# Introducción a algoritmos probabilistas:
# El problema de "Selección de Secretarie"

Una empresa quiere contratar a un asistente.
Pide a una empresa de selección de personal que le envíe $N$ candidates, uno cada día. $t=1,\dots,N$

En el día $t$ la empresa:

(1) Paga el valor $e$ de entrevistar al candidato y

(2) Si el candidato es mejor que todos los anteriormente vistos paga el valor $K$ de contratar a este nuevo candidato (y de despedir al anterior)

Suponga $K \gg e$

Asumimos que la empresa tiene un "test de aptitud" que le permite ranquear totalmente a los candidatos, sin repeticiones

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | ... | N |
| A(1) | A(2) | | ... | A(N) |

Los "scores" $A(i)$ no importan, sino sólo el orden relativo.

Sea $\text{Rank}: \{1,..,N\} \longrightarrow \{1..N\}$

$\text{Rank}(i)$ = "ranking del candidato $i$ según $A(\cdot)$"

Por ejemplo $\sigma(j)=1 \iff j$ es el mejor de todos los candidatos

<u>Problema:</u> ¿Cuánto le cuesta a la compañía llevar a cabo esta estrategia?

<u>Sol:</u> El costo depende mucho del ranking en el que vengan. Para $N$ candidatos

| candidatos | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| $Rk_1$ | 1 | 2 | 3 | 4 | 5 | $\longrightarrow K + Ne$ |
| $Rk_2$ | 5 | 4 | 3 | 2 | 1 | $\longrightarrow KN + Ne$ |

En el peor de los casos $KN + eN$

La empresa no sabe en qué Ranqueo van a venir los N candidatos.

Permutaciones de n elementos.

| Candidato | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Rank | 2 | 3 | 1 | 5 | 4 |

$\} \leftarrow \sigma \in S_n$

Podemos representar nuestra ignorancia diciendo que "$\sigma$ es una permutación ALEATORIA UNIFORMEMENTE DISTRIBUIDA en $\Omega = S_N$"

Más precisamente:

$$\Omega = S_n = \text{"Permutaciones de } N \text{ elementos"}$$

$$\mathbb{P}(\{\sigma\}) = \frac{1}{|S_n|} = \frac{1}{N!} \leftarrow \text{medida de probabilidad uniforme en } S_N$$

Si los candidatos tienen ranking $\sigma$ el costo es $C(\sigma) = Ne + K \left| \{ i : \forall t < i \ [\sigma(t) > \sigma(i)] \} \right|$

Cuando $i$ cumple "esto" contratamos a $i$

$C : S_N \longrightarrow \mathbb{R}$ es una variable aleatoria.

$$\sigma \quad 3\ 5\ 2\ 4\ 1\ 6 \rightarrow \underbrace{6e + K \cdot 2}_{C(\sigma)}$$

¿Cuál es el costo esperado?

** Si una variable aleatoria $Z$ puede escribirse como suma de variables aleatorias $Y_1, \dots, Y_N$

$$Z = \sum_{i=1}^{N} Y_i$$

entonces

$$\mathbb{E}[Z] = \sum_{i=1}^{N} \mathbb{E}[Y_i]$$

SIEMPRE, sin importar qué relaciones hay entre las $Y_i$'s

En nuestro caso definimos la variable aleatoria $Y^{(i)} : S_N \to \mathbb{R}$

$$Y^{(i)}(\sigma) = \begin{cases} 1, & \text{si deberíamos contratar al candidato } i \text{ según el ranqueo } \sigma. \\ 0, & \text{si no} \end{cases}$$

Como $Y^{(i)}$ solo asume dos valores

$$\mathbb{E}[Y^{(i)}] = \mathbb{P}\{\sigma \in S_n : \sigma(i) < \sigma(t) \ \forall t \leq i-1\}$$

**Lema:** $\mathbb{P}\{\sigma\in S_n : \sigma(i)<\sigma(t)\ \forall t\leq i-1\} = \frac{1}{i}$

$\mathbb{P}\{\sigma\in S_n :$ $\sigma(i)$ es el número más pequeño entre $\sigma(1),\sigma(2),..,\sigma(i)$ $\}$

Defina $B_j = \{\sigma\in S_n :$

**Dem:** El mínimo valor de $(\sigma(1),..,\sigma(i))$ se alcanza en la componente $j$ $\}$

(1) $B_1, B_2,.., B_i$ son disjuntos y cubren todo

(2) $S_n = B_1\cup\dots\cup B_i$.

