S10

Materia	Diseño de algoritmos
Fecha	@August 28, 2023

Ejemplo $T_p(n) = ?, T_m(n) = ?, T_{1/2}(n) = ?$

```
j=0;                         '1OE'
while((a[j]<c)&&(j<n)){      '4OE'
  j=j+1;                     '2OE'
}
```

`a` → arreglo de n

`c` → constante, `n` → constante

`j` → índice

Calcular el peor caso

$T_p(n)$ → N operaciones Elementales

$T_p(n) = 1 + 4 + n(2 + 4) = 6n + 5 \in O(n)$

> 🍇 Ejemplo con $n = 3$
> 1 + 4 + 2 + 4 + 2 + 4 + 2 + 4
> j=0, j=1, j=2, j=3

Calcular el mejor caso

$T_m(n) = 1 + 4$ → Se queda en el calculo de la comparación `while` (se realiza por completo)

$T_m(n) = 5 \in O(1)$

Calcular el caso medio

$T_{1/2}(n) = ?$

La probabilidad de que el algoritmo se realice cierto numero de veces es equiprobable, por lo tanto la probabilidad de que ocurra cualquier evento es 1 sobre el numero de eventos que hay, en este ejemplo es $\frac{1}{n+1}$

nos interesa saber el número promedio de ciclos que podría dar nuestro algoritmo (esperanza)

$\sum_{j=0}^{n} jP$; Con P → probabilidad $= P\sum_{j=0}^{n} j$

En general tendríamos algo así: $E = \sum_{j=0}^{n} jP_j$ en caso de que las probabilidades sean distintas

Para nuestro ejemplo la esperanza es $P\sum_{j=0}^{n} j = \frac{1}{\cancel{n+1}}\frac{n\cancel{(n+1)}}{2} = \frac{n}{2}$

$T_{1/2}(n) = 1 + 4 + \frac{n}{2}(2+4) = 5 + 3n$

Por lo tanto el caso medio $\in O(n)$

Procedimientos/funciones

Las llamadas/retornos equivales a 1Operacion Elemental solo si no hay argumentos

Se pueden pasar argumentos por valor y por referencia

- Valor = 1OE
- Referencia = 1OE

`R = suma(8, 10)` → `1OE 2OE(1OE, 1OE)` → llamada a función y retorno son 1OE c/u → $5OE + T_{suma}$

Tiempo de una funcion/procedimiento =

$1OE_{llamada} + 1OE_{retorno} + \sum Arg + T_{Función}$

- Switch/case

  switch( ) → calcular el n de OE que hay dentro del paréntesis

  Vamos a suponer que en este cas solo es una OE incluyendo el salto

```
switch(x){      '1OE'
  caso a:
    ...         'T_a'
    break;
  caso b:
    ...         'T_b'
    break;
  default:
    ...         'T_default'
    break;
}
```

Ejemplo

```
switch(x){
  case 0:
    'O(n)'
    break;
  case 1:
    'O(logn)'
    break;
  case 2:
    'O(1)'
    break;
}
```

El peor caso tendría que ser que recorriera toda la estructura y que el default sea también el peor caso.

Por lo que debemos de calcular el T(n) de cada caso

$T_{select}(n) =$ nuestra asignación + n de casos + n de saltos + máximo de los t de ejecución

$T_{select}(n) = 1OE + 2\sum casos + max(T_1, T_2, ..., T_{n.caso})$

(el default no se cuenta)

El mejor caso sucede si el mejor tiempo esta en el primer caso

$T_{select}(n) = 1OE + 2OE + min(T_1, T_2, ..., T_{n.caso})$

El tiempo promedio

$$T_{select}(n) = 1OE + 2OE + \frac{min(T_1, T_2, ..., T_{n.caso})}{n.\ de\ \ casos}$$

Análisis de complejidad de algoritmos recursivos

El tiempo de ejecución del algoritmo depende del mismo algoritmo

$T(n) = E(T)$ Recursivo

$T(n) = T(n-1) + T(n-2)$

Un algoritmo recursivo también esta definido por sus condiciones iniciales

$T(1) = T_{01}$
$T(2) = T_{02}$ → constantes

Ejemplo en el caso de que $T(1) = 0$ y $T(2) = 1$

$T(3) = T(2) + T(1) = 1$

$T(4) = T(3) + T(2) = 2$

….

La idea es obtener una función de tiempo de ejecución que no dependa de T, in embargo no siempre es posible, queremos que esta función dependa de n → $E(n)$

```
Recursivo(x){
  if(x==0) return 1;
  Recursivo (x-1);
}
```

🍇 $T(n) = T(n-1) + T(n-2)$ → ecuaciones en diferencias

Normalmente se resuelven haciendo un cambio de variable

$T(3) = T(2) + T(1) = 1$

$T(4) = T(3) + T(2) = 2$

….

$T(n) = x^n$ Exponencial → propuesta de cambio de variable