Corrección de *quicksort*

quicksort termina:

- partimos de la base que *partition* termina

quicksort ordena:

- partimos de la base que *partition* es correcto

```
quicksort(A, i, f):
    if f - i ≥ 0:
        p = partition(A, i, f)
        quicksort(A, i, p-1)
        quicksort(A, p+1, f)
```

Inicialmente, *quicksort* es llamado con los índices

$$i = 0 \quad \text{y} \quad f = n{-}1 \quad \rightarrow \quad d = f - i = n - 1$$

Si $d < 0$, entonces *quicksort* termina; en otro caso, es llamado recursivamente sobre dos nuevos intervalos:

$$d_1 = p - 1 - i < d \qquad d_2 = f - p - 1 < d \qquad (\ i \leq p \leq f\ )$$

Es decir, en cada nuevo nivel de la recursión d disminuye al menos en 1:

→ la profundidad máxima de la recusrsión es $n - 1$

→ *quicksort* termina

partition es correcto →

- escoge un pivote
- separa el intervalo en dos: los elementos menores que el pivote quedan a su izquierda, y los elementos mayores que el pivote quedan a su derecha
- … y retorna el índice *p* en que queda el pivote

Observación: al terminar *partition*, **el elemento elegido como pivote queda en su posición ordenada en el arreglo**

Así, basta argumentar que cuando *quicksort* termina, todos los elementos del arreglo original han sido pivotes alguna vez:

- verdadero → las llamadas recursivas a *quicksort* sólo terminan cuando el intervalo correspondiente es vacío

Complejidad de *quicksort*

Peor caso:

- cuando la partición es desbalanceada al máximo
- el elemento elegido como pivote resulta ser el menor o el mayor del intervalo → el valor del índice *p* es igual a *i* o *f*

Mejor caso (análisis muy similar al caso de *mergesort*):

- cuando la partición es perfectamente balanceada
- el pivote resulta ser (prácticamente) la mediana de los elementos del intervalo correspondiente → $p \approx (i + f)/2$

Caso promedio:

- ¿? → **próximas diapositivas**

Sea $C(n)$ el número total de comparaciones que hace *quicksort* para un arreglo de n elementos:

`if A[k] < p:` —en la diap. #21 de “04. quicksort”

En la primera llamada, *partition* hace n–1 comparaciones (i = 0, f = n–1) y luego hay dos llamadas recursivas:

- supongamos que al terminar *partition*, el pivote queda en la q-ésima posición de A
- una llamada recursiva es sobre los q–1 elementos que quedan a la izquierda del pivote, y la otra es sobre los n–q elementos que quedan a su derecha

Entonces $C(n) = n - 1 + R(n)$, en que $R(n)$ es el número de comparaciones debido a las dos llamadas recursivas

Como el pivote puede quedar en cualquiera de las *n* posiciones —suponemos que con la misma probabilidad— entonces *R*(*n*) debe calcularse como el promedio de todos los casos posibles:

$$C(n) = n - 1 + \overbrace{\frac{1}{n} \sum_{q=1}^{n} \left(C(q-1) + C(n-q)\right)}^{R(n)}$$

… con $C(1) = C(0) = 0$ y notando que

$$C(0) + C(1) + \cdots + C(n-1) = C(n-1) + \cdots + C(1) + C(0):$$

$$C(n) = n - 1 + \frac{2}{n} \sum_{q=1}^{n} C(q-1)$$

Multiplicamos ambos lados por *n*, le restamos la misma fórmula para *n*–1, y reagrupamos:

$$nC(n) = (n+1)C(n-1) + 2n - 2$$

Finalmente, dividimos esta ecuación por $n(n+1)$, descartamos el término de más a la derecha ($\approx 2/n^2$), y desarrollamos:

$$\frac{C(n)}{n+1} = \frac{C(n-1)}{n} + \frac{2}{n+1}$$

$$\cdots = \frac{C(n-2)}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

$$\cdots = \frac{C(n-3)}{n-2} + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

$$\vdots$$

$$\frac{C(n)}{n+1} = \frac{C(1)}{2} + 2\sum_{i=3}^{n+1} \frac{1}{i} \approx 2\log(n)$$

