Στις ακόλουθες φωτογραφίες υπάρχουν οι λύσεις όλων των ασκήσεων εκτός από ένα τμήμα της 7[ης], η οποία εμπεριέχεται στο τμήμα κώδικα που λέγεται 7.py αλλά η επεξήγησή της βρίσκεται παρακάτω.

1) i) $\sqrt{2^n} = 2^{\frac{n}{2}}$, δηλαδή εκθετικός

ii) έχουμε τα bits του κόστους $c2(\log c2)$, άρα ο όρος που μεταβάλλει τη ταχύτητα του αλγορίθμου είναι ο n^2. Άρα, πολυωνυμικός.

iii) Εδώ έχουμε το ίδιο το κόστος ($\min(i)$), μαζί με το πολυωνυμικό n^3. Συνεπώς, ψευδοπολυωνυμικός.

iv) Ο χρόνος είναι πολυωνυμικός, καθώς για n αντικείμενα, έχουμε 2^n διαφορετικά υποσύνολα. Αν $2^n = N$, τότε, για N δεδομένα τρέχει σε $O(N)$.

v) Έχουμε τον λογάριθμο του αθροίσματος των ως γινόμενο με το n^2. Άρα, πολυωνυμικός.

2) i) Για $k=1$, έχουμε: $F_{n+1} = F_{n+1} \cdot F_1 + F_n \cdot F_0$ | $\Rightarrow$
Όμως, $F_1 = 1$, $F_0 = 0$ |

$F_{n+1} = F_{n+1} \cdot 1 + F_n \cdot 0 \Leftrightarrow F_{n+1} = F_{n+1}$, που ισχύει.

Θα χρησιμοποιήσουμε τη μέθοδο της επαγωγής για να αποδείξουμε ότι η σχέση μας ισχύει και για $k>1$.

Έστω ότι η σχέση μας ισχύει για οποιοδήποτε $k = h > 1$. Έτσι, έχουμε: $F_{n+(h+1)} = F_{n+1} \cdot F_{h+1} + F_n \cdot F_h$ (α)

Εξ' ορισμού, λόγω της ακολουθίας ισχύει:
$F_{n+(h+1)} = F_{n+h+1} = F_{n+h} + F_{n+h-1}$.
Άρα, $F_{n+(h+1)} = F_{n+h} + F_{n+h-1}$. (β)

Λόγω της υπόθεσης ⓐ, πρέπει:

$$\begin{cases} F_{n+h} = F_{n+1} \cdot F_h + F_n \cdot F_{h-1} \\ F_{n+h-1} = F_{n+1} \cdot F_{h-1} + F_n \cdot F_{h-2} \end{cases} \quad (\gamma)$$

, όμως λόγω της ⓑ γίνεται:

$$F_{n+(h+1)} = \underbrace{(F_{n+1} \cdot F_h + F_n \cdot F_{h-1})}_{F_{n+h},\ \text{λόγω } (\gamma)} + \underbrace{(F_{n+1} \cdot F_{h-1} + F_n \cdot F_{h-2})}_{F_{n+h-1},\ \text{λόγω } (\gamma)} \Leftrightarrow$$

$$F_{n+(h+1)} = F_{n+1}(F_h + F_{h-1}) + F_n(F_{h-1} + F_{h-2})$$

Παράλληλα, εξ' ορισμού, λόγω της ακολουθίας ισχύει:

$$\begin{cases} F_h + F_{h-1} = F_{h+1} \\ F_{h-1} + F_{h-2} = F_h \end{cases} \Rightarrow$$

$F_{n+(h+1)} = F_{n+1} \cdot F_{h+1} + F_n \cdot F_h$, άρα αποδείχθη.

$F_{n+2} = F_{n+1} F_2 + F_n F_1 = F_{n+1} \cdot 1 + F_n \cdot 1 = F_{n+1} + F_n$, που ισχύει

ii) $x > 35$, $gcd(x, 35) = 1$.