Precomponiendo $\sigma$ con la transposición $(1j)$ Construimos una biyección entre $B_1$ y $B_j$ luego $\mathbb{P}(B_1)=\dots=\mathbb{P}(B_i)$ (3)

Hay la misma cantidad de $\sigma$ con mínimo en 1 que con mínimo en $j$.

(1)(2) y (3) implican que $\mathbb{P}(B_i)=\frac{1}{i}$

Ahora definamos $X=\sum_{i=1}^{N} Y^{(i)}$

y notemos que

$X(\sigma)$ = "# de candidatos que contratamos si vienen con ranqueo $\sigma$"

y que

$$C(\sigma) = Ne + K X(\sigma)$$

**Teorema:** El costo esperado de la estrategia es

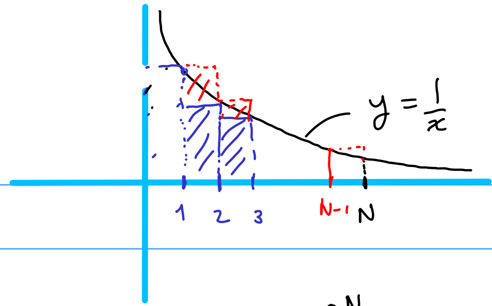
$$\mathbb{E}[C] = \Theta(K\log(N)) + Ne$$

**Dem:** Del razonamiento anterior, $X=\sum_{i=1}^{N} Y^{(i)}$

$$\mathbb{E}[C]=\mathbb{E}\left[Ne + K\sum_{i=1}^{N} Y^{(i)}\right] =$$

$$= Ne + K\sum_{i=1}^{N}\mathbb{E}[Y^{(i)}] = Ne + K\left(\sum_{i=1}^{N}\frac{1}{i}\right) =$$

¿Cómo se comporta?




Según dibujo:

$$\underbrace{\text{Area Azúl}}_{\frac{1}{2}+\frac{1}{3}+\dots+\frac{1}{N}} \leq \int_1^N \frac{1}{x}\,dx \leq \text{Área Rojo} = 1+\frac{1}{2}+\frac{1}{3}+\dots+\frac{1}{N-1}$$

$$\sum_{i=1}^{N} \frac{1}{i} \leq 1 + \int_1^N \frac{1}{x}\,dx = 1 + \log(N) - \log(1)$$

$$\underbrace{\int_1^N \frac{1}{x}\,dx}_{=\ \log(N)} \leq \sum_{i=1}^{N} \frac{1}{i} \quad \Rightarrow \quad \sum_{i=1}^{N} \frac{1}{i} \in \Theta(\log(N))$$

Luego $\mathbb{E}[C] = Ne + \Theta(K\log(N))$ ∎

② Aleatoriedad como herramienta:

En el problema anterior hicimos un análisis usando la probabilidad uniforme en $S_N$ como una representación de nuestra ignorancia. Ahora vamos a utilizarla para mejorar nuestros costos. Concretamente supongamos que

La empresa de selección nos da la lista de candidatos

$$① \begin{bmatrix} A \\ B \\ C \\ \vdots \\ Z \end{bmatrix} \xrightarrow[\text{nosotros aleatorizamos la lista}]{②} \begin{bmatrix} \sigma(A) \\ \sigma(B) \\ \vdots \\ \sigma(Z) \end{bmatrix}$$

$\sigma \in S_N$ uniforme

nosotros los entrevistamos en ESTE orden. Como $\sigma$ es uniforme, el Ranking resultante tambien lo es así que NUESTRA suposición de ranqueo uniforme ahora es verdadera y

tendremos $\mathbb{E}[C] = ne + K\,O(\log(N))$

Para poder hacer esto, tenemos que poder "generar" una permutación de $S_N$ con la medida uniforme.

