Problem Set 2 — Solution

Problem 1

Let A be the $n \times n$ adjacency matrix of G: i.e., $A_{i,j} = 1$ if there's an edge between the vertices i and j. Observe that $A^2_{i,j} = \sum_{i=1}^{n} A_{i,k}A_{k,j}$ counts exactly the number of paths of length 2 between i and j. Similarly, $(A^3)_{i,j} = \sum_{k,\ell=1}^{n} A_{i,k}A_{k,\ell}A_{\ell,j}$ counts the number of paths of length 3 between i and j.
When looking at $A_{i,i}$, i.e., paths of length 3 that begin and end at i, such a path must be a triangle that contains the vertex i since G has no self loops. However, for a triangle i, j, k there are two possible paths that begin and end at i, namely, $i \to j \to k$ and $i \to k \to j$. Further, the same triangle is similarly counted twice in $A_{j,j}$ and $A_{k,k}$.
Thus, our algorithm will first compute A^3 and then output $(\sum_{i=1}^{n}(A^3)_{i,i})/6$ (as explained above, in the sum $\sum_{i=1}^{n}(A^3)_{i,i}$ every triangle is counted exactly 6 times).
Computing A^3 can be done using two $n \times n$ matrix multiplication, in time $O(n^{2.81\cdots})$, as shown in class. The second part can be done in time $O(n)$.

Problem 2

Let $v = (v_0, \ldots, v_{n-1})$. Associate with A the polynomial $a(x) = \sum_{k=0}^{2n-1} a_k x^k$ and with v the polynomial $b(x) = \sum_{k=0}^{n-1} v_k x^k$.
We now compute the product $a(x) \cdot b(x)$. To make the notation easier to follow, we may write $b(x) = \sum_{k=0}^{2n-1} v_k x^k$ with $v_n = v_{n+1} = \cdots = v_{2n-2} = 0$.
Note that for every $i \leq 0 \leq n-1$, the coefficient of x^{n-1+i} in $a(x) \cdot b(x)$ equals

$$a(x) \cdot b(x) = \sum_{k=0}^{n-1+i} v_k \cdot a_{(n-1+i)-k} = \sum_{k=0}^{n} v_k \cdot a_{(n-1+i)-k}$$

where the last inequality follows from the fact that $v_j = 0$ for $j > n$. Note that the last expression is exactly the k-th coordinate of $(Av)_k$.
Thus, we can compute $(Av)_k$ by multiplying the two polynomials $a(x)$ and $b(x)$ using FFT in time $O(n \log n)$ and reading off the coordinates from the coefficients of $a(x) \cdot b(x)$.

Problem 3

Part a

Let $f(x) = \sum_{i=1}^{n} p_i x^i$.

Let Y_i by a random variable that denotes the outcome of the i-th roll of the dice. Since the rolls are independent,

$$\Pr[Y_1 + Y_2 = s] = \sum_{i,j:i+j=s} \Pr[Y_1 = i, Y_2 = j] = \sum_{i,j:i+j=s} \Pr[Y_1 = i] \cdot \Pr[Y_2 = j] = \sum_{i,j:i+j=s} p_i p_j.$$

Thus, this expression exactly equals the coefficient of x^{i+j} in the polynomial f^2. Thus, all these probabilities can be computed in time $O(n \log n)$ by computing $f \cdot f = f^2$ using FFT.

Part b

As in part (a), in this case the required probabilities are the coefficients of the polynomial f^{2^k}. We'll count how many arithmetic operations are needed to compute f^{2^k}. Note that by using repeated squaring we can compute $f, f^2, (f^2)^2 = f^4, f^8, \ldots, f^{2^k}$ using a total of k polynomial multiplications. However, the the degrees of the polynomials being multiplied increase at each step, so we need to be a bit careful when analyzing the total number of operations.
Let $M(d) = O(d \log d)$ denote the total number of operations needed to multiply two degree-d polynomials. It's straightforward to verify that $M(d_1 + d_2) \le M(d_1) + M(d_2)$. In our case, $\deg(f^{2^i}) = n \cdot 2^i$ and therefore the total number of operations is

$$\sum_{i=1}^{k} M(2^i n) \le M\left(\sum_{i=1}^{k} 2^i n\right) \le M(2^{k+1}n) = O(2^{k+1} n \log(2^{k+1} n)) = O(2^k n(\log n + k)).$$

Problem 4

Part a

Let $B = A^T A$. We will show that for every $v \in \mathbb{R}^m$, $Av = 0$ if and only if $Bv = 0$. This implies that $\ker(A) = \ker(B)$ and thus $\operatorname{rank}(A) = \operatorname{rank}(B)$.
In one direction, if $Av = 0$ then clearly $Bv = (A^T A)v = A^T(Av) = 0$. In the other direction, suppose $Bv = 0$. Multiply by v^T on the left to get that

$$0 = v^T(Bv) = v^T A^T A v = (v^T A^T)(Av) = \langle Av, Av \rangle = \|Av\|_2^2$$

and thus $Av = 0$.

Part b

As in part (a), we will use the fact that if $\dim \ker(B) = k$ then $\operatorname{rank}(B) = m - k$. The rank of the kernel is exactly the geometric multiplicity of the eigenvalue 0 of B. However, since B is symmetric and hence diagonalizable, this also equals the algebraic multiplicity of 0, which is exactly the number of λ_i's that equal 0.

Part c

By part (b), 0 is a root of p with multiplicity k, which implies that x^k divides $p(x)$. Further, x^k is the largest power of x dividing $p(x)$ (as otherwise the multiplicity of 0 would have been higher). Thus, $p(x) = x^k \cdot q(x)$ where q is a polynomial with non-zero constant term, which implies the claim.

Therefore, our algorithm for computing $\text{rank}(A)$ will compute $B = A^T A$ in parallel time $O(\log n)$, compute the characteristic polynomial of B in parallel time $O(\log^2 m) = O(\log^2 n)$ as seen in class, and then find the smallest k such that the coefficient of x^k is non-zero using standard comparison-tree in parallel time $O(\log m) = O(\log n)$.