Είναι $35 = 5 \cdot 7$ και ισχύει $\begin{cases} gcd(x,5) = 1 \\ gcd(x,7) = 1 \end{cases}$

Βάσει του μικρού θεωρήματος του Fermat:

- $gcd(x,7) = 1 \Rightarrow x^{7-1} \equiv 1 \pmod 7 \Leftrightarrow x^6 \equiv 1 \pmod 7$ ⓐ
- $gcd(x,5) = 1 \Rightarrow x^{5-1} \equiv 1 \pmod 5 \Leftrightarrow x^4 \equiv 1 \pmod 5$ ⓑ

Για ⓐ: $x^6 \equiv 1 \pmod 7 \Rightarrow (x^6)^2 \equiv 1^2 \pmod 7 \Rightarrow x^{12} \equiv 1 \pmod 7$

Για ⓑ: $x^4 \equiv 1 \pmod 5 \Rightarrow (x^4)^3 \equiv 1^3 \pmod 5 \Rightarrow x^{12} \equiv 1 \pmod 5$

Άρα: $x^{12} \equiv 1 (mod\, 7)$ | Βάσει Chinese Remainder
$x^{12} \equiv 1 (mod\, 5)$ | => Theorem:
και $gcd(5,7) = 1$ | $x^{12} \equiv 1 (mod\, 35)$

3) i) Είναι: $\varphi(243)$, $243 = 3 \cdot 81 = 3 \cdot 3^4 = 3^5$

Ισχύει ότι: $\varphi(p^k) = p^k - p^{k-1}$, αφού 3: πρώτος
$\Rightarrow \varphi(243) = \varphi(3^5) = 3^5 - 3^4 = 243 - 81 = 162$

Άρα, $\varphi(243) = 162$

ii)α) $7 \cdot 3^{58} (mod\, 19)$, 19: πρώτος και $gcd(3,19) = 1$.

Βάσει του μικρού Θεωρήματος του Fermat:
$3^{19-1} \equiv 1 (mod\, 19) \Rightarrow 3^{18} \equiv 1 (mod\, 19)$
Όμως, $58 = 3 \cdot 18 + 4$ | =>

$3^{58} = 3^{3 \cdot 18 + 4} = (3^{18})^3 \cdot 3^4$
$3^{18} \equiv 1 (mod\, 19)$ | $\Rightarrow 3^{58} \equiv 3^4 (mod\, 19)$, $3^4 = 81$

$\Rightarrow 3^{58} \equiv 81 (mod\, 19) \equiv 5 (mod\, 19)$
είναι $7 \cdot 3^{58} (mod\, 19)$ | $\Rightarrow 35 (mod\, 19) =$
$= 16$

β) 60: δεν είναι πρώτος, βάσει Euler:
$(3 \cdot 7^{17} + 11^{33} + 3 \cdot 13^{49}) (mod\, 60)$

$60 = 2^2 \cdot 3 \cdot 5$ $\varphi(60) = 60(1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{5}) =$
$= 60 \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{4}{5} = \frac{480}{30} = 16$

Άρα, ∀ αριθμό k: σχετικά πρώτος του 60, ισχύει:
$k^{16} \equiv 1 (mod\, 60)$ ①

$gcd(7,60) = gcd(11,60) = gcd(13,60) = 1$

Είναι: • $7^{17} = 7 \cdot 7^{16} \equiv 7 (mod 60) \Rightarrow 3 \cdot 7 = 21$,

από την ① $21 \equiv 21 (mod 60)$

• $11^{33} = 11 \cdot 11^{16} \cdot 11^{16} \equiv 11 (mod 60)$

από την ①

• $13^{49} = 13 \cdot 13^{16} \cdot 13^{16} \cdot 13^{16} \equiv 13 (mod 60) \Rightarrow 3 \cdot 13 = 49$,

από την ① $49 \equiv 49 (mod 60)$

Άρα, έχουμε: $21 + 11 + 39 (mod 60) = 71 (mod 60) =$
$= 11 (mod 60)$

(ii) Για να το αποδείξουμε, θα εφαρμόσουμε το τέστ του Fermat, άρα:

Γίνεται η πράξη $a^{n-1} mod n$, η οποία μας δίνει ως αποτέλεσμα:

• είτε $a^{n-1} \equiv 1 (mod n)$, όπου το n λανθασμένα παρουσιάζεται ως πρώτος αριθμός

• είτε $a^{n-1} \equiv 1 (mod n)$, όπου το n σωστά παρουσιάζεται ως σύνθετος αριθμός

Επίσης, το n δεν είναι αριθμός Carmichael, άρα $\exists$ τουλάχιστον ένα a τέτοιο ώστε
$a^{n-1} \not\equiv 1 (mod n)$

Ταυτόχρονα, οι $a \in \mathbb{N}$, οι οποίοι $gcd(a, n) = 1 \in \varphi(n)$, όπου η $\varphi(n)$ η συνάρτηση του Euler.