En la clase anterior vimos como generar un elemento de $\Omega$ finito cualquiera con una $\mathbb{P}$ dada. El problema es que $S_N$ es ENORME, con $n!$ elementos así que nuestro método general no sirve y requerimos algún mecanismo...

[PERMUTACIÓN UNIFORME MEDIANTE SORTING]

Algoritmo de re-ordenamiento quasi aleatorio:

**NOTA:** Todo algoritmo randomizado asume que podemos generar números aleatorios uniformes independientes

Random $(i,j)$ ← entero uniforme entre $\{i, i+1, \ldots, j-1, j\}$

Es un tema interesante y profundo cómo construir tales generadores. Por ahora asumámoslo.

**Algoritmo:**

INPUT: Vector $A$ con $N$ componentes

OUTPUT: Vector $C = \sigma(A)$ con $N$ componentes para $\sigma \in S_n$ aleatoria con dist. uniforme

```
P = [ ]    ← Vector de "Prioridades"
for i in range(len(A)):
        P[i] = Range (1, N^3)
PSorted = numpy.argsort(P)
C = [A[PSorted[i]] for i in range(N)]
```

ORDENA A de acuerdo a los tamaños de P

Tiene un time-complexity $O(N\log(N))$

[For-FREE primitive]

**Lema:** Si todas las prioridades son distintas entonces la permutación $\sigma$ resultante tiene distribución uniforme.

$\mathbb{P}(\sigma \mid \vec{P}$ tiene entradas distintas$)$

**Dem:** La distribución de probabilidad de las prioridades es invariante respecto al orden de las componentes. Como resultado de esto

Esto es suficiente.

$\boxed{\text{la probabilidad de que el mínimo ocurra en cualquier localización } i \text{ es la misma.}}$

Usando esto definiremos:

$X^{(i)} = \{$ i-ésima prioridad se asigne en la i-ésima componente $\}$

$$\mathbb{P}\{\text{generar la identidad}\} = \mathbb{P}\{X^{(1)} \cap X^{(2)} \cap \dots \cap X^{(N)}\}$$

$$= \mathbb{P}\{X^{(1)}\}\,\mathbb{P}\{X^2 \mid X^1\} \cdot \mathbb{P}\{X^3 \mid X^1 \cap X^2\} \cdot \dots \cdot \mathbb{P}\{X^{(N)} \mid X^1 \cap \dots \cap X^{N-1}\}$$

$$= \frac{1}{N} \cdot \frac{1}{(N-1)} \cdot \frac{1}{(N-2)} \cdot \dots \cdot \frac{1}{2} \cdot 1 = \frac{1}{N!}$$

Por invariatza ese es el valor de las demás tambien. ☒

Aclaración: El algoritmo muestrea las prioridades $\vec{p}$ y luego chequea si hay alguna igualdad.

Si: hay alguna igualdad. Descarte todo y vuelva a empezar

NO: Complete el algoritmo y produzca $\sigma$

$\mathbb{P}\{\sigma \in S_n \mid \vec{p} \text{ NO tiene igualdades}\}$ es uniforme. Este es el contenido preciso de Lema (✱)

Qué tan probable es que el algoritmo tenga que volver a empezar? Ver ejercicio ....

$\mathbb{P}\{(p_1, \dots, p_n) \text{ no tiene repeticiones}\}$.

③ "Selección de Secretarias ONLINE"

El término ONLINE no quiere decir en internet sino quiere decir "en tiempo real", es decir es la siguiente variante del problema: (Ver página siguiente)
⋮

Vemos N candidatos en orden secuencial,

Una empresa quiere contratar UN SOLO candidato tomando SOLO UNA decisión de contratación con el siguiente requerimiento extra: "Justamente después de la entrevista TENEMOS que DECIDIR inmediatamente si contratamos al candidato o no (e informarle)" (la decisión es instantánea = ON-LINE y los datos aparecen en orden secuencial)

significado original + antiguo del término

La empresa usará su "test de aptitud" A para comparar candidatos $i < j \iff A(i) > A(j)$.

Una estrategia posible es la siguiente:

(1) Fijamos un número $k \leq N$

(2) Entrevistamos y rechazamos los primeros k candidatos recordando su máximo score $\overline{S}$

(3) Para $i \geq k+1$ si $Score(i) > \overline{S} \Rightarrow$ Contratamos a i

Si $\forall i \geq k+1$ $Score(i) < \overline{S} \Rightarrow$ Contratamos al último, el candidato N.

