TECNOLÓGICO DE COSTA RICA **II SEMESTRE 2022**
MAESTRÍA EN COMPUTACIÓN
MC-TAREA 6102 ANÁLISIS Y DISEÑO DE ALGORITMOS

TAREA 1

1. Probar por inducción para n >= 0 que

$$\sum_{i=1}^{n} i^2 2^i = n^2 \cdot 2^{n+1} - n \cdot 2^{n+2} + 3 \cdot 2^{n+1} - 6$$

Cierto para n=0:

$$\sum_{i=1}^{0} i^2 2^i = 0 = 0^2 \cdot 2^{0+1} - 0 \cdot 2^{0+2} + 3 \cdot 2^{0+1} - 6 = 0 - 0 + 6 - 6$$

Hipótesis de inducción: suponer cierto para n.

Mostrar que es cierto para n+1:

$$\sum_{i=1}^{n+1} i^2 2^i = (n+1)^2 \cdot 2^{(n+1)+1} - (n+1) \cdot 2^{(n+1)+2} + 3 \cdot 2^{(n+1)+1} - 6$$

Lado izquierdo:

$$\sum_{i=1}^{n+1} i^2 2^i = (n+1)^2 \cdot 2^{n+1} + \sum_{i=1}^{n} i^2 2^i$$

$$\sum_{i=1}^{n+1} i^2 2^i = (n^2 + 2n + 1) 2^{n+1} + n^2 \cdot 2^{n+1} - n \cdot 2^{n+2} + 3 \cdot 2^{n+1} - 6$$

$$\sum_{i=1}^{n+1} i^2 2^i = n^2 \cdot 2^{n+1} + 2n \cdot 2^{n+1} + 1 \cdot 2^{n+1} + n^2 \cdot 2^{n+1} - n \cdot 2^{n+2} + 3 \cdot 2^{n+1} - 6$$

$$\sum_{i=1}^{n+1} i^2 2^i = n^2 \cdot 2^{n+1} + n^2 \cdot 2^{n+1} + n \cdot 2^{n+2} - n \cdot 2^{n+2} + 1 \cdot 2^{n+1} + 3 \cdot 2^{n+1} - 6$$

$$\sum_{i=1}^{n+1} i^2 2^i = n^2 \cdot 2^{n+2} + 2^{n+3} - 6$$

Lado derecho,

$$(n+1)^2 \cdot 2^{(n+1)+1} - (n+1) \cdot 2^{(n+1)+2} + 3 \cdot 2^{(n+1)+1} - 6 = (n+1)^2 \cdot 2^{n+2} - (n+1) \cdot 2^{n+3} + 3 \cdot 2^{n+2} - 6$$

$$(n^2 + 2n + 1) \cdot 2^{n+2} - (n+1) \cdot 2^{n+3} + 3 \cdot 2^{n+2} - 6$$

$$n^2 \cdot 2^{n+2} + 2n \cdot 2^{n+2} + 1 \cdot 2^{n+2} - n \cdot 2^{n+3} - 1 \cdot 2^{n+3} + 3 \cdot 2^{n+2} - 6$$

$$n^2 \cdot 2^{n+2} + n \cdot 2^{n+3} + 2^{n+2} - n \cdot 2^{n+3} - 2^{n+3} + 3 \cdot 2^{n+2} - 6$$

$$n^2 \cdot 2^{n+2} + 2^{n+2} - 2 \cdot 2^{n+2} + 3 \cdot 2^{n+2} - 6$$

$$n^2 \cdot 2^{n+2} + 2 \cdot 2^{n+2} - 6$$

$$n^2 \cdot 2^{n+2} + 2^{n+3} - 6$$

2. Suponga que se tienen n palomas distribuidas en m palomares y que n > m. Mostrar ***por contradicción*** que al menos habrá un palomar con al menos $\lceil n/m \rceil$ palomas.

Suponer que no hay palomar con al menos $\lceil n/m \rceil$ palomas.
Esto es, todos los palomares tienen menos de $\lceil n/m \rceil$ palomas

Si n es múltiplo de m, se tiene n = k*m, k>1.
Lo que causa que: $\lceil n/m \rceil = \lceil (k*m)/m \rceil = \lceil k \rceil = k$.
Sea p_i el # de palomas de cualquier palomar.
$p_i < k$, i=1..n.
Sumar todos los m palomares:
$n = \Sigma\, p_i < k*m = n$
n < n ¡Contradicción!

Si n no es múltiplo de m, se tiene n = k*m + r, k>0, 0 < r < m.
Lo que causa que: $\lceil n/m \rceil = \lceil (k*m+r)/m \rceil = \lceil k + (r/m) \rceil = k+1$, ya que 0 < r/m < 1.
Sea p_i el # de palomas de cualquier palomar.
$p_i < k+1 \rightarrow p_i \leq k$, i=1..n.
Sumar todos los m palomares:
$n = \Sigma\, p_i \leq k*m$
$k*m + r \leq k * m$
$r \leq 0$ ¡Contradicción!
Por lo tanto hay por lo menos un palomar con al menos $\lceil n/m \rceil$ palomas.

