Asesoría 2: Divide-and-Conquer y Análisis Probabilístico

May 2021

1 Divide-and-Conquer

1.1 Introducción

Divide-and-Conquer es un paradigma (o manera) para solucionar problemas. Dependiendo del problema, uno elige un paradigma para poder solucionarlo. Hay distintos paradigmas como: algoritmos voraces, programación dinámica, backtracking, divide-and-conquer, etc. Antes de aplicar un paradigma, tenemos que analizar el problema para ver si **su estructura** va de acuerdo a la estrategia que el paradigma plantea.

1.2 Divide-and-Conquer

El paradigma de divide-and-conquer expresa lo siguiente:

1. Tenemos un problema original.
2. Lo **dividimos** en *subproblemas*, donde cada subproblema es **idéntico** al problema original, pero de tamaño menor.
3. Hacemos esto para cada subproblema, resolver (o **conquistar**) recursivamente cada subproblema generado, hasta que lleguemos a un tamaño de problema donde el problema se puede solucionar de manera directa (caso base).
4. Una vez que ya tenemos estas soluciones, algunos algoritmos necesitan que **combinemos** estas soluciones chiquitas, en soluciones más grandes. De tal manera que al combinar todas las *subsoluciones*, obtenemos la solución al problema original.

Dada la naturaleza recursiva de los problemas de Divide-and-Conquer, se usan las relaciones de recurrencia para generalizar la complejidad de dichos algoritmos:

$$T(n) = aT(n/b) + D(n) + C(n)tr$$

Donde $D(n)$ es el tiempo de **dividir**, $C(n)$ es el tiempo de **combinar** y $aT(n/b)$ es la parte que ya conocemos.

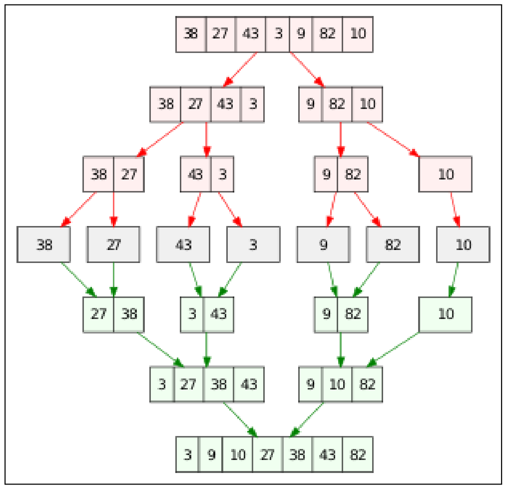


Figure 1: Merge Sort: la parte roja es la etapa de **dividir** y la parte verde es la etapa de **combinar.**

1.3 Template

Require: Un problema P
Ensure: La solución del problema P
Divide-and-Conquer(P)**:**
if P es lo suficientemente pequeño **then**
  return solución directa de P
end if

Dividir P en subproblemas P_1, P_2, $\ldots$, P_a
S_1 = **Divide-and-Conquer**(P_1)
S_2 = **Divide-and-Conquer**(P_2)
$S_{\ldots}$ = **Divide-and-Conquer**($\ldots$)
S_a = **Divide-and-Conquer**(P_a)

Combinar todos los subsoluciones S_1, S_2, $\ldots$, S_a

return solución combinada

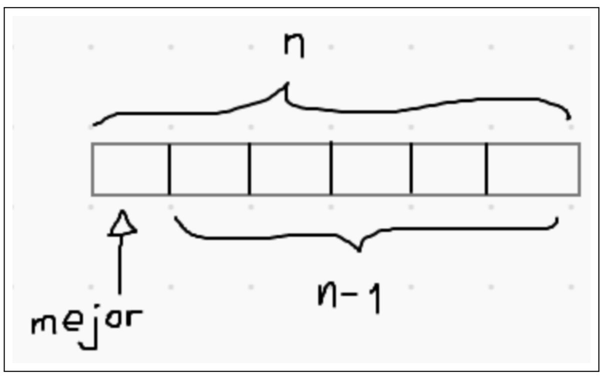
- Notar que hay algunos algoritmos que no van a necesitar que se guarden las subsoluciones, como el caso de Quicksort, donde no hay etapa de combinar.

