# Outline

1. Aplicaciones a la predicción de saltos
   - MinMax: records en permutaciones
   - Exponeciación sesgada

# MinMax: un ejemplo paradójico

Sean los algoritmos siguientes para encontrar simultáneamente el mínimo y el máximo de un array $T$ de largo $n$.

```
min = max = T[0];
for(i = 1; i < n; i++) {
    if (T[i] < min)
        min = T[i];
    if (T[i] > max)
        max = T[i];
}
```

**MinMax "ingenuo"**

$2n - 2$ comparaciones

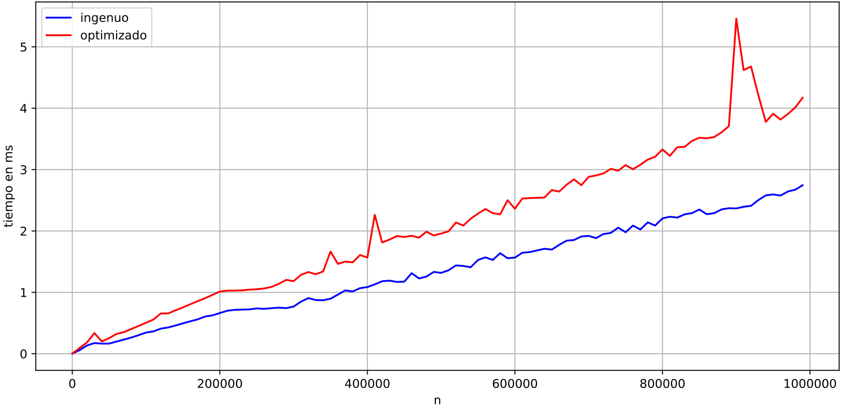
```
min = max = T[n-1];
for(i = 0; i < n - 1; i += 2) {
    if (T[i] < T[i+1]) {
        if (T[i] < min)
            min = T[i];
        if (T[i+1] > max)
            max = T[i+1];
    } else {
        if (T[i+1] < min)
            min = T[i+1];
        if (T[i] > max)
            max = T[i];
    }
}
```

**MinMax "optimizado"**

$\sim \frac{3}{2}n$ comparaciones

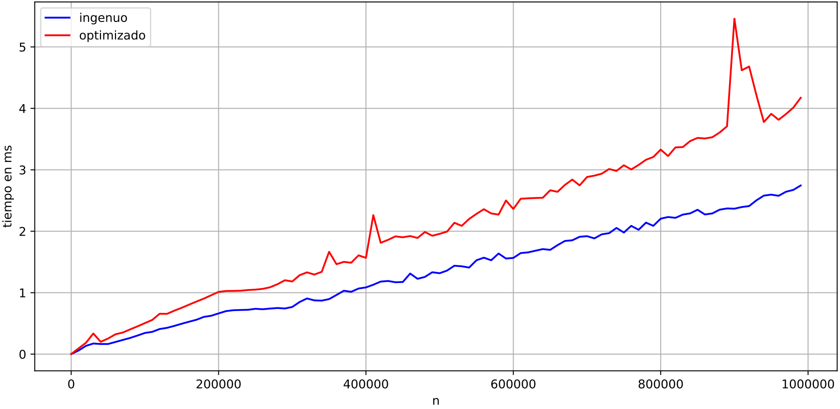
# MinMax: resultados prácticos para los algoritmos

Considerando $T$ como una permutación aleatoria:



# MinMax: resultados prácticos para los algoritmos

Considerando $T$ como una permutación aleatoria:




¿Por qué? ¿modelo?

# Optimizaciones de “bajo nivel”

La arquitectura de la computadora incluye varias optimizaciones:

- La jerarquía de memoria (*memoria cache*),
- Operaciones *SIMD* (Single Instruction, Multiple Data),
- El pipeline del procesador.

# Optimizaciones de “bajo nivel”

La arquitectura de la computadora incluye varias optimizaciones:

- La jerarquía de memoria (*memoria cache*),
- Operaciones *SIMD* (Single Instruction, Multiple Data),
- El pipeline del procesador.

En nuestro caso no hay SIMD, y acceso a memoria es esencialmente el mismo en los dos algoritmos:

$\Rightarrow$ nos vamos a concentrar en el pipeline.

# Optimizaciones de “bajo nivel”

La arquitectura de la computadora incluye varias optimizaciones:

- La jerarquía de memoria (*memoria cache*),
- Operaciones *SIMD* (Single Instruction, Multiple Data),
- El pipeline del procesador.

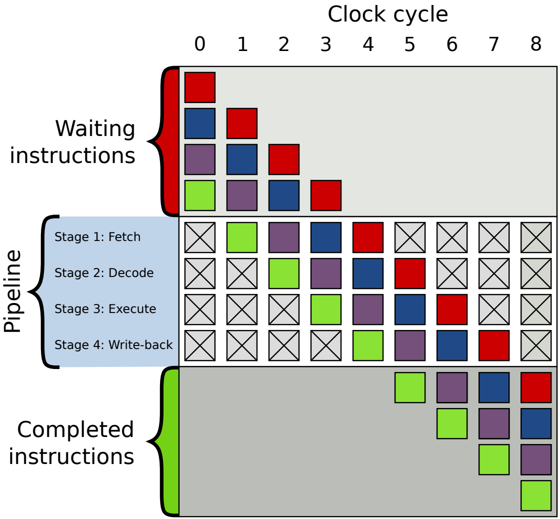
En nuestro caso no hay SIMD, y acceso a memoria es esencialmente el mismo en los dos algoritmos:

$\Rightarrow$ nos vamos a concentrar en el pipeline.