3. Cada domingo Víctor le prepara el almuerzo a Carolina. El almuerzo consiste de un emparedado, una fruta y un postre. El emparedado puede ser de jamón y queso, mantequilla de maní y jalea, ensalada de huevo, o atún. La fruta es una manzana, una naranja o un poco de coco. El postre puede ser pretzel, brownie o pie de manzana.

a) ¿Cuántos almuerzos distintos le puede hacer Víctor a Carolina?
Emparedado x fruta x postre
{jamón y queso, mantequilla demaní y jalea, ensalada de huevo, atún }
x {manzana, naranaja, coco}
x {pretzel, brownie, pie de manzana}
4 * 3 * 3 = 36

b) Si Carolina no le gusta combinar ensalada de huevo con pie de manzana, ni atún con naranja, entonces, ¿cuántos almuerzos distintos aceptables le puede hacer Víctor a Carolina?
Almuerzos totales = 36
Almuerzos no aceptables =
{ ensalada de huevo } x { pie de manzana} x Postre o
{ atún } x { naranja } x Postre
1*1*3 + 1*1*3 = 6
Almuerzos aceptables = 36 -6 = 30.

4 Dar un algoritmo para resolver el siguiente problema. El algoritmo debe ser presentado por medio de una prosa precisa y completa. NO responda usando código. Analice el número de comparaciones requeridas en el mejor y en el peor caso.

Dado un arreglo de enteros positivos T[1..n] y un valor entero positivo x, modificar el arreglo para dejar solo los elementos originales que sean menores o iguales a x, e indicar el nuevo tamaño del arreglo.

Por ejemplo, para T = [3,2,7,4,5,2,8,1,3] y x=6, el resultado podría ser como T = [3,2,4,5,2,1,3], n=7.

No necesariamente los elementos escogidos deben quedar en el orden original del arreglo. Eso dependerá de su algoritmo. El punto fundamental del ejercicio es que la descripción del algoritmo sea de ***alto nivel*** y que deje en claro las condiciones involucradas.

Algoritmo

Iniciar con **pos**=1 y **tam**=n (tamaño de T[] e índice de su última entrada).

Ciclo:
Buscar la primera entrada (de izquierda a derecha) en T[**pos**..**tam**] cuyo valor sea mayor a x.
Si dicha entrada existe (suponer que **pos** ahora apunta a ella), entonces intercambiar T[**pos**] y T[**tam**], decrementar **tam** en 1, y repetir Ciclo.
Si dicha entrada no existe, entonces devolver como resultado T[1..**tam**] con tamaño **tam**.

Todas las entradas son siempre comparadas una única vez: si es <= x simplemente se sigue con la siguiente y nunca se vuelve a comparar. Si es > x se pone al final del arreglo, se recorta el tamaño del arreglo y no se vuelve a comparar. Además las entradas del final, son enviadas a inicio y ahí son comparadas.

Por lo tanto, el mejor y peor caso son iguales: $\Theta(n)$.

El algoritmo anterior se puede codificar literalmente como se muestra a la izquierda en la tabla siguiente, para luego simplificarlo tal como se muestra a la derecha.

```
pos = 1
tam = n

ciclo:
while (pos <= tam)
    if (T[pos] <= x)
        pos ++
    else
        break

if (pos > tam)
    return T[1..tam]

T[pos] ↔ T[tam]
tam--
goto ciclo
```

```
pos = 1
tam = n

while (pos <= tam)
    if (T[pos] <= x)
        pos ++
    else
        T[pos] ↔ T[tam]
        tam--

return T[1..tam]
```

5. Dos algoritmos toman ***n^2 días*** y ***2^n segundos*** respectivamente para resolver una instancia de tamaño n de un problema. ¿Cuál es el tamaño de la instancia más pequeña para la cual el primer algoritmo es más eficiente que el segundo algoritmo? Aproximadamente, ¿cuánto tiempo le toma al primer algoritmo resolver dicha instancia?

   Se puede tener una serie exponencialmente en el dato n hasta que se encuentre un caso en que el primer algoritmo es más eficiente. Una vez establecido esto se realiza una búsqueda binaria para localizar entre la última iteración con el segundo algoritmo más eficiente y la primera iteración con el primer algoritmo más eficiente, hasta encontrar el primer valor de n que tiene como más eficiente el primer algoritmo.

   En la siguiente tabla se muestra que una serie exponencial en n localiza el valor n=32 con el primer algoritmo más eficiente. Haciendo búsqueda binaria entre 16 y 32 se determina que n=26 es el valor buscada: para n=26 el primer algoritmo es más eficiente, para n=25 el segundo algoritmo aún es más eficiente.

		Algo-1	Algo-2	
n	n^2 días	segundos de n^2 días	2^n segs	Comentario (en blanco: algo-2 más eficiente)
1	1	86400	2	
2	4	345600	4	
4	16	1382400	16	
8	64	5529600	256	
16	256	22118400	65536	
32	1024	88473600	4294967296	Algo-1 más eficiente
24	576	49766400	16777216	
28	784	67737600	268435456	Algo-1 más eficiente
26	676	58406400	67108864	Algo-1 más eficiente
25	625	54000000	33554432	