Αν $S \subseteq [1, n]$ το σύνολο των αριθμών, όπου $a^{n-1} \not\equiv 1 (\bmod n)$, τότε

$S = \{a \in \{1, 2, \dots, n-1\} \mid gcd(a, n) = 1$ και $a^{n-1} \not\equiv 1 (\bmod n)\}$

Και

$S' = T = \{a \in \{1, 2, \dots, n-1\} \mid gcd(a, n) = 1$ και $a^{n-1} \equiv 1 (\bmod n)\}$

Επίσης: $\{ 1 \in T \; (1^{n-1} = 1)$

το T είναι κλειστό ως προς τον πολλαπλασιασμό (αν $a, b \in T$, τότε $(ab)^{n-1} \equiv a^{n-1} b^{n-1} \equiv 1 . 1 \equiv 1$)

Άρα βάσει της Θεωρίας Ομάδων, το μέγεθος του συνόλου T διαιρεί το $\varphi(n)$. Άρα $|T| \leq \frac{\varphi(n)}{2}$ και ως επακόλουθο: $|S| = \varphi(n) - |T| \geq \frac{\varphi(n)}{2}$

Έτσι, το τεστ του Fermat θα μας δώσει θετικό αποτέλεσμα, δηλαδή ότι ο n είναι σύνθετος, αν $a \in S$.

Όμως $|S| \geq \frac{\varphi(n)}{2}$, και συνεπώς η πιθανότητα να δωθεί σωστό αποτέλεσμα είναι:

$P(a \in S) = \frac{|S|}{\varphi(n)} \geq \frac{\frac{\varphi(n)}{2}}{\varphi(n)} = \frac{1}{2}$, δηλαδή $P(a \in S) \geq \frac{1}{2}$, άρα

αποδείχθη.

4) $c=10,\ e=13,\ n=35 \Rightarrow M=c^d \pmod n$

Είναι: $\varphi(n)=\varphi(35) \;\Big|\; =\varphi(35)=4\cdot 6=24$
$\quad 35=5\cdot 7$

• $e\cdot d \equiv 1 \pmod{24} \Rightarrow 13d \equiv 1 \pmod{24} \Rightarrow d=13$
$\Rightarrow M^{13} \equiv 10 \pmod{35}$

Άρα, μέσω της μεθόδου επαναλαμβανόμενου τετραγωνισμού, έχουμε: $10^{13}=10^8\cdot 10^4\cdot 10$,

και: $10^2 \bmod 35 = 100 \bmod 35 = 30$
• $10^4 \bmod 35 = (10^2)^2 \bmod 35 = 30^2 \bmod 35 = 900 \bmod 35 = 25$
• $10^8 \bmod 35 = (10^4)^2 \bmod 35 = 25^2 \bmod 35 = 625 \bmod 35 =$
$= 30 \pmod{35}$

$\Rightarrow M = 10^{13} = 10^8\cdot 10^4\cdot 10^1 \pmod{35} \equiv 30\cdot 25\cdot 10 \pmod{35}$
$\equiv 7500 \pmod{35} \equiv 10 \pmod{35}$

$\Rightarrow M=10$

Οι υπολογισμοί:

$$\begin{array}{r|l} 900 & 35 \\ \hline -70 & 25 \\ 200 & \\ -175 & \\ \hline 25 & \end{array} \qquad \begin{array}{r|l} 625 & 35 \\ \hline -35 & 17 \\ 275 & \\ -245 & \\ \hline 30 & \end{array} \qquad \begin{array}{r|l} 7500 & 35 \\ \hline -70 & 214 \\ 50 & \\ -35 & \\ 150 & \\ -140 & \\ \hline 10 & \end{array}$$

5) i). • Η καλύτερη περίπτωση συμβαίνει όταν ισχύει η συνθήκη if $(h > \sqrt{n})$, επομένως γίνεται return $A[n]$, πράξη που αποτελεί $\Theta(1)$.