El pipeline del procesador:

- ejecutar una instrucción requiere *fetch*, *decode*, *execute*, *write*:
  traer instrucción de memoria, decodificar, ejecutar, escribir
- en un ciclo de reloj se pueden realizar en paralelo para varias instrucciones sucesivas, en distintas etapas del pipeline.

# El pipeline del procesador




[0] Fuente: Wikipedia, por en:User:Cburnett, **CC BY-SA 3.0**, `https://commons.wikimedia.org/w/index.php?curid=1499754`

# El pipeline del procesador

**Problema.** un if provoca un dilema:

¿qué rama (branch) de ejecución tomar (fetch)?

$\Rightarrow$ error de predicción provoca pérdida del pipeline (paralelismo)

# El pipeline del procesador

**Problema.** un if provoca un dilema:

¿qué rama (branch) de ejecución tomar (fetch)?

⇒ error de predicción provoca pérdida del pipeline (paralelismo)

**Branch prediction.** diseñar esquemas para predecir el resultado de un if

- Locales (cada if separado), globales, mixtos, ...

# El pipeline del procesador

**Problema.** un if provoca un dilema:

¿qué rama (branch) de ejecución tomar (fetch)?

⇒ error de predicción provoca pérdida del pipeline (paralelismo)

**Branch prediction.** diseñar esquemas para predecir el resultado de un if

- Locales (cada if separado), globales, mixtos, ...
- Memoria: ¿cuánta historia recuerda un predictor?

# El pipeline del procesador

**Problema.** un if provoca un dilema:

¿qué rama (branch) de ejecución tomar (fetch)?

⇒ error de predicción provoca pérdida del pipeline (paralelismo)

**Branch prediction.** diseñar esquemas para predecir el resultado de un if

- Locales (cada if separado), globales, mixtos, ...
- Memoria: ¿cuánta historia recuerda un predictor?

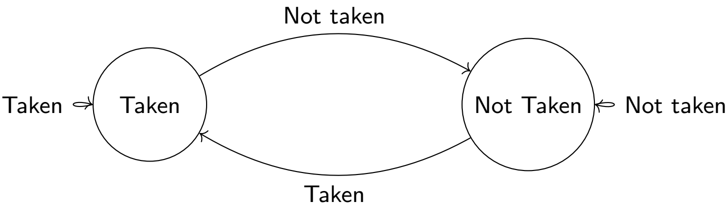
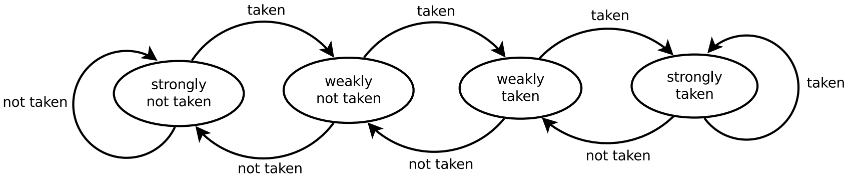



**Figura:** Predictor 1 Bit

# Esquemas de predicción de branching

Por simplicidad consideraremos los siguientes predictores **locales**:

- Predictor de 1 bit [pagina precedente],
- Predictor de 2 bits saturado




- Predictor de 3 bits saturado ...

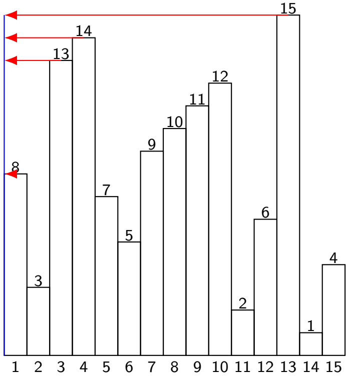
[0] Fuente: Wikipedia, Afog derivative work: ENORMATOR (talk), **CC BY-SA 3.0**, `File:Branch_prediction_2bit_saturating_counter-dia.svg`

# Records en permutaciones

Errores de predicción en los dos MinMax relacionados con los **records**

## Definición

Una posición $k$ en una permutación $\pi$ es un record máximo (mínimo) sii $\pi_i < \pi_k$ (resp. $\pi_i > \pi_k$) para todo $i < k$.



# Records en permutaciones

Errores de predicción en los dos MinMax relacionados con los **records**

**Definición**

Una posición $k$ en una permutación $\pi$ es un record máximo (mínimo) sii $\pi_i < \pi_k$ (resp. $\pi_i > \pi_k$) para $i < k$.

