Universidad Nacional de Ingeniería
Ciencias de la Computación
Análisis Probabilistico y Algoritmos Aleatórios

Yuri Nuñez Medrano [*]
ynunezm@gmail.com

Resumen

Se analizará el Analisis Probabilístico y los Algoritmos Aleatórios.

1. Analisis probabilísticos

Puede ayudarnos a medir el tiempo de ejecución de algorítmos, donde su complejidad varia según los datos de entrada. Tambien se pueden variar los algoritmos para que cumplan ciertas condiciones.

Algorithm 1: HIRE_ASSISTANT(n)

```
best = 0
for i = 1 to n do
    interview candidate i
    if cadidate i is better than candidate best then
        best=i
        hire candidate i
```

1.1. Análisis

En el algoritmo 1 tenemos el problema de contratación, a diferencia de los temas anteriores, tenemos que tener en cuenta los costos concernientes a las entrevistas y las contrataciones.
n es el número de candidatos.
m es el numero de contratado.
La entrevista tiene un costo (bajo) c_i.
La contratación tiene un costo (alto) c_h.
El costo asociado con el algorítmo seria $O(nc_i + mc_h)$.

1.2. Analisis del caso peor

En el Peor caso, seria cuando la calidad de los entrevistados sea creciente, pues se tendria que contrata a todos, entonces el tiempo de ejecución sería $O(c_h n)$.

1.3. Analisis probabilístico

Caso Mejor $t(x)$ = tiempo que tarda el algorítmo para la entrada x.
$T(n) = min\{t(x)\}, |x| = n$
Caso Peor.
$T(n) = max\{t(x)\}, |x| = n$
Caso promedio.
Se evalua con la Espectativa (esperanza) sobre la distribución de las entradas, donde se promedia el tiempo de corrida para todas ellas.
$E(T(n)) = \sum_{x\ tal\ que\ |x|=n} t(x)Pr(x)$.
Se puede suponer que los postulantes se presentan en un orden aleatório, y se puede establecer un orden total entre ellos decir cual es el mejor.
El orden total permite calificar a cada postulante con un valor entre $1..n$, y el orden aleatório permite asegurar que aparecen n! posibles permutaciones, cada una con la misma probabilidad que cualquier otra. $Pr(x)$ es igual para toda la entrada x!.

$E(T(n)) = Pr(x) \sum_{x\ tal\ que\ |x|=n} t(x)$.

2. Agorítmos Aleatorios

2.1. Algorítmos Deterministicos

Cuando se tiene un algorítmo cuyo comportamiento esta completamente determinado por su entrada. Siempre se comporta igual para multiples ejecuciones con la misma entrada.

[*]Escuela de Ciencias de la Computación, 27-08-15

2.2. Algorítmos Aleatorizados

En caso de que no se conozca la distribución de los datos de entrada, se pueden usar la aleatorización como herramienta de diseño para asegurar que parte del algorítmo se comporte aleatoriamente.
Una algorítmo aleatorizado es si su comportamiento es determinado no solo por su entrada, sino por valores producidos por un generador de números aleatórios random(a,b), que producen un entero en el rando $[a..b]$ que tienen la misma probabilidad de aparecer de los otros enteros del rango. En el algorítmo 1, se puede imponer un orden aleatório en el orden de llegada de los postulantes, con lo que se gana control en el proceso y se garantiza un ordenamiento aleatório.

2.3. Algorítmos aleatorias indicadoras

Son variable binarias que se usan para efectuar conversiones entre probabilidades y la espectativa (esperanza).
Suponga un espacio muestral S y un evento A. Se define la variable aleatória indicadora $I\{A\}$ asociada con un evento A así.

$$I\{A\} = \left\{ \begin{array}{cc} 1 & si\ A\ ocurre \\ 0 & si\ A\ no\ ocurre \end{array} \right\} \tag{1}$$

Lema: Dado un espacio de muestra S y un evento A en el espacio de muestra, y dado $X_A = I\{A\}$ entonces:
$E[X_A] = Pr\{A\}$
Prueba:
$E[X_A] = E[I\{A\}]$
$E[X_A] = 1.Pr\{A\} + 0.Pr\{\bar{A}\}$
$E[X_A] = Pr\{A\}$
Donde $\bar{A}$ es $S - A$, el complemento.

