

Algoritmos I

Fundamentos de Computación

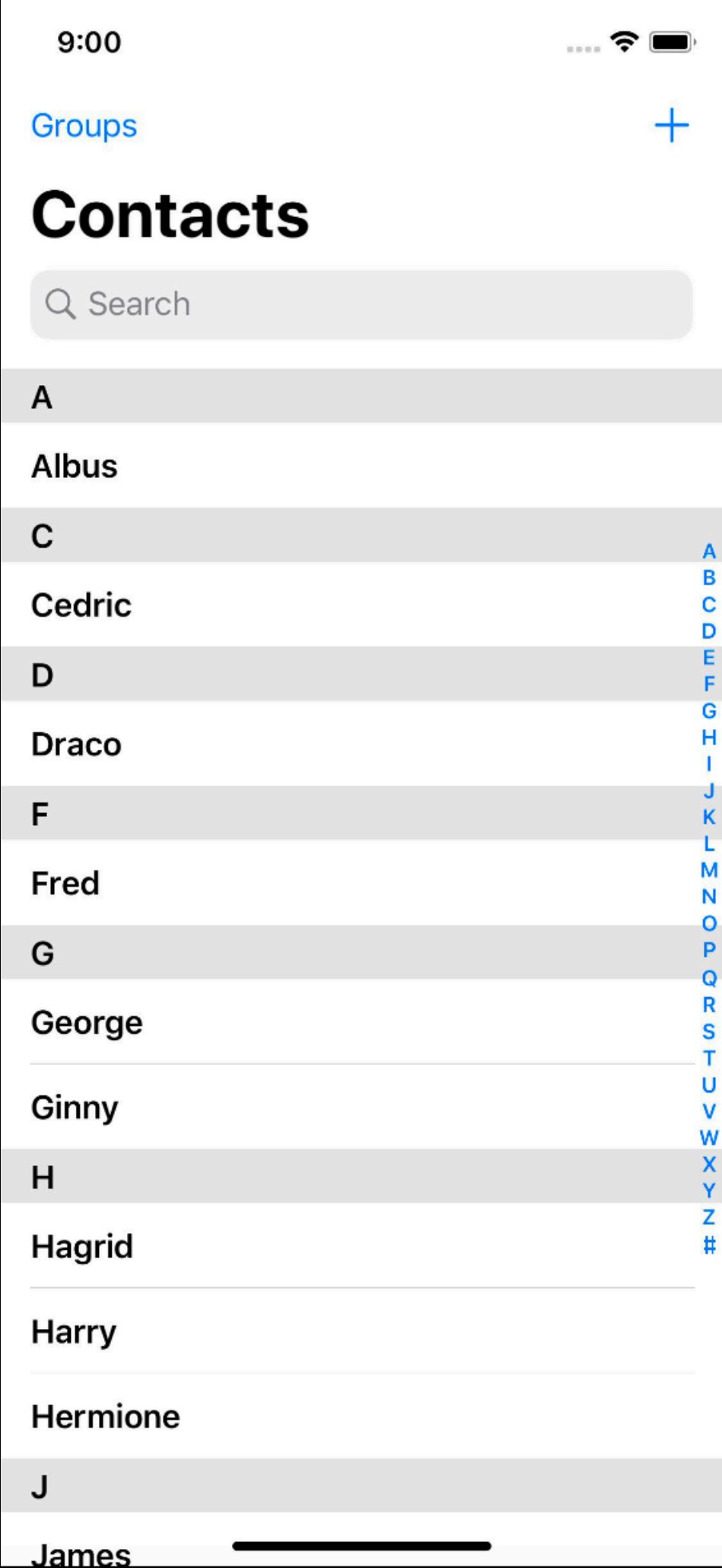
Diego Caro

2021-1

Ciencias de la computación: resolución de problemas

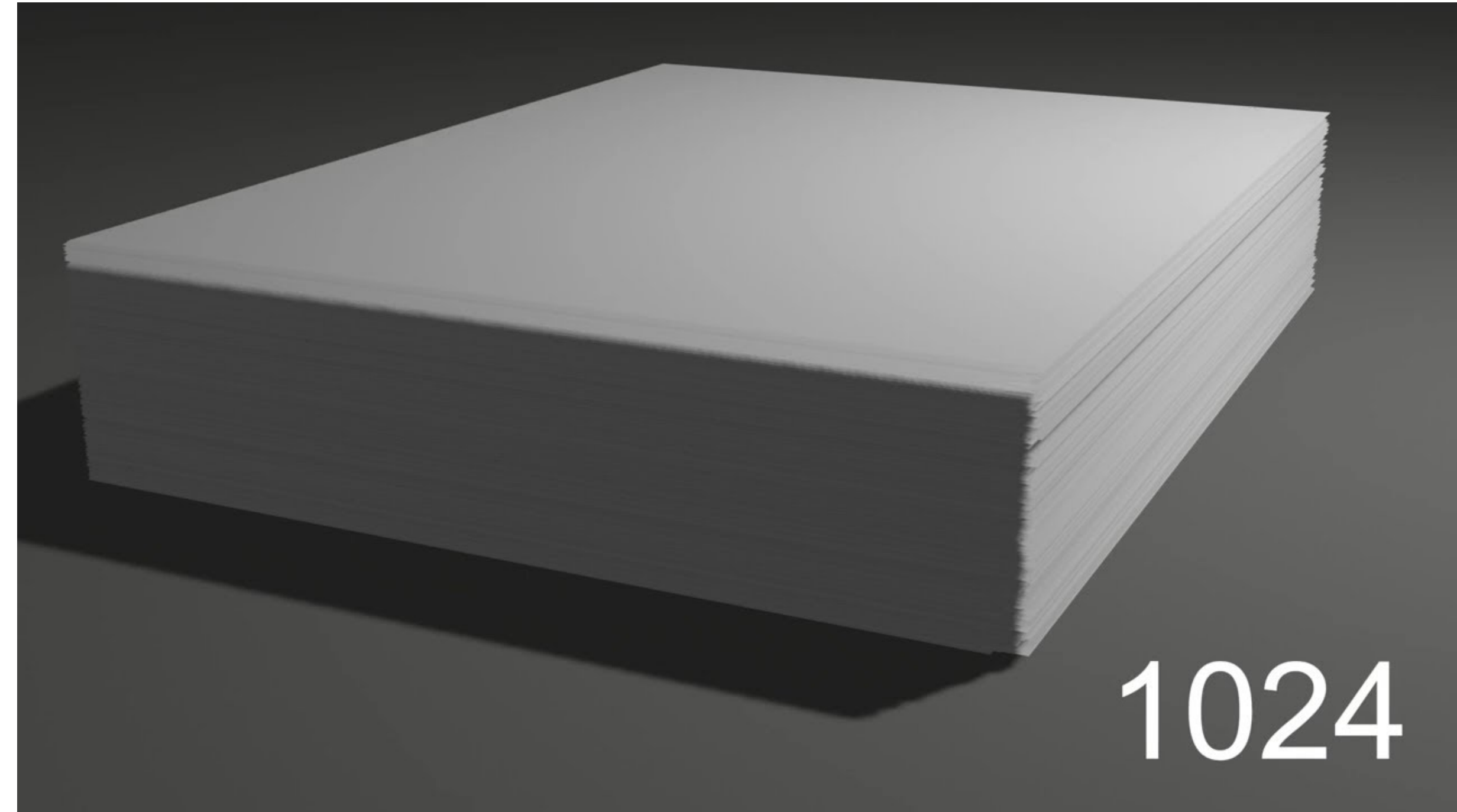






Algoritmo 1

- Partir desde la primer página, ver una página a la vez hasta encontrar a quién buscamos.
- Cada página la puedo revisar en 10s.
- La guía tiene 1024 páginas.



¿Cuánto tiempo me tomaría encontrar el número de teléfono de alguien?

**¿Podríamos acelerar la búsqueda
saltándonos dos páginas a la vez!**

¿Podríamos acelerar la búsqueda saltándonos dos páginas a la vez?