• Η χειρότερη περίπτωση συμβαίνει όταν ισχύει η συνθήκη else $(h \leq \sqrt{n})$, επομένως εκτελείται ο βρόχος for $i=1$ έως n, όπου γίνεται μια πρόσθεση. Άρα, έχουμε $O(n)$.

• Η κατανομή είναι ομοιόμορφη, άρα η πιθανότητα επιλογής κάθε συγκεκριμένου h είναι $P(h)=\frac{1}{n}$

Άρα, $E[T(n)] = \sum_{h=1}^{n} P(h) \cdot Cost(h) = \frac{1}{n} \sum_{h=1}^{n} Cost(h) =$

$$= \frac{1}{n} \Big(\underbrace{\sum_{h=1}^{\lfloor \sqrt{n} \rfloor} \Theta(n)}_{\sqrt{n} \cdot cn \,\to\, n \text{ φορές}} + \underbrace{\sum_{h=\lfloor \sqrt{n}+1 \rfloor}^{n} \Theta(1)}_{(n-\sqrt{n})k} \Big) =$$

$\approx \sqrt{n}$ · το σταθερό κόστος · $\approx$ · σταθερό κόστος

μέση περίπτωση

$$= \frac{1}{n} [\sqrt{n}\, cn + (n-\sqrt{n})k] = c\sqrt{n} + k\left(1 - \frac{1}{\sqrt{n}}\right) = O(\sqrt{n}),$$

ii). • Η καλύτερη περίπτωση συμβαίνει όταν ισχύει η συνθήκη if $(h \leq \frac{n}{2})$, επομένως γίνεται return $A[n]$, πράξη που αποτελεί $O(1)$.

• Η χειρότερη περίπτωση συμβαίνει όταν ισχύει η συνθήκη else $(h > \frac{n}{2})$, επομένως εκτελείται ο βρόχος for $i=1$ έως h, όπου γίνεται μια πρόσθεση. Άρα, έχουμε $O(n)$.

- Η μέση περίπτωση αντίστοιχα, είναι:

$E[T(n)] = \frac{1}{2}\Theta(1) + \frac{1}{2}\Theta(n) = \Theta(n)$

β) Η μέγιστη ροή φαίνεται οπτικά λόγω των εξής μονοπατιών:

- Υπάρχει μονοπάτι $s \to 1 \to t$ με χωρητικότητα X.
- Υπάρχει μονοπάτι $s \to 2 \to t$ με χωρητικότητα X.

Άρα, η συνολική μέγιστη ροή είναι $2X$.

Στη χειρότερη περίπτωση, ο αλγόριθμος εκμεταλλεύεται την ακμή $1 \to 2$ (χωρητικότητας 1), αυξάνοντας τη ροή μόνο κατά 1 μονάδα σε κάθε βήμα.

1η επανάληψη

Επιλέγεται το μονοπάτι: $s \to 1 \to 2 \to t$, Άρα, η χωρητικότητα του μονοπατιού είναι: $\min(X, 1, X) = 1$. Συνεπώς, η ροή αυξάνεται κατά μία μονάδα με residual graph: καθώς η ακμή $1 \to 2$ γεμίζει πλήρως (forward capacity=0), δημιουργείται μια αντίστροφη ακμή $2 \to 1$, με χωρητικότα 1. Οι ακμές $s \to 1$ και $2 \to t$ έχουν πλέον διαθέσιμη χωρητικότητα $X-1$. (Ροή=1)