2 Análisis Probabilístico

Ejercicio 10. En el pseudocódigo de HIRE-ASSISTANT, suponiendo que los candidatos se presentan en de manera aleatoria uniforme, ¿cual es la probabilidad que se contrate exactamente una vez?, ¿cual es la probabilidad de que se contrate n veces?

Para facilitar la resolución de este ejercicio, veamos a los candidatos como un arreglo de números, donde cada número representa qué tan bueno es el candidato. Además, el arreglo tiene n números por los n candidatos.

¿Cuál es la probabilidad de que se contrate una sola vez? Esto significa que el mejor candidato, se presenta primero. Es decir, que el máximo elemento en el arreglo, esté en la primera posición.



Claramente, $n!$ es nuestro mar de posibilidades, ya que representa todos los arreglos posibles que se pueden formar con los n candidatos. De ese mar de posibilidades, tenemos $(n-1)!$ escenarios posibles donde el mejor candidato se presenta primero, es decir, el primer elemento del arreglo es el mayor. Por lo tanto, la probabilidad de que se contrate exactamente una vez es:

$$\frac{(n-1)!}{n!}$$

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

¿Cuál es la probabilidad de contratar a todos los candidatos? Esto significa que cada candidato que se presenta es mejor que el anterior. Es decir, que el arreglo esté ordenado de manera ascendente.

Nuevamente, $n!$ es nuestro mar de posibilidades y dentro de ese mar de posibilidades, solo un arreglo está ordenado de manera ascendente. Por lo tanto, la probabilidad de que contratemos n veces es:

$$\frac{1}{n!}$$

Ejercicio 9. Sea X una variable aleatória que guarda el número de caras en dos lanzamientos de uma moneda justa. ¿Cuánto vale $E[X^2]$? ¿Cuánto vale $E[X]^2$?

Primero, recordar que:

$$E[X] = \sum_{x} x.Pr(X = x)$$

Dado este experimento, nuestra variable aleatoria X puede tomar los valores $x = 0$, $x = 1$ o $x = 2$, ya que lanzando dos veces una moneda, nos pueden tomar 0 caras, 1 cara o 2 caras. Por lo tanto:

$$X = \{0, 1, 2\}$$

Entonces,

$$E[X]^2 = [\ \sum_{x=0}^{2} x.Pr(X = x)\ ]^2$$

$$= [\ 2.\frac{1}{4} + 1.\frac{1}{2} + 0.\frac{1}{4}\ ]^2$$

$$E[X]^2 = 1$$

$2.\frac{1}{4}$ porque la probabilidad de que toquen dos caras seguidas es la multiplicación de la probabilidad de que toque una cara (Pr=$\frac{1}{2}$) **y** que luego toque otra (Pr=$\frac{1}{2}$). La multiplicación se hace por la regla de la multiplicación: cuando son eventos independientes y queremos que ambos ocurran. Un caso análogo ocurre con $1.\frac{1}{2}$ y con $0.\frac{1}{4}$.

Ahora para el otro caso:

$$E[X^2] = [\ \sum_{x=0}^{2} x^2.Pr(X = x)\ ]$$

$$= 2^2.\frac{1}{4} + 1^2.\frac{1}{2} + 0^2.\frac{1}{4}$$

$$E[X^2] = \frac{3}{2}$$

Ejercicio 4 (4 ptos). Considere el siguiente algoritmo que determina el segundo mayor elemento de un vector $v[1 \ldots n]$ com $n \geq 2$ números positivos distintos.

```
Algo(v, n)
 1: mayor = 0
 2: segundo_mayor = 0
 3: for i = 1 to n
 4:    if v[i] > mayor
 5:       segundo_mayor = mayor
 6:       mayor = v[i]
 7:    else
 8:       if v[i] > segundo_mayor
 9:          segundo_mayor = v[i]
10: return  segundo_mayor
```

Suponga que v es una permutación de 1 a n escogida de entre todas las permutaciones de 1 a n con distribución uniforme de probabilidad. Sea X la variable aleatória que guarda el número de veces que la variable *segundo_mayor* es alterada (osea, el número de ejecuciones de las líneas 5 y 9 del algoritmo) en una llamada a Algo(v, n). Calcule el valor esperado de X.