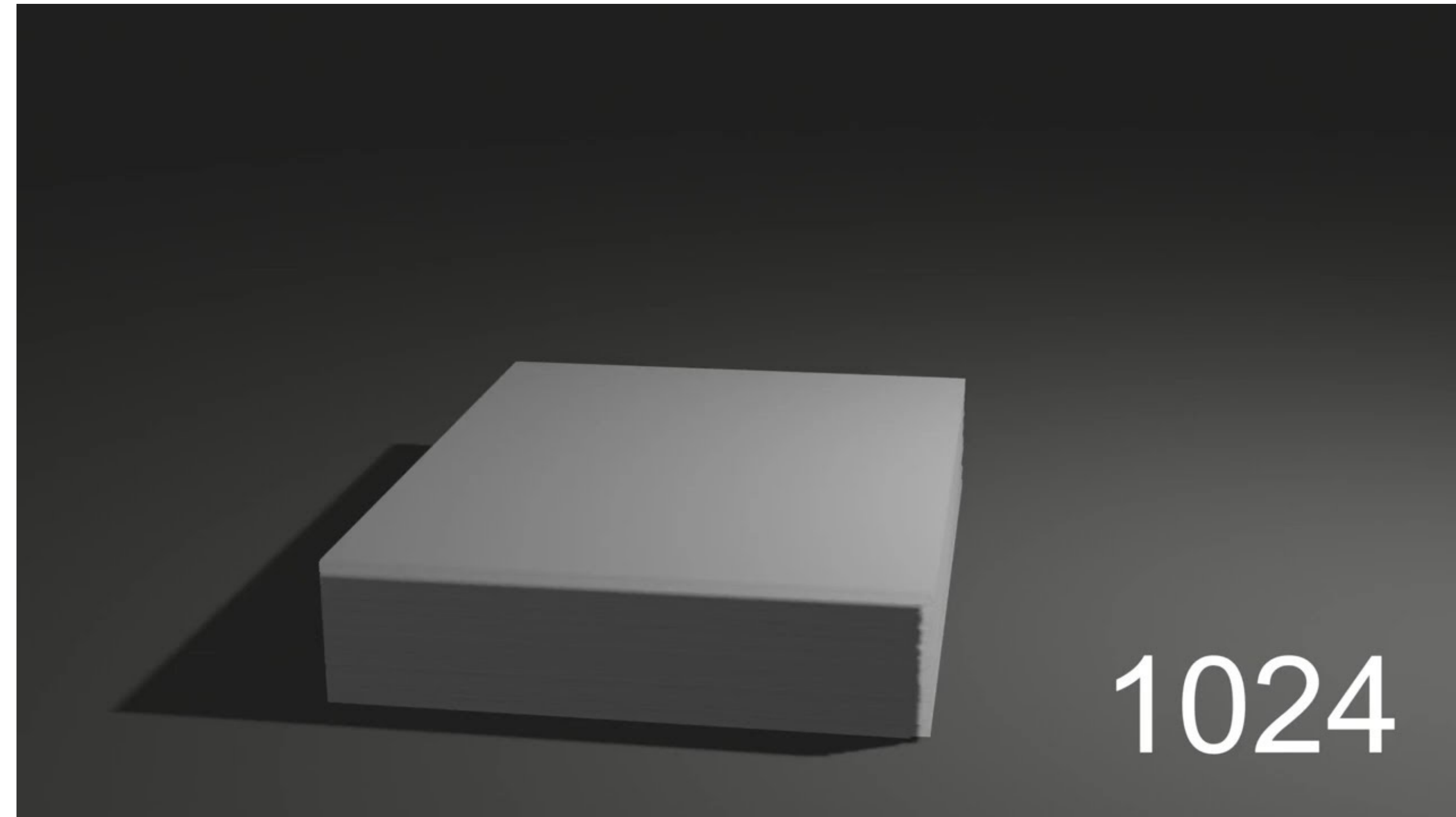
No sería correcto, porque

podríamos saltarnos el nombre que buscamos (podría estar en esa página que saltamos!).