# Records en permutaciones

Errores de predicción en los dos MinMax relacionados con los **records**

**Definición**

Una posición $k$ en una permutación $\pi$ es un record máximo (mínimo) sii $\pi_i < \pi_k$ (resp. $\pi_i > \pi_k$) para $i < k$.

a) En la línea $(3)$, la condición es verdadera sii $i$ es un record de mínimo.

b) En la línea $(5)$, la condición es verdadera sii $i$ es un record de máximo.

```
1 min = max = T[0];
2 for(i = 1; i < n; i++) {
3     if (T[i] < min)
4         min = T[i];
5     if (T[i] > max)
6         max = T[i];
7 }
```

# Records en permutaciones

Errores de predicción en los dos MinMax relacionados con los **records**

### Definición

Una posición $k$ en una permutación $\pi$ es un record máximo (mínimo) sii $\pi_i < \pi_k$ (resp. $\pi_i > \pi_k$) para $i < k$.

a) En la línea $(3)$, la condición es verdadera sii $i$ es un record de mínimo.

b) En la línea $(5)$, la condición es verdadera sii $i$ es un record de máximo.

```
1 min = max = T[0];
2 for(i = 1; i < n; i++) {
3     if (T[i] < min)
4         min = T[i];
5     if (T[i] > max)
6         max = T[i];
7 }
```

### Observación

Por simetría basta estudiar records de máximo.

# La cantidad esperada de records

Sea $R_n(\pi)$ la cantidad de records en $\pi \in \mathcal{S}_n$, y $e_n := \mathbb{E}[R_n]$.

**Proposición**

La cantidad esperada de records es $e_n = H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$.

# La cantidad esperada de records

Sea $R_n(\pi)$ la cantidad de records en $\pi \in \mathcal{S}_n$, y $e_n := \mathbb{E}[R_n]$.

**Proposición**

La cantidad esperada de records es $e_n = H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$.

**Demostración.**

Sea $E_j = \{\pi \in \mathcal{S}_n : j$ es un record de $\pi\}$.

Observar que:

1. Tenemos $R_n = \sum_{j=1}^{n} \mathbf{1}_{E_j}$, así $e_n = \sum_{j=1}^{n} \Pr(E_j)$,

# La cantidad esperada de records

Sea $R_n(\pi)$ la cantidad de records en $\pi \in \mathcal{S}_n$, y $e_n := \mathbb{E}[R_n]$.

**Proposición**

La cantidad esperada de records es $e_n = H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$.

**Demostración.**

Sea $E_j = \{\pi \in \mathcal{S}_n : j \text{ es un record de } \pi\}$.

Observar que:

1. Tenemos $R_n = \sum_{j=1}^{n} \mathbf{1}_{E_j}$, así $e_n = \sum_{j=1}^{n} \Pr(E_j)$,
2. Los eventos $E_j$ satisfacen $\Pr(E_j) = \frac{1}{j}$. $\square$.

# Branch misses: MinMax ingenuo

**Proposición[Auger,Nicaud,Pivoteau'16]**

La cantidad esperada de errores de predicción en el MinMax ingenuo para una permutación aleatoria es asintóticamente:

$$4 \log n \quad \text{(predictor 1-bit)}, \qquad 2 \log n \quad \text{(predictor 2-bit,3-bit,...)}$$

# Branch misses: MinMax optimizado

**Proposición [Auger,Nicaud,Pivoteau'16]**

La cantidad esperada de errores de predicción en el MinMax optimizado es asintóticamente:

$$\frac{1}{4}n + O(\log n)\,,$$

para los predictores de $1, 2, 3, \ldots$ bits.

a) Condición de línea $(3)$, es decir $T[i] < T[i+1]$, se cumple con probabilidad $1/2$ para un $i$ dado.

b) El evento $T[i] < T[i+1]$ es independiente de la historia

$$T[0], \ldots, T[i-2], T[i-1]\,.$$

c) Los otros ifs contribuyen $O(\log n)$.

```
1 min = max = T[n-1];
2 for(i = 0; i < n - 1; i += 2) {
3     if (T[i] < T[i+1]) {
4         if (T[i] < min)
5             min = T[i];
6         if (T[i+1] > max)
7             max = T[i+1];
8     } else {
9         if (T[i+1] < min)
              ...
```

# Branch misses en una sola ejecución

Probamos $\mathbb{E}[R_n] \sim \log n$, pero ¿y si ejecutamos el algoritmo una sola vez?

[0] Para MinMax ingenuo, sabemos que la cantidad de errores de predicción es $O(R_n)$.

# Branch misses en una sola ejecución

Probamos $\mathbb{E}[R_n] \sim \log n$, pero ¿y si ejecutamos el algoritmo una sola vez?

**Proposición**

*Se cumple que* $R_n \sim \log n$ *en probabilidad*

[0] Para MinMax ingenuo, sabemos que la cantidad de errores de predicción es $O(R_n)$.

# Branch misses en una sola ejecución

Probamos $\mathbb{E}[R_n] \sim \log n$, pero ¿y si ejecutamos el algoritmo una sola vez?

**Proposición**

*Se cumple que $R_n \sim \log n$ en probabilidad*

**Recordamos.** Una secuencia de variables aleatorias $X_n$ satisface $X_n \sim f(n)$ en probabilidad sii, para cada $\varepsilon > 0$ fijo,

$$\Pr(X_n \in [(1-\varepsilon)f(n), (1+\varepsilon)f(n)]\ ) \to 1.$$

[0] Para MinMax ingenuo, sabemos que la cantidad de errores de predicción es $O(R_n)$.

