuition is correct in the remaining parts. ■

   (b) Prove that probability of having any triple collision is $o(1)$ when $k = o(k^*)$. That is, if $k = ck^*$, the probability approaches 0 as c approaches 0.

   Solution: Suppose $k = ck^*$ where $c \to 0$. Them Markov's inequality implies

   $$\Pr[X \geq 1] \leq \mathrm{E}[X] = c^3 \frac{(k^*)^3}{n^2} = c^3 \to 0.$$

   ■

[1] Recall that $f(n) = o(g(n))$ means that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ and $f(n) = \omega(g(n))$ means that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

(c) Show that for any non-negative integer random variable X, we have

$$\Pr[X = 0] \leq \frac{\mathrm{E}[(X - \mathrm{E}[X])^2]}{\mathrm{E}[X]^2}.$$

Solution: For any non-negative integer random variable X, we have

$$\Pr[X = 0] \leq \Pr\left[(X - \mathrm{E}[X])^2 \geq E[X]^2\right],$$

because the event on the right contains the event on the left. Applying Markov's inequality to the random variable $(X - \mathrm{E}[X])^2$ gives us

$$\Pr[X = 0] \;\leq\; \Pr\left[(X - \mathrm{E}[X])^2 \geq E[X]^2\right] \;\leq\; \frac{\mathrm{E}[(X - \mathrm{E}[X])^2]}{\mathrm{E}[X]^2}.$$

The quantity $\mathrm{E}[(X - \mathrm{E}[X])^2]$ is called the *variance* of X; from now on we will abbreviate

$$\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}[X]^2.$$

■

(d) Using part (c), prove that probability of having any triple collisions is $1 - o(1)$ when $k = \omega(k^*)$. That is, if $k = ck^*$, the probability approaches 1 as c gets arbitrarily large.

Solution: Let X be the total number of triples with triple-collisions as defined in part (a) above. By part (b) we know that $\mathrm{E}[X] = \Theta(k^3/n^2)$, so we compute the variance

$$\begin{aligned}\mathrm{Var}(X) &= \sum_p (\mathrm{E}[X_p^2] - \mathrm{E}[X_p]^2) + \sum_{p \neq q} (\mathrm{E}[X_p X_q] - \mathrm{E}[X_p]\,\mathrm{E}[X_q]) \\ &= \sum_p \mathrm{Var}(X_p) \;+\; \sum_{p \neq q} (\mathrm{E}[X_p X_q] - \mathrm{E}[X_p]\,\mathrm{E}[X_q]).\end{aligned}$$

The first of these sums is easy to bound. For each triple p, we have

$$\mathrm{Var}(X_p) = \mathrm{E}[X_p^2] - \mathrm{E}[X_p]^2 = \frac{1}{n^2} - \frac{1}{n^4} \leq \frac{1}{n^2}$$

It follows that $\sum_p \mathrm{Var}(X_p) \leq \binom{k}{3}/n^2 = O(k^3/n^2)$.

To bound the other sum, consider two arbitrary triples p and q; there are three cases to consider:

i. Suppose $p \cap q = \emptyset$. In this case X_p and X_q are independent, which implies $\mathrm{E}[X_p X_q] - \mathrm{E}[X_p]\,\mathrm{E}[X_q] = 0$.

ii. Suppose $|p \cap q| = 1$. Then $X_p X_q = 1$ if and only if all five people in $p \cup q$ have the same birthday, so

$$\mathrm{E}[X_p X_q] - \mathrm{E}[X_p]\,\mathrm{E}[X_q] = n \cdot \frac{1}{n^5} - \frac{1}{n^2} \cdot \frac{1}{n^2} = 0.$$

iii. Finally, suppose $|p \cap q| = 2$. Then $X_p X_q = 1$ if and only if all four people in $p \cup q$ have the same birthday, so

$$\mathrm{E}[X_p X_q] - \mathrm{E}[X_p]\,\mathrm{E}[X_q] = n \cdot \frac{1}{n^4} - \frac{1}{n^2} \cdot \frac{1}{n^2} = \frac{1}{n^3} - \frac{1}{n^4} = O\left(\frac{1}{n^3}\right)$$

There are $O(k^4)$ pairs p, q such that $|p \cap q| = 2$—for each set of four people, there are $12 = O(1)$ ways to cover that set with two triples.

