

# Análisis del algoritmo Knuth-Morris-Pratt con énfasis en la programación funcional

Ángel Iván Gladín García

Resumen de Proyecto de Tesis

8 de marzo de 2021.

No. cuenta: 313112470

## 1. Resumen

Los algoritmos de búsqueda de subcadenas son una clase de algoritmos de cadenas que tratan de buscar un(os) *patron(es)* en una cadena o texto. Normalmente cuando es una cadena grande se busca mejorar el desempeño del mismo. El algoritmo de Knuth-Morris-Pratt es útil para buscar un solo patrón (con longitud  $m$ ) en un texto (de longitud  $n$ ) con una complejidad en tiempo de  $\Theta(n + m)$ .

En este trabajo se hará énfasis en éste algoritmo centrado en la programación funcional; primero se dará una especificación formal y por medio de razonamiento ecuacional se refinará una versión más eficiente en complejidad en tiempo. En éste análisis se desglosará la teoría necesaria para poder llevar a cabo este proceso.

Finalmente se usará este algoritmo para resolver algunos problemas famosos que llegan a aparecer en competencias o acertijos de programación competitiva y se discutirá su uso con sus respectivas implicaciones.

## 2. Índice tentativo

1. **Fundamentos:** En esta sección se darán definiciones y teoremas necesarios usadas en la programación funcional para poder profundizar en el razonamiento ecuacional. También se darán propiedades y definiciones útiles respecto a *cadenas*, análisis de tiempo en algoritmos y un primer acercamiento a búsqueda de subcadenas.
2. **Algoritmos de subcadenas con perspectiva en la programación imperativa:** Se presentará dos algoritmos; la versión ingenua y el algoritmo KMP y se analizarán algunos ejemplos y el algoritmo *per se*.
3. **Algoritmos de subcadenas con perspectiva en la programación funcional:** Aquí es donde se hará uso de todos los fundamentos presentados en el primer capítulo para hacer la *derivación* del algoritmo ingenuo al algoritmo Morris-Pratt después a Knuth-Morris-Pratt. También se discutirá sobre la *función de error* que es usada en KMP. Se hará hincapié en los teoremas y *leyes* usadas en la programación funcional.
4. **Quickcheck:** Se hablará acerca de esta biblioteca en Haskell para poder probar las propiedades que deberían de cumplir las funciones, es decir, cada función tiene propiedades deseables lo que se logra con QuickCheck es ver si se cumplen total o parcialmente estas propiedades. Una ventaja notoria es que la propiedad es probada con una gran cantidad de casos generados aleatoriamente.  
Será usado para probar las implementaciones derivadas con la implementación ingenua.
5. **Jueces en línea:** Se hablará brevemente en qué consiste la programación competitiva y algunas “heurísticas” a seguir en la resolución de problemas. Al final se atacarán tres problemas en los lenguajes de programación Haskell y C++ usando el algoritmo KMP y la función de error.
6. **Conclusiones:** Se centrará en las ventajas y desventajas en el razonamiento ecuacional en la programación funcional, se hablará sobre algunas herramientas para poder verificar nuestros programas y se darán ideas a trabajos futuros.
7. **Apéndices\*:** Se abordarán cuestiones técnicas.

## 3. Bibliografía básica

- [1] Bird, R. (2010). Pearls of Functional Algorithm Design. Cambridge: Cambridge University Press.
- [2] Bird, R. (2014). Thinking Functionally with Haskell. Cambridge: Cambridge University Press.
- [3] Graham Hutton. “A Tutorial on the Universality and Expressiveness of Fold”. En: J. Funct. Program. 9.4 (jul. de 1999), págs. 355-372. <https://doi.org/10.1017/S0956796899003500>.
- [4] Richard Bird. “On building cyclic and shared structures in Haskell”. En: Formal Aspects of Computing 24 (jul. de 2012).
- [5] Introduction to Algorithms T. Cormen, C. Leiserson, R. Rivest, and C. Stein. The MIT Press, 3rd Edition.