Please solve Problem 1, and **any two** of the remaining problems. Note that problems 2 & 3 naturally go together, as do 4 & 5. If you want to solve 3 or 5 without solving 2 or 4, you can assume any results you need from the other problem.

1. **Bottleneck Paths.** Given a graph with edge weights w_e, the bottleneck of a path P from s to t is the edge with the smallest weight. The goal of the bottleneck-path problem is to find a path P such that the bottleneck is as large as possible.

   (a) (Don't hand in) Show how to modify Dijkstra's algorithm to compute the single-source bottleneck-path problem on directed graphs in time $O(m + n \log n)$.

   (b) Given an undirected graph G and a pair s, t, show how to solve the s-t bottleneck-path problem on G in deterministic $O(m + n)$ time. (Hint: Use medians.)

   (c) Given an undirected graph G, show how to solve the *all-pairs* bottleneck-path problem on G in near-linear deterministic time or expected linear randomized time.

2. **Ackermann Shortcuts.** You are given a directed path $\langle v_0, v_1, \ldots, v_n \rangle$ with n arcs, all arcs pointing from left to right. You are allowed to add m more arcs (also going from left to right), which should ensure that for any $i < j$, you can go from v_i to v_j using at most k arcs. (If a set of arcs achieves this property, we say the resulting graph has "di-diameter" k.) The goal is to explore the trade-off between the di-diameter k and the number of edges needed to achieve this di-diameter.

   (a) (Don't hand in) If $k = 1$, observe that you need to add in $m_1(n) := \binom{n+1}{2} - n$ arcs.

   (b) Give a solution where adding in $m_2(n) = n \log_2 n$ arcs guarantees a di-diameter of 2. (Hint: divide and conquer.)

   (c) Suppose you can achieve di-diameter k using $m_k(n)$ edges for some even value $k \geq 2$. Show that for any $1 < t < n$, you can get
   $$m_{k+2}(n) \leq 2n + m_k(n/t) + (n/t) \cdot m_{k+2}(t).$$

   (d) Use the above recurrence to show that $m_4(n) \leq 3n \log^* n$.

   (e) Recall that for a non-decreasing function g such that $g(x) < x$, we define
   $$g^*(x) = \min\{t \mid g^{(t)}(x) < 1\}.$$
   Show that if
   $$m_\ell(n) \leq (2\ell - 1) \cdot n \cdot g(n)$$
   then
   $$m_{\ell+2}(n) \leq (2\ell + 1) \cdot n \cdot g^*(n).$$

   (f) define $\alpha(n) = \min\{k \mid \log^{**\cdots*}(n) \leq 2\}$, where the number of stars is k. Show that you can achieve di-diameter $\alpha(n)$ by adding at most $O(n\alpha(n))$ arcs.

3. **LCAs, Semigroups and Partial Sums.** A semigroup is a set S of elements with an *associative* binary operation $\circ : S \times S \to S$.

   (a) (Don't hand in) Given any set S with a total order defined on it, show that $(S, \min)$, $(S, \max)$ are semigroups.

(b) Suppose you are given an array $A[1..n]$ where each position contains an element from a semigroup. You want to construct a data structure that does some preprocessing and then answers queries of the form: *given* $i < j$, *what is* $A[i] \circ A[i+1] \circ \ldots \circ A[j]$*?* (These are called partial sum queries over semigroups.

Show how to use the construction of good short-cutting schemes from the previous problem to give a solution that has $O(n\alpha)$ preprocessing time and that answers partial sum-queries in $O(\alpha)$ time per query. (You should not assume that the operation is commutative.)

(c) Given a tree $T = (V, E)$ rooted at $r \in V$, show how you can use the data structure above to quickly answer queries of the form: *given* $x, y \in T$, *which node is the least common ancestor of* (x, y) *in* T*?* (Hint: Euler tour.)

