Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2283 - Diseño y Análisis de Algoritmos

Ayudantía 6 - Algoritmos Monte Carlo

15 de octubre de 2020

Profesor Marcelo Arenas

Bernardo Barías

Resumen Algoritmos Aleatorizados

- **Monte Carlo**: el algoritmo siempre entrega un resultado, pero hay una probabilidad de que este sea incorrecto.
- **Las Vegas**: si el algoritmo entrega un resultado entonces este es correcto, pero hay una probabilidad de que el algoritmo no entregue ningún resultado.

Pregunta 1 - Pattern Matching

Consideremos el problema de *pattern matching* sobre strings. Un texto es un string

$$X = x_1 x_2 \cdots x_n,$$

y un patrón es un string

$$Y = y_1 y_2 \cdots y_m,$$

ambos sobre un alfabeto Σ finito. Sin pérdida de generalidad supongamos que $\Sigma = \{0, 1\}$. El patrón ocurre en el texto si existe un $j \in \{1, ..., n - m + 1\}$ tal que para $i \in [1, m]$ se cumple que $x_{j+i-1} = y_i$. El problema de *pattern matching* es encontrar alguna ocurrencia del patrón en el texto (si es que existe).

a) De una solución de fuerza bruta al problema.

b) Para un número primo p defina la función $F_p : \mathbb{N} \to \{0, 1, ..., p - 1\}$ como

$$F_p(x) = x \bmod p$$

Demuestre que si $0 \leq a, b \leq 2^n$ entonces

$$Pr_{p \leq \tau}(F_p(a) = F_p(b) \mid a \neq b) \ \leq \ \frac{n}{\pi(\tau)},$$

donde $\pi(\tau)$ retorna la cantidad de números primos menores o iguales a τ.

c) Se propone el siguiente algoritmo aleatorizado (Monte Carlo):

Algoritmo 1 PatternMatching(X, Y)

```
p = random_prime(τ)
for j in {1, ..., n} do
    X(j) = Σ_{i=0}^{m−1} x_{j+i} · 2^{m−1−i}
    if F_p(X(j)) == F_p(Y) then
        return j
    end if
end for
return No match
```

Demuestre la complejidad asintótica y la probabilidad de error del algoritmo. Para la complejidad asuma que $random\_prime(\tau)$ demora $O(1)$, y para la probabilidad de error asuma el siguiente resultado conocido sobre la función $\pi(x)$:

$$\pi(x) \geq \frac{x}{\log x}$$

d) Haga una (pequeña) modificación al algoritmo de manera que la complejidad asintótica sea $O(n + m)$.