Otros: primalidad, árboles binarios aleatorizados, hashing universal y perfecto, skiplists.

Primalidad Miller-Rabin

- → n NO es primo
- → n puede ser primo.

aplicaciones en criptografía (llaves públicas y privadas, factorización prima de #s grandes)

```
primo(n)
    Sean s, d tq.
        n-1 ≡ 2^s · d , d impar.
    repetir k veces {
        elegir a ∈ [1, n-1] al azar
        si a^d ≠ 1 mod n y                                   (*)
           ∀ r ∈ [0, s-1], a^(2^r · d) ≠ -1 mod n   (**)
           retornar "compuesto"
    retornar "probablemente compuesto"
```

"a es testigo de que n es compuesto"

(Se sabe (?) de que la probabilidad de que este algoritmo se equivoque es de 1/4)

- → si n es compuesto, existen al menos $\frac{3}{4}n$ testigos $a \in [1, n-1]$ de que lo es.
- → probabilidad de que el test se equivoque es $1/4^k$
- → es MonteCarlo y aleatorizado, es 1 sided
- → tiempo $\Theta(k \cdot \log n)$ multiplicaciones

(*) $\Theta(\log n)$ → $a, a^2, a^4, a^8, a^{16}, \ldots$ $r = 0$

(**) $\Theta(\log n)$ → $r = 1 \rightarrow (a^d)^2$, $r = 2 \rightarrow ((a^d)^2)^2$

Hace unos años salió paper "Primes is in P", $\Theta(\log^8 n)$ multiplicaciones

$\Rightarrow k \sim \log^7 n$

pero n es gigantesco!! → Sigue siendo preferible y menos costoso Miller-Rabin. Para k no muy grande, P de error es muy baja!

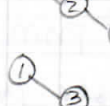
Para mismo input puedo tener árboles distintos (input = secuencia)

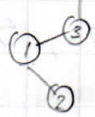
Arboles Aleatorizados Binarios

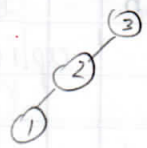
2015年11月19日(木)

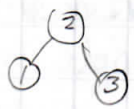
$S_1 = 6$, depende del orden en que llegan las claves

123 

132 

312 

321 

{231 / 213 

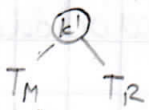
Cut(T, k) devuelve dos árboles con las claves $< k$ y $> k$

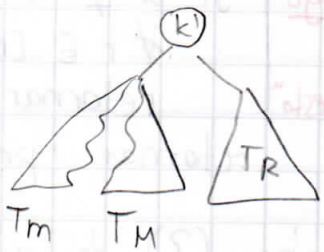
Si $T = \square$ retornar $(\square, \square)$

Sea $T =$ 

Si $k < k'$

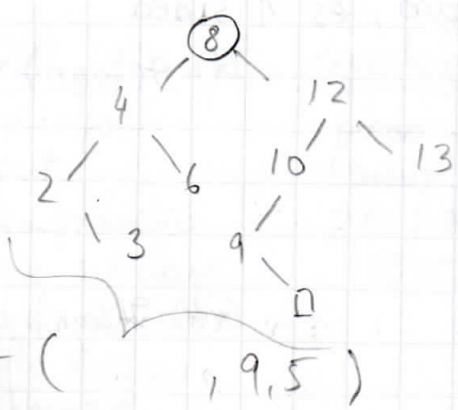
$(T_m, T_M) \leftarrow Cut(T_L, k)$

return $(T_m,$  $)$

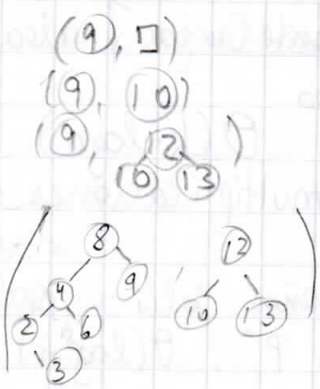


$(T_m, T_M) \leftarrow Cut(T_R, k)$

retornar $($  $, T_M)$



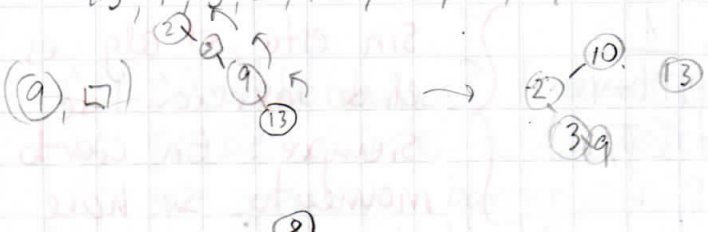
$Cut($ $)$



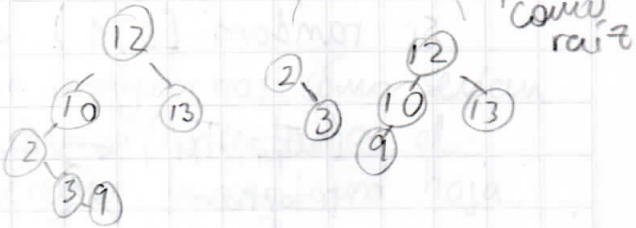
Es como si 9,5 se hubiese insertado al comienzo

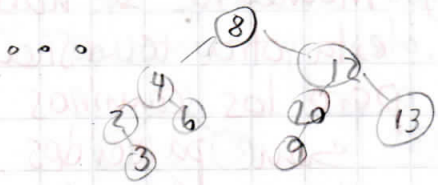
Insertemos con método de Cut.

13, 9, 3, 2, 10, 12, 6, 4, 8



así corta el 6, y luego se pone como raíz



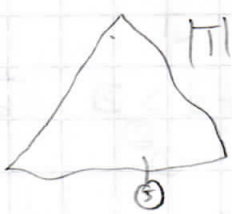


Es el mismo árbol que se obtiene al insertar de forma normal 8, 4, 6, 12, 10, 2, 3, 9, 13. !!

Ahora supongamos que todas las secuencias posibles tienen dist uniforme de aparecer.

Prob (nueva clave sea la primera en insertarse)

$= \boxed{\frac{1}{n+1}}$



bajo algún algoritmo que produce árboles ≠'s dependiendo del orden de inserción.

123

⋮

321 y ahora insertan 4.

En todas las secuencias de 4 números, ¿cuál es la prob. de que el 4 sea la primera en insertarse, de todas las secuencias?

En nuestro caso ya NO depende el árbol. Estamos simulando todos los posibles casos en que la nueva clave podría haberse insertado primero según algún algoritmo que depende del orden

Insertar (T, k)

$T = \square$, retornar (k)

si random $[0,1) < \frac{1}{|T|+1}$

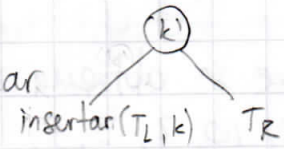
$(T_m, T_M) \leftarrow cut(T, k)$

retornar

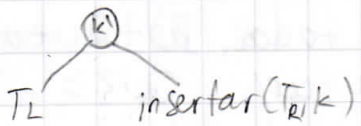
} Sin esto, el alg. es el de inserción de siempre. En cierto momento, se hace esta otra recursión, pero los caminos son parecidos, de igual costo.

Sea $T =$

si $k < k'$ retornar



else retornar

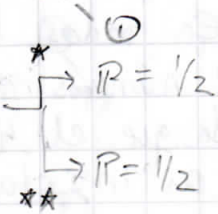


Ej:

insertar (1)

Para 123 ...

insertar (2)



} esos 2 árboles salen con prob. $\frac{1}{2}$ cada 1.

insertar (3)