We conclude that

$$\sum_{p \neq q} (\mathrm{E}[X_p X_q] - \mathrm{E}[X_p]\,\mathrm{E}[X_q]) \;=\; O\left(\frac{k^4}{n^3}\right).$$

and therefore

$$\mathrm{Var}(X) \;=\; O(k^3/n^2 + k^4/n^3).$$

Thus, the inequality from part (c) implies

$$\Pr[X = 0] \leq \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2} = \frac{O(k^3/n^2 + k^4/n^3)}{\Theta(k^6/n^4)} = O\left(n^2/k^3 + n/k^2\right)$$

Finally, if $k = ck^* = cn^{2/3}$, we have

$$\Pr[X = 0] \;\leq\; O\left(n^2/k^3 + n/k^2\right) \;=\; O\left(1/c + 1/cn^{1/3}\right),$$

which approaches to 0 as $c \to \infty$. ■

5. A synchronous optical network (SONET) ring is an undirected cycle with n nodes, numbered consecutively from 0 to $n-1$; let e_i denote the edge between node i and node $(i+1) \bmod n$. Suppose we are given a (multi-)set C of m ordered pairs representing *calls*, where each ordered pair (i, j) represents a call originating at node i and destined for node j. Each call can be routed either clockwise or counterclockwise around the cycle. The *SONET ring loading* problem is to route the calls so as to minimize the maximum load on the network. For each index i, let L_i denote the number of calls that use edge e_i in either direction; our goal is to minimize $\max_i L_i$.

   (a) Write an LP relaxation for this problem, and use it to give a 2-approximation algorithm that *deterministically* rounds the LP solution.

Solution: For each call (i, j), let $i{\to}j$ and $j{\leftarrow}i$ respectively denote the clockwise and counterclockwise paths through the cycle from node i to node j. For each edge e_k, let C_k denote the set of possible call paths that could include e_k:

$$C_k := \left\{ i{\to}j \;\middle|\; (i,j) \in C \text{ and } e_k \in i{\to}j \right\} \cup \left\{ j{\leftarrow}i \;\middle|\; (i,j) \in C \text{ and } e_k \in j{\leftarrow}i \right\}$$

Our linear program has two variables $x_{i\to j}$ and $x_{j\leftarrow i}$ for each call (i, j), which *intuitively* indicate whether the call from i to j uses the corresponding path, along with a variable L indicating (an upper bound on) the maximum load. Here is the linear program relaxation:

$$\begin{aligned}
\text{minimize} \quad & L \\
\text{subject to} \quad & \sum_{i\to j \in C_k} x_{i\to j} + \sum_{j\leftarrow i \in C_k} x_{j\leftarrow i} - L \le 0 && \text{for each edge } e_k \\
& x_{i\to j} + x_{j\leftarrow i} = 1 && \text{for each call } (i,j) \in C \\
& x_{i\to j} \ge 0 && \text{for each call } (i,j) \in C \\
& x_{j\leftarrow i} \ge 0 && \text{for each call } (i,j) \in C
\end{aligned}$$

The linear program relaxation replaces each constraint of the form $x_\bullet \in \{0, 1\}$ with the inequality $x_\bullet \ge 0$. In the LP relaxation, $x_{i\to j}$ represents the *fraction* of the call from i to j that is routed clockwise around the cycle, and $x_{i\leftarrow j}$ represents the *fraction* that is routed counterclockwise.

Let x^* and L^* denote any optimal fractional solution to our linear program. Our approximation algorithm routes each call (i, j) clockwise if $x^*_{i\to j} \ge 1/2$ and counterclockwise otherwise; in other words, we route every call in the direction that receives the most weight in x^*, breaking ties clockwise. Rounding at most doubles the load on each edge e_k from each call (i, j), and thus at most doubles the load on each edge, and therefor at most doubles the maximum load.