2η επανάληψη

Επιλέγεται το μονοπάτι: $s \to 2 \to 1 \to t$. Άρα, η χωρητικότητα του μονοπατιού είναι: $\min(X, 1, X) = 1$. Συνεπώς, η ροή αυξάνεται κατά μία μονάδα με residual graph: καθώς η ακμή 2->1 αδειάζει (forward capacity =1), επαναδιατίθεται η ακμή 1->2 με χωρητικότητα 1. Οι ακμές s->2 και 1->t έχουν πλέον διαθέσιμη χωρητικότητα X-1. (Ροή =2)

3η επανάληψη

Επιλέγεται το μονοπάτι: s -> 1 -> 2 -> t. Άρα, η χωρητικότητα του μονοπατιού είναι: min(X-1, 1, X-1) = =1. Συνεπώς, η ροή αυξάνεται κατά μία μονάδα με residual graph: καθώς η ακμή 1->2 γεμίζει πλήρως (forward capacity =0), δημιουργείται μια αντίστροφη ακμή 2->1 με χωρητικότητα 1. Οι ακμές s->1 και 2->t έχουν πλέον διαθέσιμη χωρητικότητα X-2. (Ροή =3)

Το μοτίβο που παρατηρούμε είναι ότι:

Στις περιττές επαναλήψεις, χρησιμοποιείται η ακμή 1->2 για να σταλεί 1 μονάδα ροής, ενώ στις άρτιες επαναλήψεις, χρησιμοποιείται η αντίστροφη ακμή 2->1 για να ακυρωθεί η ροή της γέφυρας και να σταλεί πάλι 1 μονάδα ροής μέσω της άλλης διαδρομής. Έτσι σε κάθε επανάληψη, η συνολική ροή αυξάνεται με μόλις 1 μονάδα.

Συνεπώς, καθώς η μέγιστη ροή είναι 2X και στη χειρότερη περίπτωση κάθε επανάληψη του αλγορίθμου αυξάνει τη ροή μόνο κατά 1 μονάδα, στη χειρότερη περίπτωση θα χρειαστούν 2X επαναλήψεις.

7) Ο αλγόριθμος βρίσκεται στο αρχείο 7.py. Αν τον συγκρίνουμε με τον αλγόριθμο πινάκων (Matrix Exponentiation), ο οποίος επίσης χρειάζεται O(log n) χρόνο, διαπιστώνουμε πως ο παραπάνω αλγόριθμος ("Fast Doubling") είναι στην πράξη πιο αποδοτικός, για τους εξής λόγους:

- **Αριθμητικές Πράξεις:** Στον αλγόριθμο πινάκων, κάθε βήμα απαιτεί τον πολλαπλασιασμό δύο πινάκων 2 x 2. Δηλαδή, 8 πολλαπλασιασμούς και 4 προσθέσεις ανά βήμα. Αντιθέτως, στον παραπάνω αλγόριθμο, μέσω της χρήσης των τύπων F_2k και F_2k+1, αποφεύγουμε τους περιττούς υπολογισμούς, μειώνοντας τις πράξεις σε περίπου 2 πολλαπλασιασμούς και 2 προσθέσεις ανά αναδρομικό βήμα. Συνεπώς, αν και οι δύο έχουν την ίδια πολυπλοκότητα, ο παραπάνω αλγόριθμος φέρει μικρότερο σταθερό παράγοντα.
- **Διαχείριση Μνήμης και Αναδρομή:** Παρόλο που η αναδρομική υλοποίηση δεσμεύει χώρο στη στοίβα, το βάθος της αναδρομής είναι λογαριθμικό O(log n). Ακόμα και για πολύ μεγάλα n, το βάθος της στοίβας είναι αμελητέο καθώς μειώνεται σε πολύ μεγάλο βαθμό. Πρακτικά, αποτελεί το μέγεθος των bits του n, έναντι του ίδιου του n. Το χρονικό κέρδος από τη μείωση των αριθμητικών πράξεων υπερκαλύπτει κατά πολύ το ελάχιστο κόστος διαχείρισης των κλήσεων της στοίβας.

Συμπερασματικά, λόγω των λιγότερων αριθμητικών πράξεων ανά βήμα, η βέλτιστη προσέγγιση για τον υπολογισμό της ακολουθίας είναι αυτή του παραπάνω αλγορίθμου (Fast Doubling) και όχι η κλασική ύψωση πίνακα σε δύναμη.