# Branch misses en una sola ejecución

Probamos $\mathbb{E}[R_n] \sim \log n$, pero ¿y si ejecutamos el algoritmo una sola vez?

**Proposición**

*Se cumple que* $R_n \sim \log n$ *en probabilidad*

**Recordamos.** Una secuencia de variables aleatorias $X_n$ satisface $X_n \sim f(n)$ en probabilidad sii, para cada $\varepsilon > 0$ fijo,

$$\Pr(X_n \in [(1-\varepsilon)f(n), (1+\varepsilon)f(n)]\ ) \to 1.$$

$\Rightarrow$ Típicamente $R_n$ está "cerca" de $\log n$.

[0] Para MinMax ingenuo, sabemos que la cantidad de errores de predicción es $O(R_n)$.

# Desigualdad de Chebyshev

**Proposición (Concentración)**

*Supongamos que* $\mathbb{E}[X_n^2] \sim \mathbb{E}[X_n]^2$, *y* $\mathbb{E}[X_n] \to \infty$, *cuando* $n \to \infty$.

*Entonces* $X_n \sim \mathbb{E}[X_n]$ *en probabilidad.*

**Lema (Chebyshev)**

*Sea* $X$ *una variable aleatoria, entonces*

$$\Pr(|X - \mathbb{E}[X]| \geq \epsilon) \leq \frac{\texttt{Var}(X)}{\epsilon^2}.$$

# Concentración de la cantidad de records

**Demostración.**

Probamos que $d_n := \mathbb{E}[R_n^2] \sim \mathbb{E}[R_n]^2 = e_n^2$.

Sea $E_j = \{\pi \in S_n : j \text{ es un record de } \pi\}$. Observar que:

1. Los eventos $E_j$ satisfacen $\Pr(E_j) = \frac{1}{j}$.

# Concentración de la cantidad de records

**Demostración.**

Probamos que $d_n := \mathbb{E}[R_n^2] \sim \mathbb{E}[R_n]^2 = e_n^2$.

Sea $E_j = \{\pi \in S_n : j \text{ es un record de } \pi\}$. Observar que:

1. Los eventos $E_j$ satisfacen $\Pr(E_j) = \frac{1}{j}$.
2. Los eventos $E_j$ y $E_k$ son independientes para $j \neq k$,
$$\Pr(E_j \cap E_k) = \frac{1}{j \cdot k} = \Pr(E_j) \cdot \Pr(E_k)\,.$$

# Concentración de la cantidad de records

## Demostración.

Probamos que $d_n := \mathbb{E}[R_n^2] \sim \mathbb{E}[R_n]^2 = e_n^2$.

Sea $E_j = \{\pi \in S_n : j$ es un record de $\pi\}$. Observar que:

1. Los eventos $E_j$ satisfacen $\Pr(E_j) = \frac{1}{j}$.
2. Los eventos $E_j$ y $E_k$ son independientes para $j \neq k$,

$$\Pr(E_j \cap E_k) = \frac{1}{j \cdot k} = \Pr(E_j) \cdot \Pr(E_k)\,.$$

$\Rightarrow$ las indicatrices $X_j = \mathbf{1}_{E_j}$ son independientes: para $j \neq k$

$$\mathbb{E}[X_j X_k] = \mathbb{E}[X_j]\mathbb{E}[X_k] = \frac{1}{j \cdot k}\,.$$

# Concentración de la cantidad de records

## Demostración.

Probamos que $d_n := \mathbb{E}[R_n^2] \sim \mathbb{E}[R_n]^2 = e_n^2$.

Sea $E_j = \{\pi \in S_n : j$ es un record de $\pi\}$. Observar que:

1. Los eventos $E_j$ satisfacen $\Pr(E_j) = \frac{1}{j}$.
2. Los eventos $E_j$ y $E_k$ son independientes para $j \neq k$,
$$\Pr(E_j \cap E_k) = \frac{1}{j \cdot k} = \Pr(E_j) \cdot \Pr(E_k)\,.$$
$\Rightarrow$ las indicatrices $X_j = \mathbf{1}_{E_j}$ son independientes: para $j \neq k$
$$\mathbb{E}[X_j X_k] = \mathbb{E}[X_j]\mathbb{E}[X_k] = \frac{1}{j \cdot k}\,.$$

Usando $R_n = \sum_{j=1}^{n} X_j$ obtenemos

$$\mathbb{E}[R_n^2] = H_n + 2\sum_{j=2}^{n} \frac{H_{j-1}}{j}\,.$$

# Concentración de la cantidad de records

**Demostración.**

Probamos que $d_n := \mathbb{E}[R_n^2] \sim \mathbb{E}[R_n]^2 = e_n^2$.