More formally, we define an integral solution x', L' to our LP by setting

$$x'_{i\to j} = \left[x^*_{i\to j} \ge 1/2 \right] \quad \text{and} \quad x'_{j\leftarrow i} = \left[x^*_{i\to j} < 1/2 \right]$$

for each call (i, j), and defining

$$L' = \max_k \left(\sum_{i \to j \in C_k} x'_{i \to j} + \sum_{j \leftarrow i \in C_k} x'_{j \leftarrow i} \right)$$

Straightforward definition-chasing implies that this is a feasible integer solution to the LP. Moreover, because

$$x'_{i \to j} \le 2x^*_{i \to j} \quad \text{and} \quad x'_{j \leftarrow i} < 2x^*_{j \leftarrow i}$$

for each call (i, j), we have $L' \le 2L^*$. ■

(b) Now suppose we are also given a positive real *capacity* $c(e_i)$ for each edge e_i in the cycle and a positive real *demand* $d(i, j)$ for each call $(i, j) \in C$. A natural generalization of the SONET ring loading problem refines the load L_i to be the sum of the demands of all calls that use edge e_i divided by the capacity $c(e_i)$; the objective is still to minimize the maximum load $\max_i L_i$. (In the original problem, all capacities and demands are equal to 1.) Describe and analyze a deterministic 2-approximation algorithm for this more general problem.

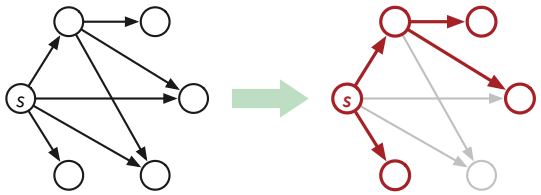
Solution: Again, our linear program has two variables $x_{i \to j}$ and $x_{j \leftarrow i}$ for each call (i, j), which now represent the fraction *of the demand* $d(i, j)$ that is routed clockwise or counterclockwise, respectively, and a variable L indicating (an upper bound on) the maximum load.

$$\begin{array}{rl} \text{minimize } L & \text{subject to} \\ \displaystyle\sum_{i \to j \in C_k} d(i,j) \cdot x_{i \to j} + \sum_{j \leftarrow i \in C_k} d(i,j) \cdot x_{j \leftarrow i} - c(e_k) \cdot L \le 0 & \text{for each } e_k \\ x_{i \to j} + x_{j \leftarrow i} = 1 & \text{for each } (i,j) \in C \\ x_{i \to j} \ge 0 & \text{for each } (i,j) \in C \\ x_{j \leftarrow i} \ge 0 & \text{for each } (i,j) \in C \end{array}$$

Again, our approximation algorithm routes each call (i, j) clockwise if $x^*_{i \to j} \ge 1/2$ and counterclockwise otherwise, where x^* is the optimal fractional solution. This rounding at most doubles the load on each edge and therefore at most doubles the maximum load. ■

6. 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.



Solution: This is a tuple selection / generalized matching problem; we are attempting to *assign* a parent to every vertex except s, so that each vertex is the parent of at most two other vertices.

We construct a flow network $G' = (V', E')$ as follows:

- G' has a source vertex s', a vertex v_{child} for every vertex $v \neq s$ in G, a vertex v_{parent} for every vertex v in G, and a target vertex t'. Altogether, G' has $2V + 1$ vertices.
- G' has the following capacitated edges:
  - An edge $s' \to v_{\text{child}}$ with capacity 1, for every vertex $v \neq s$ in G.
  - An edge $v_{\text{child}} \to u_{\text{parent}}$, with capacity 1, for every edge $u \to v$ in G,
  - An edge $u_{\text{parent}} \to t'$ with capacity 2, for every vertex u in G.

  Altogether, G' has $2V + E - 1 = O(E)$ edges. (If $E < V - 1$, then G is disconnected, so we can immediately return FALSE.)

Finally, we compute a maximum flow from s' to t' in G', and then return TRUE if and only if every edge out of s' is saturated. If every edge leaving s is saturated, then the $v_{\text{child}} \to u_{\text{parent}}$ with flow value 1 correspond to edges $u \to v$ in a binary spanning tree of G. Conversely, given any binary spanning tree T of G, we can construct a feasible flow in G' that saturates every edge leaving s' by sending one unit of flow along the path $s' \to v_{\text{child}} \to u_{\text{parent}} \to t'$, for every edge $u \to v$ in T.

We can easily construct G' in $O(V + E)$ time by brute force. The maximum flow value in G' is at most $V - 1$, so we can compute the maximum flow in $O(V'E') = O(VE)$ time using Ford-Fulkerson. Overall, our algorithm runs in $\boldsymbol{O(VE)}$ ***time***. ■