1. Consider a text t and a pattern p such that $|t| \leq 2|p|$. Prove that all occurrences of p in t create a single arithmetical progression (that is, are exactly of the form $x + \alpha \cdot y$ where $0 \leq \alpha < k$).

2. Fibonacci words f_n are defined recursively as follows: $f_1 = b$, $f_2 = a$ and $f_{n+2} = f_{n+1}f_n$. For a word of length at least 2, let $c(w)$ denote the word obtained from w by swapping its last two letters, for example $c(\mathtt{ababa}) = \mathtt{abaab}$. Show that $f_{n-2}f_{n-1} = c(f_{n-1}f_{n-2})$ for any $n \geq 3$.

3. Let $w = f_n$ for some $n \geq 3$ and consider the corresponding π array. Prove by induction that, for every $|f_{n-1}| - 1 \leq i < |f_n| - 1$, $\pi[i] = i - |f_{n-2}|$. You might find the previous question useful.

(2 points) 4. We proved that, if $\pi'[i] \neq -1$ and $\pi'[\pi'[i]] \neq -1$, then $i \geq \pi'[i] + \pi'[\pi'[i]] + 2$, and replacing π by π' in the KMP algorithm decreases its *delay* (worst-case time to process a letter of the text) to $O(\log n)$. Write a pseudocode of the resulting algorithm. Then design an instance on which the delay is $\Omega(\log n)$ (for some letter of the text) by considering the pattern f_n and looking at the values of $\pi'[|f_k| - 2]$ for $3 \leq k \leq n$. Again, you might find the previous question useful.

(3 points) 5. Modify the KMP algorithm so that the delay becomes $O(1)$. There are different solutions, one proceeds as follows: the more times we need to iterate the line $j = \pi[j]$ to update the current ℓ the further to the right is the the last letter of the nearest possible occurrence of the pattern, so might hope to demoartise computing the new ℓ. Can you modify the preprocessing stage (computing the π array) to also have constant delay?