Caso Promedio

El número de comparaciones $C(n)$ promedio es aproximadamente:

$$(n+1) \cdot 2 \log n$$

Por lo tanto la complejidad promedio de QuickSort es

$$O(n \cdot \log n)$$

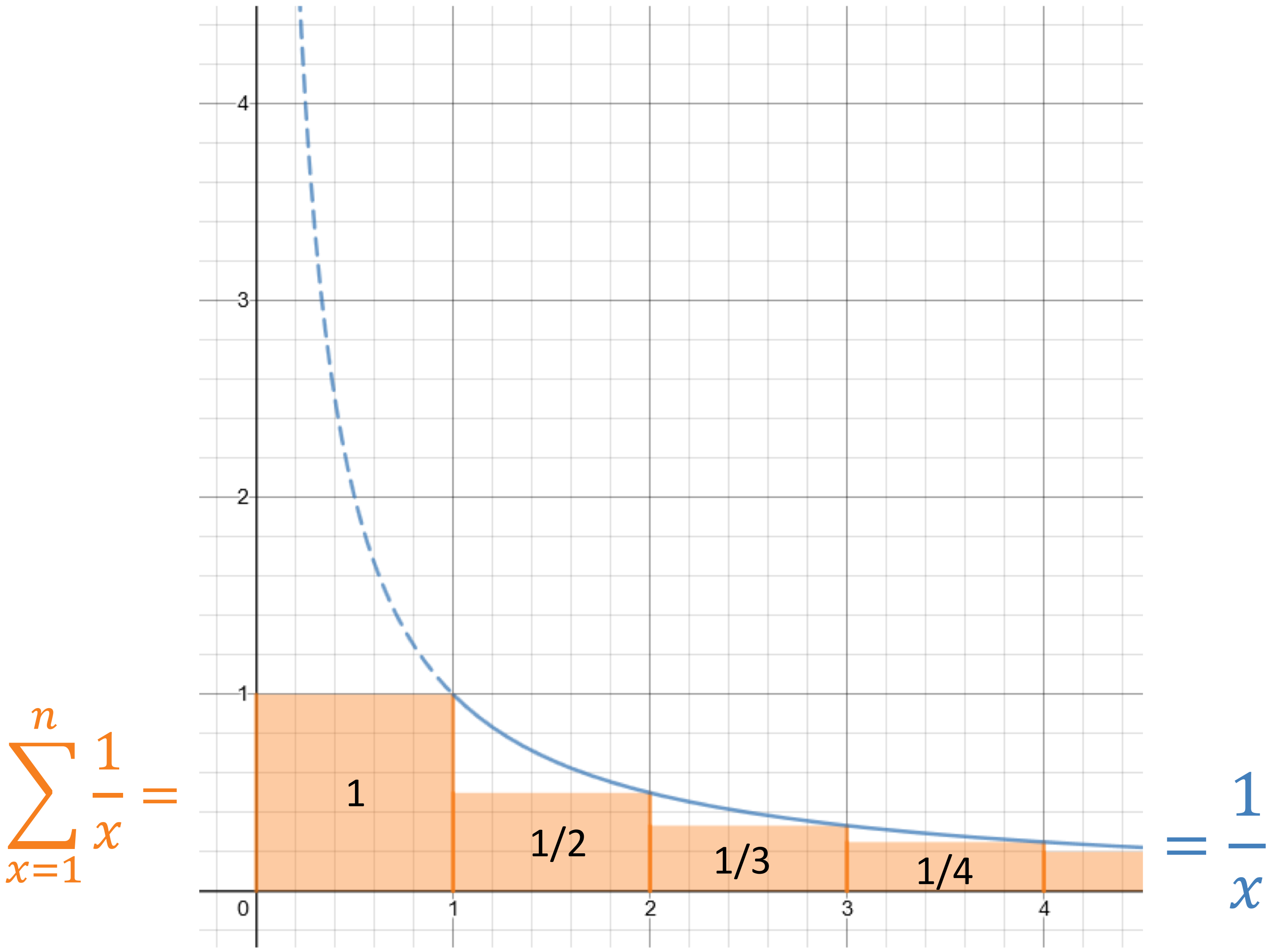
Posibles mejoras

Cambiar a `insertionSort()` para subarreglos pequeños ($n \leq 20$)

Usar la mediana de tres elementos como pivote

Si hay cantidades grandes de claves duplicadas en el arreglo de entrada —p.ej., un archivo de fechas— es muy probable que `quickSort()` particione, innecesariamente, subarreglos en que todas las claves son iguales:

- podemos particionar el arreglo en tres partes: ítemes con claves menores que el pivote, iguales al pivote, y mayores que el pivote



ANEXO: de sumatoria a log(n)

Acotando por arriba:

$$\sum_{i=2}^{n} \frac{1}{i} < \int_{1}^{n} \frac{1}{x} dx$$

$$\sum_{i=1}^{n} \frac{1}{i} < \int_{1}^{n} \frac{1}{x} dx + 1$$

$$\sum_{i=1}^{n} \frac{1}{i} < \ln n + 1$$