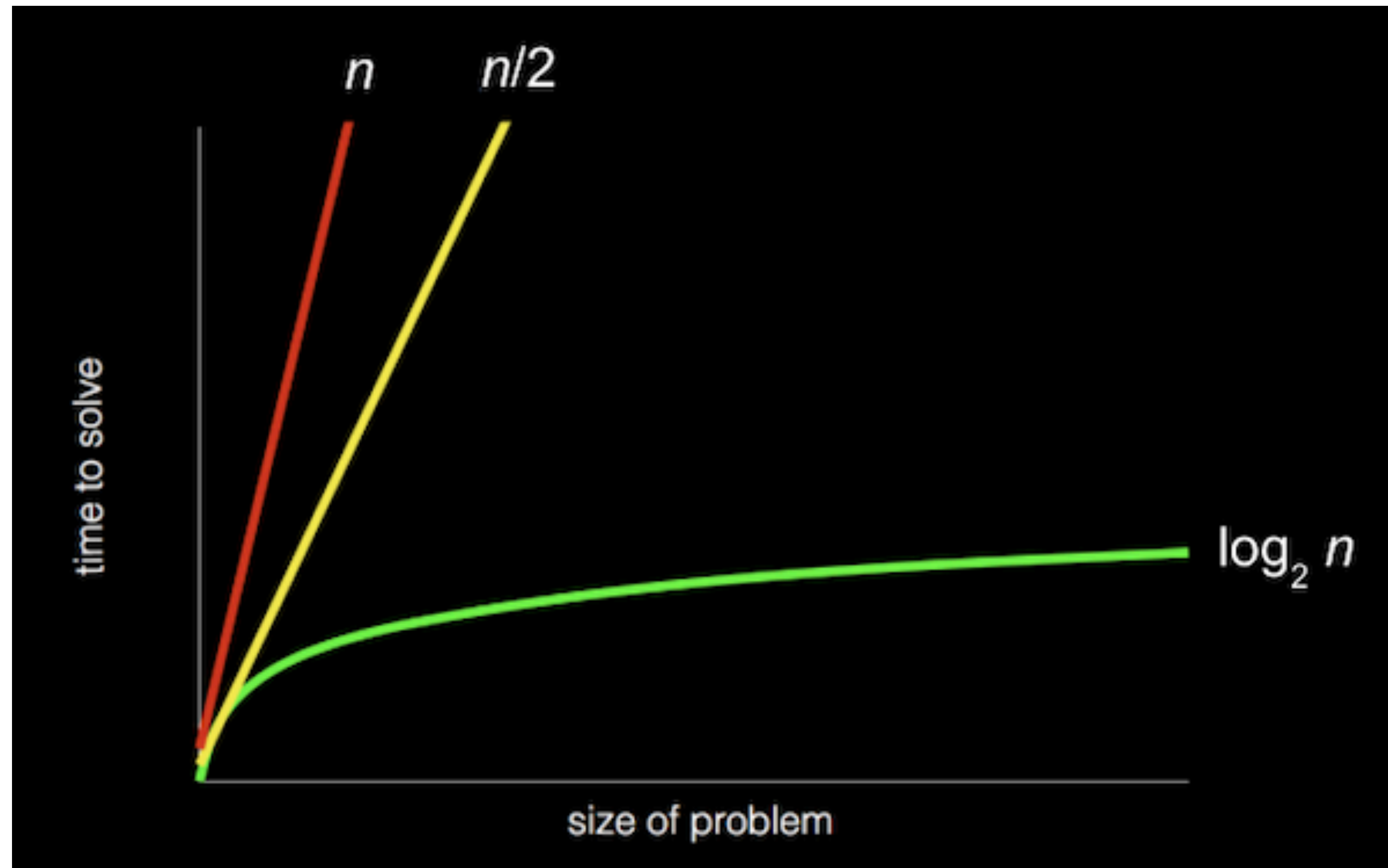
Podríamos **solucionarlo** volviendo **una página atrás** cada vez que “nos pasemos”. Ahí estaría resuelto el problema.

Algoritmo 3

- Abrir la guía por la mitad. Decidir si el nombre está en la primera mitad (o segunda mitad), así reducimos el tamaño del problema en 2. Recordar que la guía está ordenada por nombre.
- Seguir usando este procedimiento hasta que encontremos el nombre que buscamos.











Si la guía tiene 1024 páginas, en 10 pasos podemos encontrar cualquier nombre :)



Intermezzo: Juego de las 8 preguntas

- Escoge un número entero entre 0 y 128.
- En al menos de 8 preguntas sabré qué número escogiste

Intervalo	Tamaño Intervalo	Pregunta	Respuesta
	128	< 64?	no
	64	< 96?	si
	32	< 80?	si
	16	< 72?	no
	8	< 76?	no
	4	< 78?	si
	2	< 77?	no
	1	Escogiste el número 77!	