Esta estrategia depende de K ¿Cómo escoger K de manera óptima? Hay varios criterios de optimalidad. Nos enfocaremos en escoger al mejor candidato:

<u>Teorema:</u> Si S es el evento de escoger al mejor candidato entonces

$$\frac{k}{n}\left[\ln(n) - \ln(k)\right] \leq \mathbb{P}(S) \leq \frac{k}{n}\left(\ln(n-1) - \ln(k-1)\right)$$

Si $k \approx \frac{n}{e}$ tendremos la/el mejor con

$$\mathbb{P}(S) \geq \frac{1}{e} \approx \underbrace{0.367879}_{\text{más de } \frac{1}{3} \text{ de las veces!!}}$$

$0.367 \cdot n$

<u>Dem:</u> $S = \{\sigma :$ escogemos el mejor$\}$

$S_i = \{\sigma :$ escogemos el mejor y este está en posición $i\}$

$S = \bigsqcup_i S_i$ y $S_i = \emptyset$ pra $i \leq k$ pues este nunca se escoge.

$$\mathbb{P}(S) = \sum_{i=k+1}^{N} \mathbb{P}(S_i)$$

$S_i$ = Best applicant en posición $i$ y No escogemos $j$ en $k+1, \dots, i-1$ (porque $A(j) < \bar{S}$, $j \in \{k+1, \dots, i-1\}$ $= B_i \cap O_i$

$O_i$ depende de los scores $1, \dots, i-1$ o mejor, de su ordenamiento relativo = Máximo esta en posiciones $1, 2, \dots, k$

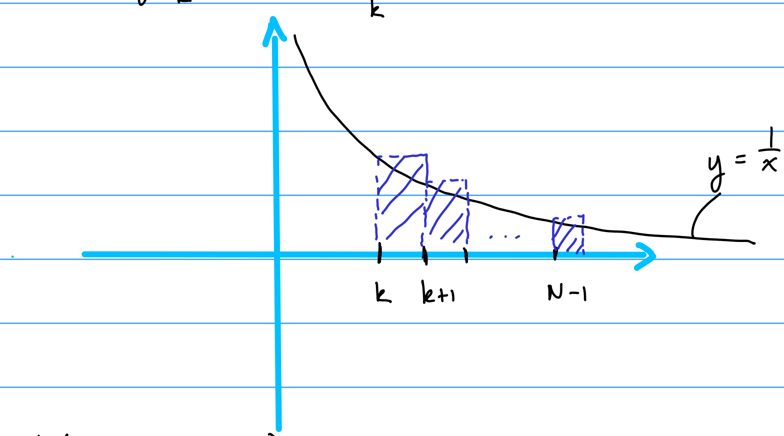
$B_i$ depende de si el $A(i)$ es más grande que todos los demás.

Así que $O_i$ y $B_i$ son independientes luego

$$\mathbb{P}(S_i) = \mathbb{P}(B_i)\,\mathbb{P}(O_i) = \frac{1}{N}\,\frac{k}{(i-1)}$$

$$\mathbb{P}(S) = \sum_{i=k+1}^{N} \frac{k}{N(i-1)} = \frac{k}{N}\sum_{i=k+1}^{N}\frac{1}{i-1}$$

$$= \frac{k}{N}\sum_{i=k}^{N-1}\frac{1}{i} \geq \frac{k}{N}\int_k^N \frac{1}{x}\,dx = \frac{k}{N}\left[\log(N) - \log(k)\right]$$




$$\frac{k}{N}\left(\log(N) - \log(k)\right) = \varphi(k)$$

$$\varphi'(k) = \frac{1}{N}\left(\log(N) - \log(k)\right) + \frac{k}{N}\left[-\frac{1}{k}\right] = 0$$

$$\frac{\log(N)}{N} - \frac{\log(k)}{N} - \frac{1}{N} = 0$$

$$\log(k) = \log(N) - 1$$

$k^* = \frac{N}{e}$ y reemplazando $\mathbb{P}(S) \geq \frac{1}{e}$