4. **The Boolean Product Witness Matrix problem.** The input to the BPWM problem consists of two $n \times n$ Boolean matrices A, B. The output is an integer valued matrix W such that for any integer $k > 0$,

$$W_{ij} = k \implies A_{ik} = 1 \text{ and } B_{kj} = 1.$$

I.e., W_{ij} tells us which which entry in the i^{th} row of A, and in the j^{th} column of B would give us a 1 in $(AB)_{ij}$.

(a) *The Single Witness case.* Suppose for some i, j, there is a single value k such that $A_{ik} = B_{kj} = 1$. For each $t \in 1 \ldots n$, multiply all entries of the t^{th} column of A by t; call the resulting matrix $\hat{A}$. Show that $(\hat{A}B)_{ij}$ contains the witness for the pair i, j. Conclude that for all pairs i, j which have a single witness, this witness can be found using a single matrix multiply.

(b) *Multiple Witnesses.* Suppose for some i, j, the number of witnesses lies in some range $[2^s, 2^{s+1})$. Consider a uniformly random subset $I \subseteq [n]$ of size $n/2^s$, and let A^I be the matrix formed by choosing the columns of A whose indices lie in I. Similarly let B^I be the matrix formed by B whose *rows* lie in I.

i. Show that for i, j, with constant probability there is a unique index k' such that $A_{ik'} = 1$ and $B_{k'j} = 1$.

ii. By using the idea from the previous part, give an algorithm that succeeds in finding the witness for any such pair i, j (that has about 2^s witnesses) with constant probability.

iii. Suppose we know how to multiply two square $N \times N$ matrices in N^ω time for all N. How much time would it take to multiply an $n \times k$ by $k \times n$ matrix? Hence, how much time would the above witness-finding step take?

iv. Repeat the process $\Theta(\log n)$ times for this value of s. Show that with high probability, we would have found witnesses for all pairs i, j that have $\approx 2^s$ witnesses.

(c) Using the above two parts, and the fact that the number of witnesses for any i, j pair lies between 1 and n, give an algorithm that solves the BPWM problem in expected $O(n^\omega \log n)$ time. You may assume that $\omega > 2$ for this problem.

5. **Seidel's Algorithm: Finding Paths.** In this problem we will develop an algorithm to find (an implicit) representation of all-pairs shortest paths in unweighted undirected graphs.

Since there could be graphs such that the total lengths of the $\binom{n}{2}$ shortest paths is $\Omega(n^3)$, and we want to run in $O(M(n) \operatorname{poly} \log n) = o(n^3)$ time, we want merely want to build a successor matrix S, such that $S_{ij} = k$ if a i-j shortest path is obtained by the arc (i, k) concatenated with the k-j shortest path. We assume that we have already used the UUAPSP algorithm to compute the shortest-path distances (d_{ij}) in G.

(a) Suppose $d_{ij} = r$. Show that if we set A to be adjacency matrix for G, and B to be the matrix $B_{pq} = \mathbf{1}_{(d_{pq}=r-1)}$, and $W \leftarrow BPWM(A, B)$, then W_{ij} is indeed the next hop in the shortest-path from i to j.

(b) If the largest distance between any two nodes in G is Δ, show how $\Delta - 1$ BPWM computation suffice to compute the successor matrix.

(c) Now to do better. Suppose $d_{ij} = 1 \pmod 3$. Show that if we set A to be adjacency matrix for G, and B to be the matrix $B_{pq} = \mathbf{1}_{(d_{pq}=0 \pmod 3)}$, and $W \leftarrow BPWM(A, B)$, then W_{ij} is still the next hop in the shortest-path from i to j.

(d) Use this idea to show that 3 BPWM computations suffice to compute the successor matrix.

Hence if each BPWM computation takes $O(n^{\omega} \log n)$ time (as we showed in Problem #4, this problem has shown that we can compute the successor matrix in aymptotically the same amount of time as the time to compute the shortest path distances.