Sea $E_j = \{\pi \in S_n : j \text{ es un record de } \pi\}$. Observar que:

1. Los eventos $E_j$ satisfacen $\Pr(E_j) = \frac{1}{j}$.
2. Los eventos $E_j$ y $E_k$ son independientes para $j \neq k$,

$$\Pr(E_j \cap E_k) = \frac{1}{j \cdot k} = \Pr(E_j) \cdot \Pr(E_k) .$$

$\Rightarrow$ las indicatrices $X_j = \mathbf{1}_{E_j}$ son independientes: para $j \neq k$

$$\mathbb{E}[X_j X_k] = \mathbb{E}[X_j]\mathbb{E}[X_k] = \frac{1}{j \cdot k} .$$

Usando $R_n = \sum_{j=1}^n X_j$ obtenemos

$$\mathbb{E}[R_n^2] = H_n + 2 \sum_{j=2}^{n} \frac{H_{j-1}}{j} .$$

Considerando $(\log n)^2 \sim H_n^2 = \sum_{j=1}^n \frac{1}{j^2} + 2 \sum_{j=2}^n \frac{H_{j-1}}{j}$ $\square$

# Concentración del MinMax optimizado

Sea $A_i = \{$ branch miss en $T[i] < T[i+1]\}$. Observamos que:

- $\Pr(A_i) = 1/2$,
- $A_i$ es independiente de $A_j$ para $|i - j| > 1$.

# Concentración del MinMax optimizado

Sea $A_i = \{$ branch miss en $T[i] < T[i+1]\}$. Observamos que:

- $\Pr(A_i) = 1/2$,
- $A_i$ es independiente de $A_j$ para $|i-j| > 1$.

Tenemos $m \approx \frac{n}{2}$ variables aleatorias Bernoulli $\frac{1}{2} - \frac{1}{2}$ independientes:

$$C_m := \sum_{i=0}^{m-1} \mathbf{1}_{A_{2i}}$$

# Concentración del MinMax optimizado

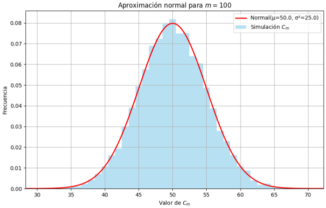
Sea $A_i = \{$ branch miss en $T[i] < T[i+1]\}$. Observamos que:

- $\Pr(A_i) = 1/2$,
- $A_i$ es independiente de $A_j$ para $|i-j| > 1$.

Tenemos $m \approx \frac{n}{2}$ variables aleatorias Bernoulli $\frac{1}{2} - \frac{1}{2}$ independientes:

$$C_m := \sum_{i=0}^{m-1} \mathbf{1}_{A_{2i}}$$

Se tiene el Teorema Central del Límite $(C_m - m/2)/\sqrt{m/4} \to N(0,1)$



# Sesgar algoritmos para acelerarlos

Se necesita un compromiso:

- Un **if** con una condición que es `True` con proba. $50\,\%$ (e independiente del pasado) es un problema para el predictor.
- Un **if** con una condición que no es $50-50$ e independiente del pasado presenta redundancias.

# Sesgar algoritmos para acelerarlos

Se necesita un compromiso:

- Un **if** con una condición que es `True` con proba. $50\,\%$ (e independiente del pasado) es un problema para el predictor.
- Un **if** con una condición que no es $50 - 50$ e independiente del pasado presenta redundancias.

Veamos otro ejemplo: la exponenciación...

# Exponenciación sesgada

```
r = 1;
while (n > 0) {
    // n es impar
    if (n & 1)
        r = r * x;
    n /= 2;
    x = x * x;
}
```

**Potencia clásica**

```
r = 1;
while (n > 0) {
    t = x * x;
    // n1 n0 != 0 0
    if (n & 3) {
        if (n & 1)
            r = r * x;
        if (n & 2)
            r = r * t;
    }
    n /= 4;
    x = t * t;
}
```

**Potencia sesgada**

# Exponenciación sesgada

```
r = 1;
while (n > 0) {
    // n es impar
    if (n & 1)
        r = r * x;
    n /= 2;
    x = x * x;
}
```

**Potencia clásica**

```
r = 1;
while (n > 0) {
    t = x * x;
    // n1 n0 != 0 0
    if (n & 3) {
        if (n & 1)
            r = r * x;
        if (n & 2)
            r = r * t;
    }
    n /= 4;
    x = t * t;
}
```

**Potencia sesgada**

- En la potencia clásica, a priori cada bit de $n$ es $1/2 - 1/2$ independiente.
- En la potencia sesgada, el primer **if** aumenta la probabilidad de los otros dos!

# Exponenciación sesgada

```
r = 1;
while (n > 0) {
    // n es impar
    if (n & 1)
        r = r * x;
    n /= 2;
    x = x * x;
}
```

**Potencia clásica**

```
r = 1;
while (n > 0) {
    t = x * x;
    // n1 n0 != 0 0
    if (n & 3) {
        if (n & 1)
            r = r * x;
        if (n & 2)
            r = r * t;
    }
    n /= 4;
    x = t * t;
}
```

**Potencia sesgada**

- En la potencia clásica, a priori cada bit de $n$ es $1/2 - 1/2$ independiente.
- En la potencia sesgada, el primer **if** aumenta la probabilidad de los otros dos!
- Igual cantidad de multiplicaciones, pero más ifs ! ¿Quién ganará?

# Análisis de la exponenciación sesgada

**Modelo.** Consideramos $k > 0$ y $n \in [0, 2^{2k} - 1]$ aleatorio:

$$n = n_{2k-1} n_{2k-2} \ldots n_1 n_0 \,,$$

con cada $n_i$ independiente y $n_i \sim \texttt{Ber}(1/2)$.

