1.

Invertibilní (regulární) matice musí mít lineárně nezávislé řádky. To znamená, že žádný následující řádek nelze vyjádřit jako lineární kombinaci předchozích. Pokud máme tedy matici $n \times n$ nad $\mathbb{Z}_p$, tak vektory jsou z prostoru $\mathbb{Z}_p^n$.
První vektor tedy musí být lineárně nezávislý, což v tomto případě znamená nenulový. Počet možností jak vybrat 1. vektor tedy je $p^n - 1$, protože různých vektorů je p^n a neuvažujeme nulový. Obdobně pro druhý. Druhý nesmí být lineární kombinací prvního vektoru. Lineárních kombinací prvního vektoru je právě p. Toto plyne z definice lineární kombinace (všechny násobky prvního vektoru skalárem, kterých je p). Pro druhý vektor máme tedy $p^n - p$ možností.
Pro i-tý vektor máme tedy právě $p^n - p^{i-1}$ možností, protože počet lineárních kombinací předchozích vektorů je právě p^{i-1} (pro první mám p možností, pro druhý také p atd. a předchozích vektorů je $i - 1$) viz definice lineární kombinace $\rightarrow \{a_1 \cdot v_1 + a_2 \cdot v_2 + \cdots + a_{i-1} \cdot v_{i-1} : v_i \in \mathbb{Z}_p^n, a_i \in \mathbb{Z}_p, i \in \{1, \ldots, i-1\}\}$. Celkově tedy platí, že počet invertibilních matic je $\prod_{i=0}^{n-1}(p^n - p^i)$.

2.

Celkově tedy je 10^8 různých kombinací. Střední hodnota je $EX = \sum_{i=1}^{10^8} p_i \cdot i$, kde X je náhodná veličina, která značí pravděpodobnost, že v i-tý vyzkoušený PIN je správný. Tato pravděpodobnost je pro každý PIN stejná, tedy $\forall i : p_i = \frac{1}{10^8}$. Střední hodnota tedy je $EX = p \cdot \sum_{i=1}^{10^8} i \overset{\text{dosazení}}{=} \frac{1}{10^8} \cdot \sum_{i=1}^{10^8} i \overset{\text{vypočet sumy}}{=} \frac{1}{10^8} \cdot \frac{10^8 \cdot (10^8+1)}{2} = \frac{10^8+1}{2} = 50000000.5 \approx 5 \cdot 10^7$ pokusů.

Z algoritmu plyne, že 8. číslice je pevně určena těmi předchozími. Celkový počet různých kombinací klesl na 10^7 (stačí vyzkoušet všechny sedmimístné a osmou doplnit jako danou sumu).
Z algoritmu také plyne, že potřebujeme vyzkoušet pouze 10^4 kombinací. Jelikož můžeme zkoušet PINy od 0000 0000 do 9999 9999. Ale budeme inkrementovat v každé čtveřici zároveň, nejdříve PIN 0000 0000 poté 0001 0001 atd. Takových možností je právě 10^4, tedy nám stačí vyzkoušet maximálně 10^4 PINů a jistě se dostaneme do pračky.
Poslední test nám moc nepomůže (v nejhorším případě). V nejhorším případě totiž pračka může mít PIN 9999 9993. K jeho nalezení potřebujeme

$9994 + 1$ pokusů. 9994 proto, že poslední čtveřici zjistíme na tolik pokusů. Pokud již známe poslední číslo 3, víme, že součet první čtveřice modulo 10 musí dát 3. Neprozkoumali jsme ještě 6 zbývající čtveřic, ale pouze 1 z nich dává 3 v součtu modulo 10.
Toto je nejhorší případ, jelikož v ostatních situacích můžeme zjistit PIN na méně pokusů.
Pokud zjistíme 2. čtveřici nejdříve, zbylé možnosti pro 1. čtveřici se nám zmenší na $\frac{1}{10}$, protože zhruba jedna desetina dává v součtu požadované číslo modulo 10.
Naopak, jestliže zjistíme 1. čtveřici jako první, tak nám stačí zkoušet jen všechny zbývající třetice (maximálně 10^3), jelikož poslední číslice je určena těmi sedmi předchozími.
V nejhorším případě tedy útočník musí vyzkoušet 9995 PINů.

3. Text je očividně z názvu přístroje zašifrován Vigenèrovou šifrou. Nejprve tedy potřebujeme zjistit délku klíče, to uděláme pomocí testu indexu koincidence pro různé délky klíče. Při délce klíče 7 nám indexy koincidence "rozsekaných" textů dávají vysoké hodnoty odpovídající anglickému textu $(0.089, 0.068, 0.054, 0.079, 0.069, 0.066, 0.055)$. Klíč je tedy pravděpodobně délky 7, pokud je to pravda, tak můžeme šifru vyluštit pomocí frekvenční analýzy všech 7 textů, které nám vznikly po rozsekání (například 1. text je složen z 1. , 8. , 15. ... písmena).
Nejčastější písmeno v prvním textu je G, tudíž je velice pravděpodobné že G bude zašifrované E nebo T. Z toho zjistíme klíč jednoduše vyxorováním písmena z ciphertextu s jeho kandidátem. Takto stačí postupovat u dalších šesti textů dokud dešifrovaný text nezačne dávat smysl. Lze také například využít toho, že klíč musí být špatný, jestliže se nám v dešifrovaném plaintextu objeví písmeno Z, které by tam být nemělo. Dále lze hádat, podle toho, jaká slova se tam asi objevují (jako AND, THE apod.). Po delší době se dostaneme ke klíči "PYTONIK", který nám text dešifruje na text z Monty Pythona.