Idea para búsqueda en guía telefónica

- 1 **Tomar** la guía telefónica
- 2 **Abrir** la mitad de la guía telefónica
- 3 **Mientras** reviso la guía:
 - 4 Si persona está en la página actual:
 - 5 **Llamar** a persona
 - 6 **Dejar** de revisar guía y **Salir**
 - 7 Si persona está antes en la guía
 - 8 **Abrir** la mitad izquierda del libro
 - 9 Si persona está después en la guía
 - 10 **Abrir** la mitad derecha del libro
- 11 Persona **no está** en la guía

Pseudocódigo formal

```
1. def Buscar(X, Y, nombre):
2.   inf = 1 // límite inferior de la búsqueda
3.   sup = N // límite superior de la búsqueda
4.   pos = -1 // posición donde está nombre
5.   while inf <= sup and pos == -1:
6.     i = inf + (sup-inf)/2
7.     if nombre > Xi:
8.       inf = i + 1
9.     elif nombre < Xi:
10.      sup = i - 1
11.    else:
12.      pos = i
13.  if pos != -1: retornar Yi
14.  else: retornar "No encontrado"
```



X es una lista de nombres X1, X2, ..., XN
Y es una lista de teléfonos Y1, Y2, ..., YN
Tal que el teléfono de la persona Xi está en Yi
nombre es la persona que estamos buscando

Y:

9939382	3123123	7123315	9488943	9584434	9348934	1823494	239923
---------	---------	---------	---------	---------	---------	---------	--------

X:

Andrea	Caro	John	Luis	Ricardo	Rosario	Valeria	Yerka
1	2	3	4	5	6	7	8

↑

inf

↑

i

↑

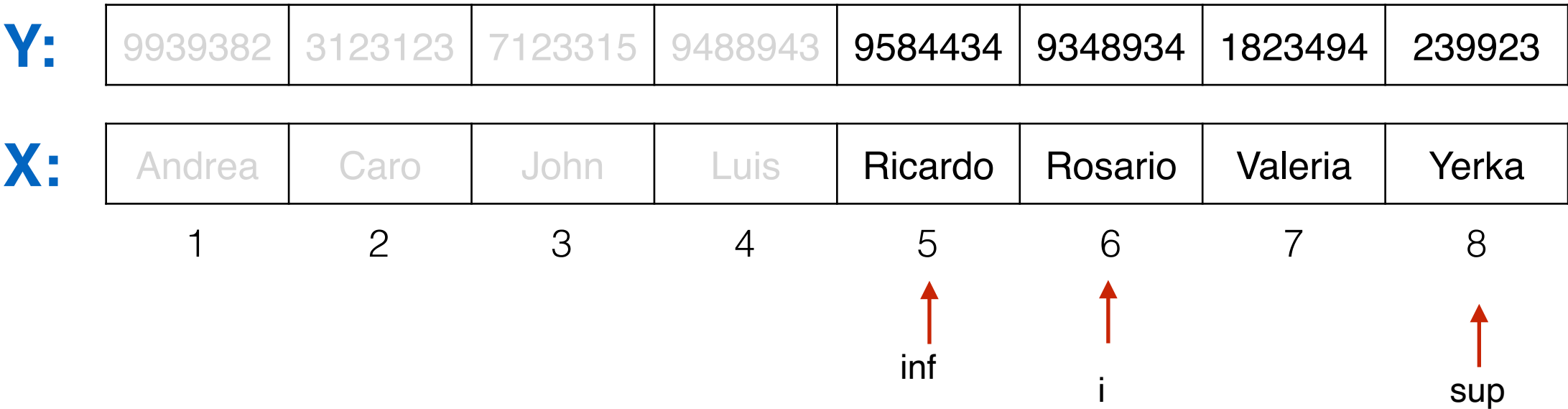
sup

inf	sup	i	X _i	nombre	nombre > X _i	nombre < X _i
1	8	4	Luis	Valeria	si	no

Pseudocódigo formal

X es una lista de nombres X_1, X_2, \dots, X_N
Y es una lista de teléfonos Y_1, Y_2, \dots, Y_N
Tal que el teléfono de la persona X_i está en Y_i
nombre es la persona que estamos buscando

```
1. def Buscar(X, Y, nombre):
2.   inf = 1 // límite inferior de la búsqueda
3.   sup = N // límite superior de la búsqueda
4.   pos = -1 // posición donde está nombre
5.   while inf <= sup and pos == -1:
6.     i = inf + (sup-inf)/2
7.     if nombre > Xi:
8.       inf = i + 1
9.     elif nombre < Xi:
10.      sup = i - 1
11.    else:
12.      pos = i
13.  if pos != -1: retornar Yi
14.  else: retornar "No encontrado"
```



inf	sup	i	X _i	nombre	nombre > X _i	nombre < X _i
1	8	4	Luis	Valeria	si	no
5	8	6	Rosario	Valeria	si	no