Consideramos predictores de $1$-bit y $2$-bits.

# Análisis de la exponenciación sesgada

**Modelo.** Consideramos $k > 0$ y $n \in [0, 2^{2k} - 1]$ aleatorio:

$$n = n_{2k-1} n_{2k-2} \dots n_1 n_0 \,,$$

con cada $n_i$ independiente y $n_i \sim \texttt{Ber}(1/2)$.

Consideramos predictores de $1$-bit y $2$-bits.

**Plan para el análisis.** Modelamos estado de predictor como una *Cadena de Markov*, nos interesa contar transiciones asociadas a "branch-miss"

# Análisis de la exponenciación sesgada

**Modelo.** Consideramos $k > 0$ y $n \in [0, 2^{2k} - 1]$ aleatorio:

$$n = n_{2k-1} n_{2k-2} \dots n_1 n_0 \, ,$$

con cada $n_i$ independiente y $n_i \sim \texttt{Ber}(1/2)$.

Consideramos predictores de $1$-bit y $2$-bits.

**Plan para el análisis.** Modelamos estado de predictor como una *Cadena de Markov*, nos interesa contar transiciones asociadas a "branch-miss"

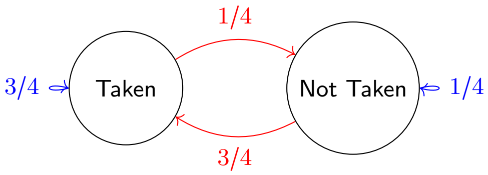



**Figura:** Predictor 1 Bit para if exterior

# Modelo: cadenas de Markov

- Leemos pares [independientes] $(n_{2i+1}, n_{2i})$, $i = 0, 1, 2, \ldots, k-1$.
- Seguimos el estado del predictor de cada if.

Resultado : cadenas de Markov.

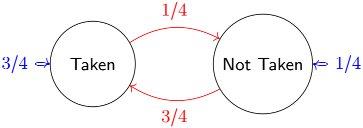



**Figura:** Predictor 1 Bit para if exterior

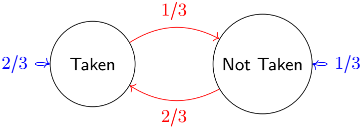



**Figura:** Predictor 1 Bit para ifs interiores

- Para el **if exterior**, tenemos $n\&3 \neq 0$ con $\Pr(\texttt{Taken}) = \frac{3}{4}$.
- Para los **ifs interiores**, dado que pasamos el if exterior, tenemos $n\&1$ y $n\&2$ con $\Pr(\texttt{Taken}) = \frac{2}{3}$.
- Branch misses en rojo en las figuras.

# Cadena de Markov y distribución estacionaria

Un proceso $X_0, X_1, \ldots$ con valores en $\{s_1, \ldots, s_K\}$, el conjunto de estados, es una Cadena de Markov sii existe una matriz $P \in \mathcal{M}_{K\times K}([0,1])$, fija, que define las probabilidades de transición

$$[P]_{i,j} = \Pr(X_{n+1} = s_j | X_n = s_i)\,,$$

para todo $n \geq 0$.

# Cadena de Markov y distribución estacionaria

Un proceso $X_0, X_1, \dots$ con valores en $\{s_1, \dots, s_K\}$, el conjunto de estados, es una Cadena de Markov sii existe una matriz $P \in \mathcal{M}_{K \times K}([0,1])$, fija, que define las probabilidades de transición

$$[P]_{i,j} = \Pr(X_{n+1} = s_j | X_n = s_i) \,,$$

para todo $n \geq 0$.

**Lema**

Sea $\mu^{(n)} = (\mu_1^{(n)}, \dots, \mu_K^{(n)})$ la distribución de $X_n$, i.e., $\Pr(X_n = s_i) = \mu_i^{(n)}$. Entonces tenemos la recurrencia matricial $\mu^{(n+1)} = \mu^{(n)} P$.

# Cadena de Markov y distribución estacionaria

Un proceso $X_0, X_1, \ldots$ con valores en $\{s_1, \ldots, s_K\}$, el conjunto de estados, es una Cadena de Markov sii existe una matriz $P \in \mathcal{M}_{K\times K}([0,1])$, fija, que define las probabilidades de transición

$$[P]_{i,j} = \Pr(X_{n+1} = s_j | X_n = s_i)\,,$$

para todo $n \geq 0$.

**Lema**

Sea $\mu^{(n)} = (\mu_1^{(n)}, \ldots, \mu_K^{(n)})$ la distribución de $X_n$, i.e., $\Pr(X_n = s_i) = \mu_i^{(n)}$. Entonces tenemos la recurrencia matricial $\mu^{(n+1)} = \mu^{(n)} P$.

**Definición**

Un vector $\pi = (\pi_1, \ldots, \pi_K)$ con $\pi_i \geq 0$ y $\sum \pi_i = 1$ es una distribución estacionaria para $P$ sii $\pi = \pi P$.

# Teorema Ergódico para Cadenas de Markov

Para asegurar que la distribución converge a una estacionaria $\mu^{(n)} \to \pi$, necesitamos algunas condiciones técnicas relacionadas con el digrafo de $P$.