$$X_i = \begin{cases} 1 : \text{la línea 5 o 9 es ejecutada en la iteración i} \\ \\ 0 : \text{caso contrario} \end{cases}$$

Como X_i es una variable aleatoria indicadora, su esperanza sería:

$$E[X_i] = Pr(X_i = 1)$$

$$= Pr(\text{"la línea 5 es ejecutada en la iteración i"}) + Pr(\text{"la línea 9 es ejecutada en la iteración i"})$$

$$= Pr(\text{"i es el mayor elemento en A[1...i]"}) + Pr(\text{"i es el segundo mayor elemento en A[1...i]"})$$

$$= \frac{1}{i} + \frac{1}{i}$$

$$E[X_i] = \frac{2}{i}$$

Ahora:

$$E[X] = \sum_{i=1}^{n} \frac{2}{i}$$

$$= 2\sum_{i=1}^{n}\frac{1}{i}$$

$$= 2H_n$$

$$= 2\ln n$$

Por lo tanto, el número esperado de veces en las que se alterará a la variable *segundo_mayor* es $2\ln n$.

Ejercicio 3. Considere el siguiente algoritmo que determina el el mayor y menor elemento de un vector $v[1 \dots n]$ con números positivos distintos.

```
MAYORMENOR(v, n)
1: mayor = v[1]
2: menor = v[1]
3: for i = 2 to n
4:   if v[i] > mayor
5:     mayor = v[i]
6:   else
7:     if v[i] < menor
8:       menor = v[i]
9: return mayor, menor
```

Suponga que la entrada del algoritmo es una permutación de 1 a n escogida uniformemente dentre todas las permutaciones de 1 a n. ¿Cual es el número esperado de comparaciones ejecutadas en la línea 7 del algoritmo? ¿Cual es el número esperado de atribuiciones efectuadas en la línea 8 del algoritmo?

Notar que la línea 7 es ejecutada siempre y cuando el IF de la línea 4 evalúa a FALSE. Es decir, cuando el elemento en la posición i del arreglo no es el máximo en $A[1 \dots i]$.

Notar que la línea 8 es ejecutada si el elemento el la posición i del arreglo, es el menor en $A[1 \dots i]$.

Sea X la variable aleatoria que cuenta el número de ejecuciones de la línea 7. Sea X_i la variable indicadora que cuenta si la línea 7 es ejecutada en la iteración i para $i = 2 \dots n$.

$$E[X_i] = Pr(X_i = 1)$$

$$Pr(X_i = 1) = Pr(\text{"se ejecuta la línea 7 en la iteración i"})$$

$$= Pr(\text{"A[i] no es el elemento máximo en A[1...i]"})$$

$$= 1 - Pr(\text{"A[i] es el elemento máximo en A[1...i]"})$$

$$= 1 - \frac{1}{i}$$

Entonces:

$$E[X] = \sum_{i=2}^{n} E[X_i]$$

$$= \sum_{i=2}^{n} 1 - \frac{1}{i}$$

$$= \sum_{i=1}^{n} 1 - \frac{1}{i}$$

$$= n - H_n$$

$$= n - \ln n$$

Sea Y la variable aleatoria que cuenta el número de veces que se ejecuta la línea 8. Sea Y_i la variable indicadora que cuenta si la línea 8 es ejecutada en la iteración i para $i = 2 \dots n$.

$$E[Y_i] = Pr(Y_i = 1)$$

$$Pr(Y_i = 1) = Pr(\text{"se ejecuta la línea 8 en la iteración i"})$$

$$= Pr(\text{"A[i] es mínimo en A[1\dots i]"})$$

$$= \frac{1}{i}$$

Entonces:

$$E[Y] = \sum_{i=2}^{n} E[Y_i]$$

$$= \sum_{i=2}^{n} \frac{1}{i}$$

$$= \sum_{i=1}^{n} \frac{1}{i} - 1$$

$$= H_n - 1$$

$$= \ln n - 1$$

3 Recursos Extras

- Capítulo 5 del Tardos, en especial la sección 5.3.
- Sección 2.3, Capítulo 4 y 5 del Cormen.
- Abdul Bari: `https://www.youtube.com/channel/UCZCFT11CWBi3MHNlGf019nw`.
- Back to Back SWE: `https://www.youtube.com/c/BackToBackSWE/playlists`