Pseudocódigo formal

X es una lista de nombres X_1, X_2, \dots, X_N
Y es una lista de teléfonos Y_1, Y_2, \dots, Y_N
Tal que el teléfono de la persona X_i está en Y_i
nombre es la persona que estamos buscando

```
1. def Buscar(X, Y, nombre):
2.   inf = 1 // límite inferior de la búsqueda
3.   sup = N // límite superior de la búsqueda
4.   pos = -1 // posición donde está nombre
5.   while inf <= sup and pos == -1:
6.     i = inf + (sup-inf)/2
7.     if nombre > Xi:
8.       inf = i + 1
9.     elif nombre < Xi:
10.      sup = i - 1
11.    else:
12.      pos = i
13.  if pos != -1: retornar Yi
14.  else: retonar "No encontrado"
```

Y:

9939382	3123123	7123315	9488943	9584434	9348934	1823494	239923
---------	---------	---------	---------	---------	---------	---------	--------

X:

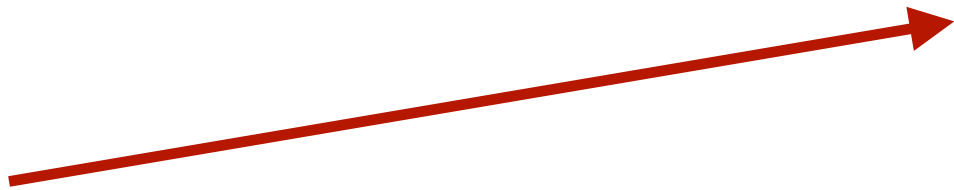
Andrea	Caro	John	Luis	Ricardo	Rosario	Valeria	Yerka
1	2	3	4	5	6	7	8

						i	sup
						↑	↑
						inf	

inf	sup	i	X _i	nombre	nombre > X _i	nombre < X _i
1	8	4	Luis	Valeria	si	no
5	8	6	Rosario	Valeria	si	no
7	8	7	Valeria	Valeria	no	no

Pseudocódigo formal

```
1. def Buscar(X, Y, nombre):
2.   inf = 1 // límite inferior de la búsqueda
3.   sup = N // límite superior de la búsqueda
4.   pos = -1 // posición donde está nombre
5.   while inf <= sup and pos == -1:
6.     i = inf + (sup-inf)/2
7.     if nombre > Xi:
8.       inf = i + 1
9.     elif nombre < Xi:
10.      sup = i - 1
11.    else:
12.      pos = i
13.  if pos != -1: retornar Yi
14.  else: retonar "No encontrado"
```



X es una lista de nombres X1, X2, ..., XN
Y es una lista de teléfonos Y1, Y2, ..., YN
Tal que el teléfono de la persona Xi está en Yi
nombre es la persona que estamos buscando

Y:

9939382	3123123	7123315	9488943	9584434	9348934	1823494	239923
---------	---------	---------	---------	---------	---------	---------	--------

X:

Andrea	Caro	John	Luis	Ricardo	Rosario	Valeria	Yerka
1	2	3	4	5	6	7	8

						↑	↑
						i	sup
						inf	

inf	sup	i	X _i	nombre	nombre > X _i	nombre < X _i
1	8	4	Luis	Valeria	si	no
5	8	6	Rosario	Valeria	si	no
7	8	7	Valeria	Valeria	no	no

nombre encontrado
en posición 7
:)

Búsqueda binaria: análisis

- ¿Cuántas comparaciones $X_i <, >, =$ realiza la búsqueda binaria?
 - Para una lista X de tamaño N : hace una comparación, y luego busca en una lista más pequeña de tamaño $N/2$.
 - $N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \rightarrow N/16 \rightarrow \dots \rightarrow 2 \rightarrow 1$
- P: ¿Cuántas veces se puede dividir un número hasta llegar a 1?
- R: $\log_2 N$

1
2 → 1
4 → 2 → 1
8 → 4 → 2 → 1
16 → 8 → 4 → 2 → 1
32 → 16 → 8 → 4 → 2 → 1
64 → 32 → 16 → 8 → 4 → 2 → 1
128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
512 → 256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
1024 → 512 → 256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1

Acknowledges

- Idea de agenda telefónica adaptada de curso CS50, lecture 0, slide 95 - 113. Disponible en <https://cdn.cs50.net/2020/fall/lectures/0/lecture0.pdf>
- Búsqueda binaria adaptada de Capítulo 4.2 “Computer Science, An Interdisciplinary Approach”, disponible en <https://introcs.cs.princeton.edu/java/42sort/>