# Teorema Ergódico para Cadenas de Markov

Para asegurar que la distribución converge a una estacionaria $\mu^{(n)} \to \pi$, necesitamos algunas condiciones técnicas relacionadas con el digrafo de $P$.

## Definición

– Una Cadena de Markov es **irreducible** sii existe un camino con probabilidad positiva entre cada par de estados.
– Una Cadena de Markov es **aperiódica** sii el máximo común divisor de la longitud de todos los ciclos es $1$.

# Teorema Ergódico para Cadenas de Markov

Para asegurar que la distribución converge a una estacionaria $\mu^{(n)} \to \pi$, necesitamos algunas condiciones técnicas relacionadas con el digrafo de $P$.

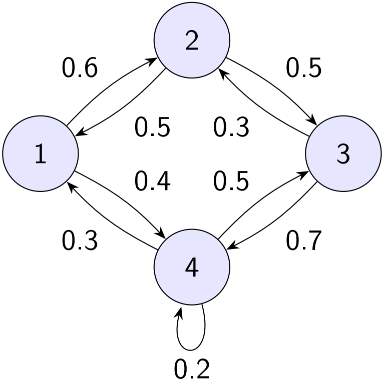
**Definición**

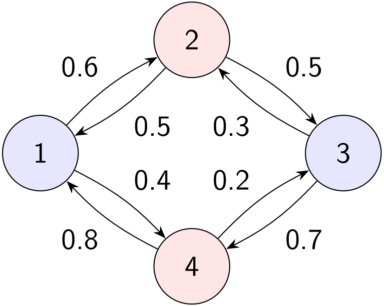
– Una Cadena de Markov es **irreducible** sii existe un camino con probabilidad positiva entre cada par de estados.
– Una Cadena de Markov es **aperiódica** sii el máximo común divisor de la longitud de todos los ciclos es $1$.

La condición de irreductibilidad permite afirmar que no tenemos nodos "transitorios", que no visitaremos más a partir de un cierto momento.

# Cadenas periódicas y aperiódicas

Una cadena aperiódica (izq.) y una periódica (der.):




– La aperiodicidad se cumple inmediatamente cuando tenemos *loops*.
– Una cadena periódica presenta periodicidades en $\mu^{(n)}$.

# Teorema Ergódico para Cadenas de Markov

**Teorema**

*Sea* $(X_0, X_1, \ldots)$ *una Cadena de Markov* **irreducible** *y* **aperiódica** *con matriz de transición* $P$ *y distribución inicial arbitraria* $\mu^{(0)}$*.*

*Existe una única distribución estacionaria* $\pi$ *tal que* $\mu^{(n)} \to \pi$*. Más aún,* $\pi$ *no depende de la elección de la distribución inicial* $\mu^{(0)}$*.*

En este caso $\pi$ es el único vector propio de $\lambda = 1$ para $P$, con $\sum \pi_i = 1$.

# Teorema Ergódico para Cadenas de Markov

**Teorema**

*Sea* $(X_0, X_1, \ldots)$ *una Cadena de Markov* **irreducible** *y* **aperiódica** *con matriz de transición* $P$ *y distribución inicial arbitraria* $\mu^{(0)}$.

*Existe una única distribución estacionaria* $\pi$ *tal que* $\mu^{(n)} \to \pi$. *Más aún,* $\pi$ *no depende de la elección de la distribución inicial* $\mu^{(0)}$.

En este caso $\pi$ es el único vector proprio de $\lambda = 1$ para $P$, con $\sum \pi_i = 1$.

Para contar las transiciones

**Proposición**

*En las hipótesis del teorema, para cada transición* $v = (s_i, s_j)$,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \mathbf{1}_{(X_k, X_{k+1})=v} = \pi_i P_{i,j} \,.$$

# Teorema Ergódico aplicado: 1 bit

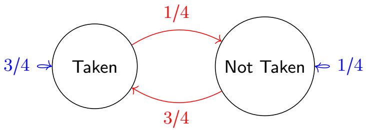



**Figura:** Predictor 1 Bit para if exterior

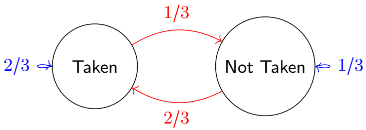
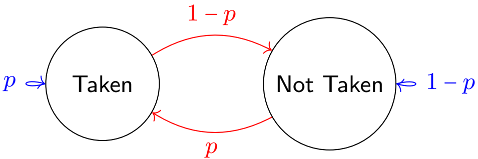



**Figura:** Predictor 1 Bit para ifs interiores

Sea entonces



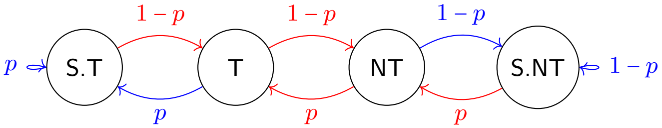

Basta estudiar $\pi = \pi(p)$, donde $1 = T$, $2 = NT$. En este caso

$$\pi = (p, 1 - p)\,.$$

Y para las transiciones en rojo tenemos la frecuencia $\alpha_1(p) := 2p(1-p)$.

