Tarea 1.
Inducción.
Fecha de entrega: 27 de agosto

21 de agosto de 2022

1. Resumen

1.1. Inducción

Pasos para una prueba inductiva

1. Identificar la hipótesis de inducción $P(k)$.
2. Mostrar que el predicado es cierto para el caso base, el cual es generalmente $P(0)$ o $P(\{\})$; aunque puede ser un de valor diferente, o incluso puede ser más de un sólo caso base.
3. Suponer que $P(k)$ (esto es, la hipótesis de inducción) es cierto y mostrar que por lo tanto $P(k+1)$ también lo es. Otra alternativa (llamada *inducción fuerte*) es suponer como hipótesis de inducción que la propiedad P es cierta para todos los valores hasta e incluyendo k, y entonces probar que eso implica que P es cierta para $k+1$.

1.2. Árbol de recursión

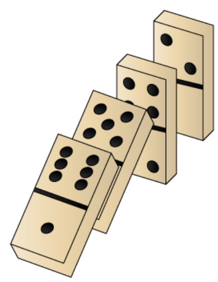
Una forma intuitiva de obtener la complejidad de un algoritmo recursivo A es obtener y analizar el árbol de recursión A con una entrada x.

El *árbol de recursión* de un algoritmo A con una entrada x es el árbol cuyos vértices son todas las instancias del algoritmo A que ocurren al ejecutar el algoritmo $A(x)$, la raíz del árbol corresponde a la instancia x y los hijos de un vértice son las instancias recursivas de la instancia correspondiente. Las hojas corresponden a invocaciones de casos base.

2. Ejemplos

- **Ejemplo 0:** Si tengo n piezas de domino acomodadas una tras otra y tiro la primera pieza entonces las n piezas también caerán.

  1. Hipótesis de inducción $P(k)$ la pieza k se cae.
  2. Caso base $P(1)$, Si tengo 1 sola pieza y la tiro entonces mi predicado es cierto.
  3. La hipótesis de inducción $P(k)$ es cierta, entonces la pieza k se cae, debemos probar que la pieza $k+1$ también se cae. $P(k+1)$ es cierto porque si la pieza k se cae va a tirar a la pieza $k+1$ por lo tanto las n piezas caen.



- **Ejemplo 1:** Probar la siguiente proposición

$$\sum_{i=0}^{k} i = \frac{k(k+1)}{2}.$$

La hipótesis de inducción $P(k)$ es :

$$P(k) : \sum_{i=0}^{k} i = \frac{k(k+1)}{2}$$

El caso base es cierto:

$$P(0) : \sum_{i=0}^{0} i = 0 = \frac{0 \times (0+1)}{2}$$

Paso Inductivo:
La hipótesis de inducción (IH) supone que para $k \in \mathbb{N}$, $P(k)$ es cierta. Ahora nosotros debemos probar que para $P(k+1)$ el predicado sigue siendo cierto.

$$P(k+1) : \sum_{i=0}^{k+1} i = \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}$$

Utilizando la hipótesis de inducción:

$$\sum_{i=0}^{k+1} i = \sum_{i=0}^{k} i + (k+1) = \frac{k(k+1)}{2} + (k+1)$$

$$\begin{aligned} &= (k+1) \times \left(\frac{k}{2} + 1\right) \\ &= (k+1) \times \left(\frac{k+2}{2}\right) \\ &= \frac{(k+1)(k+2)}{2} \end{aligned} \tag{1}$$

y por lo tanto $P(k)$ implica $P(k+1)$.

3. Tarea

1. Probar la siguiente proposición

$$\sum_{i=0}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}.$$

2. **Problema 1 (clase).** Dado un entero no negativo n, devolver el elemento s_n de la siguiente secuencia

$$\mathrm{s} = \underset{\mathrm{s}_0}{0}, \quad \underset{\mathrm{s}_1}{1}, \quad \ldots, \quad \mathrm{s}_n$$

y el resto de valores se definen de la siguiente forma:

$$s_{n+2} = 3s_{n+1} + 2s_n.$$

   a) Probar por inducción en n que el algoritmo $fpair()$ soluciona correctamente la variante del Problema 1, en donde dado un entero positivo n, el algoritmo devuelve la pareja s_n, s_{n-1}. Usando ese hecho, demostrar que el algoritmo $fpairsol()$ resuelve correctamente el Problema 1.

   b) Probar por inducción en n que el algoritmo $fexpmat()$ soluciona correctamente el Problema 1, es decir que dado un entero no negativo n, el algoritmo devuelve el elemento s_n. Además, a partir del árbol de recursión, obtén el *(*orden asintótico) del número de operaciones lógico/aritméticas que $fexpmat()$ realiza en la instancia n.

```
def fpair(n) :
    if n==1:
        return (1,0)                              ▷ (s_1, s,0)
    else:
        (P, PP) = fpair(n − 1)                    ▷ (s_{n−1}, s_{n−2}) = fpair(s_{n−1})
        return (3 ∗ P + 2 ∗ PP, P )               ▷ (3 ∗ s_{n−1} + 2 ∗ s_{n−2}, s_{n−1})
```

```
def fpairsol(n) :
    if n==0:
        return (0)
    else:
        (V, PP) = fpair(n)
        return V
```

```
def mat2mul(m_1, m_2) :
    (a, b, c, d) = m_1
    (p, q, r, s) = m_2
    return (a ∗ p + b ∗ r, a ∗ q + b ∗ s, c ∗ p + d ∗ r, c ∗ q + d ∗ s)
```

```
def matexp(m, n) :
    if n==0: return (1,0,0,1)                     ▷ return matriz identidad
    elif n%2 ==1:                                 ▷ Si n es impar
        return mat2mul(m, matexp(m, n − 1))
    else:                                         ▷ Si n es par
        powhalf = matexp(m, n//2)
        return mat2mul(powhalf, powhalf)
```

```
def fexpmat(m, n) :
    m = (3,2,1,0)
    return matexp(m, n)[2]                        ▷ (a,b,c,d)
```

3. **Problema 2.** Dado un entero no negativo n, devolver el elemento s_n de la siguiente secuencia

$$\mathrm{s} = \underset{\mathrm{s}_0}{0}, \quad \underset{\mathrm{s}_1}{1}, \quad \underset{\mathrm{s}_2}{2}, \quad \ldots, \quad \mathrm{s}_n$$

y el resto de valores se definen de la siguiente forma:

$$s_{n+3} = 5s_{n+2} + 4s_{n+1} - 2s_n.$$

a) Adaptando el algoritmo $fexpmat()$ plantea una solución al problema 2 y demuestra que la solución planteada resuelve el problema 2.

b) Obtén el órden asintótico del número de operaciones aritméticas que la solución planteada realiza en la instancia n.

4. Demuestra que el siguiente algoritmo ($maximo(A, n)$) devuelve el número más grande de un arreglo de números enteros A de longitud n, tal que $n > 0$. Además, obtén el orden asintótico del número de operaciones aritmético/lógicas que el algoritmo realiza para una instancia de longitud n.

```
def maximo(A, n) :
    max=A[0]
    for i de 1 a n − 1:
        if max < A[i]: max=A[i]
    return max
```