Ejm: Lanzar una mondeda al aire.
Se tiene un espacio $S = Cara, Sello$. Se tiene una variable aleatória. Y que se pueden tomar los valores $Cara$ o $Sello$, cada uno con una probabilidad de 1/2. Se define una variable aleatória X_{Cara} asociada con el evento de que la variable aleatória Y sea $Cara$, lo que se denota como:

$$X_{Cara} = I\{Y = Cara\} = \left\{ \begin{array}{cc} 1 & si\ Y = Cara \\ 0 & si\ Y = Sello \end{array} \right\} \tag{2}$$

La espectativa (valor esperado) para el evento en que la moneda cae en $Cara$ es la espectativa de la variable X_{Cara}.
$E[X_{Cara}] = E[I\{Y = Cara\}]$
$E[X_{Cara}] = 1.Pr\{Y = Cara\} + 0.Pr\{Y = Sello\}$
$E[X_{Cara}] = Pr\{Y = Cara\} = \frac{1}{2}$

Ejm. Lazar una moneda varias veces
Se tiene un espacio $S = Cara, Sello$. Se tiene una variable aleatória. Y_i que se pueden tomar los valores $Cara$ o $Sello$, lanzada número i.

Se define una variable aleatória indicadora X_i asociada con el evento de que la variable aleatória Y sea $Cara$, en la lanzada numero i, lo que se denota como.

$$X_i = I\{Y = Cara\} = \left\{ \begin{array}{cc} 1 & si\ Y_i\ Cara \\ 0 & si\ Y_i\ Sello \end{array} \right\} \tag{3}$$

Sea X la variable aleatória para el número de caras que se obtienen en n lanzamiento $X = \sum_{i=1}^{n} X_i$
La espectativa (valor esperado) para el número de $Caras$ es:
$E[X] = E[\sum_{i=1}^{n} X_i]$
$= \sum_{i=1}^{n} E[X_i]$
$= \sum_{i=1}^{n} \frac{1}{2}$
$= \frac{n}{2}$

Analisis del problema para el algoritmo 1.
Sea X la variable aleatória cuyo valor es igual al número de veces que se contrata a un nuevo postulante, se tiene que.
$E[X] = \sum_{x=1}^{n} xPr\{X = x\}$
Tambien que,

$$X_i = I\{candi.\ i\ es\ contrat.\} = \left\{ \begin{array}{cc} 1 & si\ el\ candi.\ i\ es\ contrat. \\ 0 & si\ el\ candi.\ i\ no\ es\ contrat. \end{array} \right\} \tag{4}$$

Ademas
$X = X_1 + X_2 + ... + X_n$
Entonces
$E[X_i] = Pr\{candidato\ i\ es\ contratado\}$

Por la linea 4 del algoritmo 1, el postulante i es contratado si es mejor que los candidatos $1..i-1$, por lo que tiene una probabilidad de $E[X_i] = \frac{1}{i}$, entonces.
$E[X] = E[\sum_{i=1}^{n} X_i]$
$= \sum_{i=1}^{n} E[X_i]$
$= \sum_{i=1}^{n} \frac{1}{i}$
$= lnn + O(1)$
Si se supone que los candidatos se presentan en un orden aleatorio, el algoritmo 1 tiene un costo de contratación de $O(c_h lnn)$.

En lugar de suponer una distribución en los datos de entrada, se aplica un algorítmo para aletorizarlos. Esto garantiza que cada permutación tenga la misma probabilidad que las otras.

Referencias

[H.Cormen et al., 2009] H.Cormen, T., Leiserson, C., and Riverson, R. L. (2009). *Algorithms*. The MIT Press.