# Teorema Ergódico aplicado: 2 bits

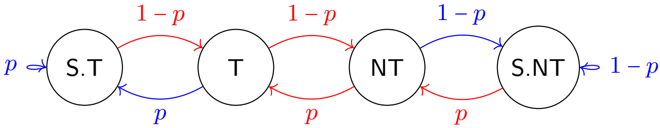
Para dos bits tenemos




Hay que calcular $\pi = \pi(p)$, donde $1 = S.T$, $2 = T$, $3 = NT$, $4 = S.NT$.

# Teorema Ergódico aplicado: 2 bits

Para dos bits tenemos



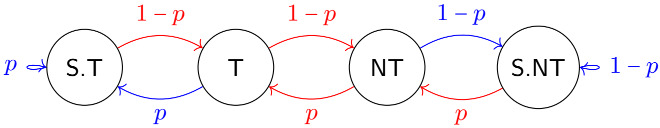

Hay que calcular $\pi = \pi(p)$, donde $1 = S.T$, $2 = T$, $3 = NT$, $4 = S.NT$.

En este caso [cálculo en pizarrón]

$$\pi_i = C \cdot \left(\frac{1-p}{p}\right)^{i-1}, \qquad C = \frac{1-\left(\frac{1-p}{p}\right)}{1-\left(\frac{1-p}{p}\right)^4} = \frac{p^3}{1-2p(1-p)}.$$

# Teorema Ergódico aplicado: 2 bits

Para dos bits tenemos




Hay que calcular $\pi = \pi(p)$, donde $1 = S.T$, $2 = T$, $3 = NT$, $4 = S.NT$.

En este caso [cálculo en pizarrón]

$$\pi_i = C \cdot \left(\frac{1-p}{p}\right)^{i-1}, \qquad C = \frac{1-\left(\frac{1-p}{p}\right)}{1-\left(\frac{1-p}{p}\right)^4} = \frac{p^3}{1-2p(1-p)}.$$

El conjunto de transiciones marcadas en rojo tiene frecuencia:

$$\alpha_2(p) = \pi_1(1-p) + \pi_2(1-p) + \pi_3 p + \pi_4 p = \frac{p(1-p)}{1-2p(1-p)}.$$

# Errores de predicción en la exponenciación sesgada

## Proposición (Simplificada)

*Sea* $N = 4^k$ *y consideremos* $n \in \{0, \ldots, N-1\}$ *uniforme.*
*La cantidad media de errores de predicción, cuando* $k \to \infty$*, con el predictor para* $i$*-bits es asintóticamente*

$$k \times \left(\alpha_i(3/4) + \frac{3}{4} \cdot 2 \cdot \alpha_i(2/3)\right), \qquad k = \frac{1}{2}\log_2 N\,.$$

---

El factor $\frac{3}{4}$ es la probabilidad de efectuar los ifs internos.

---

0 $\frac{25}{48} \approx 0{,}52$, $\frac{9}{20} = 0{,}45$, $\frac{1095}{2788} \approx 0{,}39$.

# Errores de predicción en la exponenciación sesgada

## Proposición (Simplificada)

*Sea* $N = 4^k$ *y consideremos* $n \in \{0, \ldots, N-1\}$ *uniforme.*
*La cantidad media de errores de predicción, cuando* $k \to \infty$*, con el predictor para* $i$*-bits es asintóticamente*

$$k \times \left(\alpha_i(3/4) + \tfrac{3}{4} \cdot 2 \cdot \alpha_i(2/3)\right), \qquad k = \tfrac{1}{2}\log_2 N\,.$$

El factor $\frac{3}{4}$ es la probabilidad de efectuar los ifs internos.

## Proposición (Auger,Nicaud,Pivoteau'2016)

*Sea* $N \to \infty$ *arbitrario y consideremos* $n \in \{0, \ldots, N-1\}$ *uniforme.*
*La cantidad esperada de errores de predicción en la exponenciación sesgada para los predictores saturados de 1, 2 y 3 bits es:*

$$M_{\textit{1 bit}}(N) \sim \log_2(N) \times \frac{25}{48}\,,\; M_{\textit{2 bit}}(N) \sim \log_2(N) \times \frac{9}{20},\; M_{\textit{3 bit}}(N) \sim \log_2(N) \times \frac{1095}{2788}\,.$$

[0] $\frac{25}{48} \approx 0{,}52,\ \frac{9}{20} = 0{,}45,\ \frac{1095}{2788} \approx 0{,}39.$

# Para aprender más


- *Olle Häggström*,
  Finite Markov Chains and Algorithmic Applications.
  London Mathematical Society Student Texts 52.

- *Nicolas Auger, Cyril Nicaud, y Carine Pivoteau*,
  Good Predictions Are Worth a Few Comparisons.
  `https://www-igm.univ-mlv.fr/~nicaud/articles/stacs16.pdf`

- *Cyril Nicaud, Carine Pivoteau y Stéphane Vialette*
  Branch Prediction Analysis of Morris-Pratt and Knuth-Morris-Pratt Algorithms.
  `https://arxiv.org/abs/2503.13694`

- *Conrado Martínez, Markus E. Nebel y Sebastian Wild*
  Analysis of branch misses in quicksort.
  `https://dl.acm.org/doi/10.5555/2790216.2790227`