4. Četnost znaků (nejčastější písmena jsou E, T, I, H, O) v prvním textu odpovídá četnosti písmen v anglických textech. Tudíž se dá předpokládat, že první text byl zašifrován transpoziční šifrou, která pouze permutuje písmena a nemění jejich četnost.

Ve třetím textu četnost znaků odpovídá četnosti znaků z prvního textu, akorát je u jiných písmen. Takže text byl pravděpodobně zašifrován substituční šifrou, která tyto četnost nemění, ale pouze u nich vymění písmena.
Na druhý text nezbývá nic jiného než Hillova šifra. Také to ale plyne z toho, že četnosti znaků jsou různé než u textů 1 a 3.

5.

Index vzájemné koincidence dvou textů a, b (délky n_a, n_b) napsaných v abecedě $\mathbb{A}$ se z definice spočítá jako $I_c(a,b) = \frac{\sum_{i\in\mathbb{A}} f_i^a \cdot f_i^b}{n_a \cdot n_b}$, kde f_i^a, f_i^b jsou četnosti písmena i v textech a, b.
Tedy pro texty p a $e_k(p)$ (oba mají stejnou délku n, jelikož Caesarova šifra nemění délku textu) platí

$$I_c(p, e_k(p)) = \frac{\sum\limits_{i\in\mathbb{A}} f_i^p \cdot f_i^{e_k(p)}}{n^2} \overset{\text{Cauchyova nerovnost}}{\leq} \frac{\sqrt{\sum\limits_{i\in\mathbb{A}}(f_i^p)^2 \cdot \sum\limits_{i\in\mathbb{A}}(f_i^{e_k(p)})^2}}{n^2}$$

Sumy $\sum\limits_{i\in\mathbb{A}}(f_i^p)^2$, $\sum\limits_{i\in\mathbb{A}}(f_i^{e_k(p)})^2$ jsou ale stejné, jelikož $\forall i \in \mathbb{A}\, \exists j \in \mathbb{A} : f_i^p = f_j^{e_k(p)}$. To plyne z toho, že Caesarova šifra zachovává četnosti písmen (tedy i index koincidence). Tedy:

$$\frac{\sqrt{\sum\limits_{i\in\mathbb{A}}(f_i^p)^2 \cdot \sum\limits_{i\in\mathbb{A}}(f_i^{e_k(p)})^2}}{n^2} = \frac{\sqrt{(\sum\limits_{i\in\mathbb{A}}(f_i^p)^2)^2}}{n^2} = \frac{\sum\limits_{i\in\mathbb{A}}(f_i^p)^2}{n^2} = I_c(p,p)$$

Tedy nerovnost $I_c(p, e_k(p)) \leq I_c(p,p)$ nezávisí na klíči k, ale pouze na vlastnostech šifry.

6.

Oba texty jsou zašifrované Vernamovou šifrou pomocí stejného klíče. Díky komutativitě operace xor (kterou šifra používá) a tomu, že např. $x \oplus k \oplus k = x$, můžeme vyxorovat oba ciphertexty a dostaneme xor plaintextů (protože se klíč vyxoruje sám se sebou).
Díky tomu, že víme, že v druhém textu je někde slovo "JABBERWOCKY", tak nám xorovat xor plaintextů slovem "JABBERWOCKY" na různých pozicích, dokud výsledek xoru není něco jako anglické slovo. Zjistíme tedy, že "JABBERWOCKY" začíná na 15. pozici a v prvním plaintextu tomu odpovídá slovo "BLETCHLEYPA". Díky Googlu

se dá zjistit, že po "BLETCHLEY" pravděpodobně následuje slovo "PARK". To nám zase dá rozumná písmena v 2ṫextu, který již má tvar "JABBERWOCKYWE". Po dalším zkoušení, lze například zjistit, že po "WE" následuje "HAVE" nebo že před "BLETCHLEYPARK" předchází "INTHE".
První text tedy pravděpodobně obsahuje:
INTHEBLETCHLEYPARKSTOP
a druhý text:
HATEDJABBERWOCKYWEHAVE
. Díky Googlu jsem ale také narazil na zajímavou možnost, že za slovem "JABBERWOCKY" následuje slovo "KOAN", které má v podstatě stejný význam (viz https://en.wikipedia.org/wiki/Nonsense). To by ale odpovídalo prvnímu textu "BLETCHLEYPAVING", což je méně pravděpodobné než "PARK".