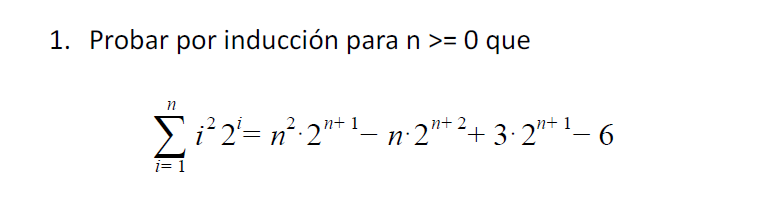
**Análisis de Algoritmos**

**Maestría en Ciencias de la Computación**

**Andrey Arguedas Espinoza - 2020426569**

**Tarea 1**



* Primer paso probamos el **caso base con n=0**

Se cumple el **caso base n=0**

* Continuamos con la hipótesis inductiva usando **“h”**
* Tesis inductiva **h+1**
* Demostración de la tesis con base a la hipótesis

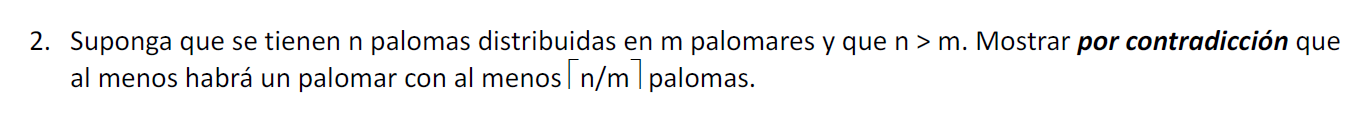
+ [

Aplicando la hipótesis de inducción

+

+

Por lo tanto queda demostrado.



Supongamos los siguiente:

* Las palomas son parte del conjunto **P**={p1,p2,p3… pn}, donde existen **n** palomas
* El conjunto de **m** palomares con las palomas adentro podemos decir que ∀**j = 1,...m palomares. Cm ⊆ P.** Quiere decir que en la suma de palomas contenidas en los palomares tiene que ser igual a la cantidad de palomas en **P.**
* Entonces definimos **|Cj| como el número de palomas en el palomar j.**
* La suma de todas las palomas en todos los palomares la podemos denotar con la siguiente sumatoria = n

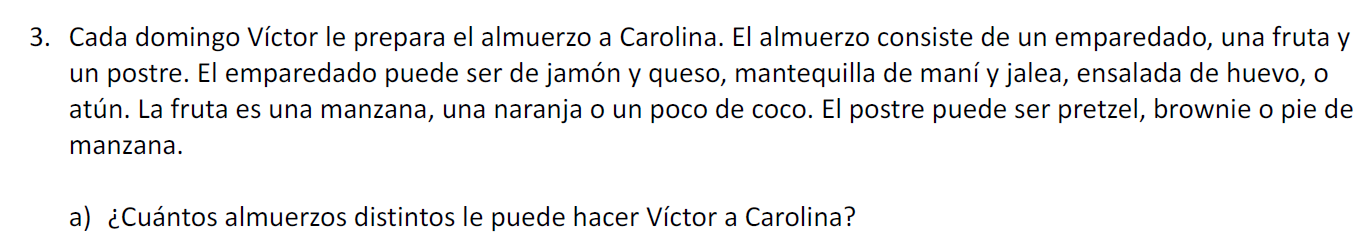
Con las definiciones anteriores queremos demostrar que Ǝ**j ∈ {1,..., m} | |Cj| ≥ n/m**

Para esto negamos la hipótesis anterior ∀**j ∈ {1,..., m}, |Cj| ≤ n/m**

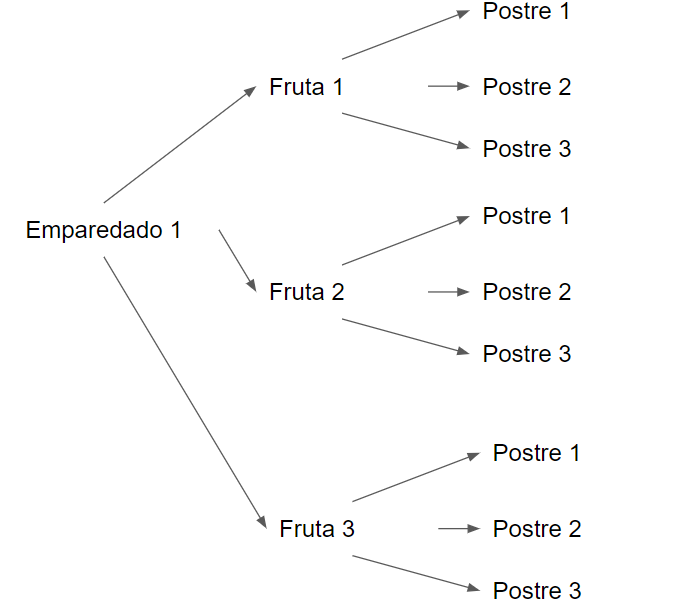
Podemos representar la cantidad de palomas en todos los palomares con la siguiente sumatoria:

La sumatoria de todas las palomas en palomares es igual a **n** palomas

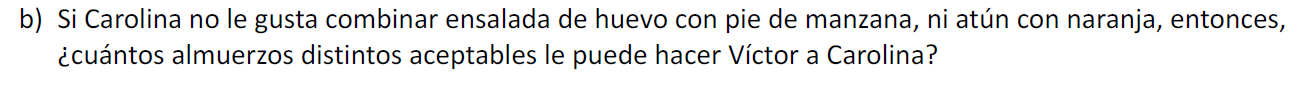
Esto es una contradicción siempre y cuando **m** sea mayor a 1 por lo que queda demostrado por contradicción.



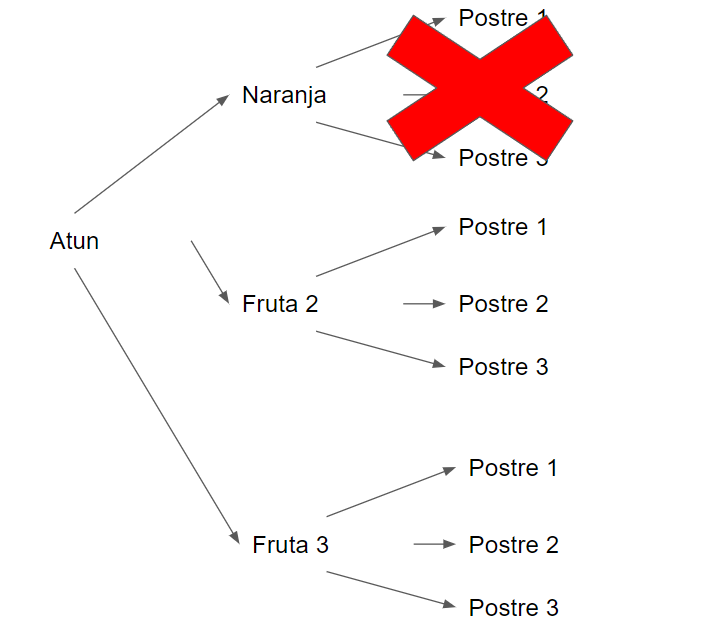
Supongamos el emparedado genérico llamado E1, para combinarlo podríamos hacerlo con las tres frutas y a su vez con los tres postres por lo que se generaría algo similar a esto:

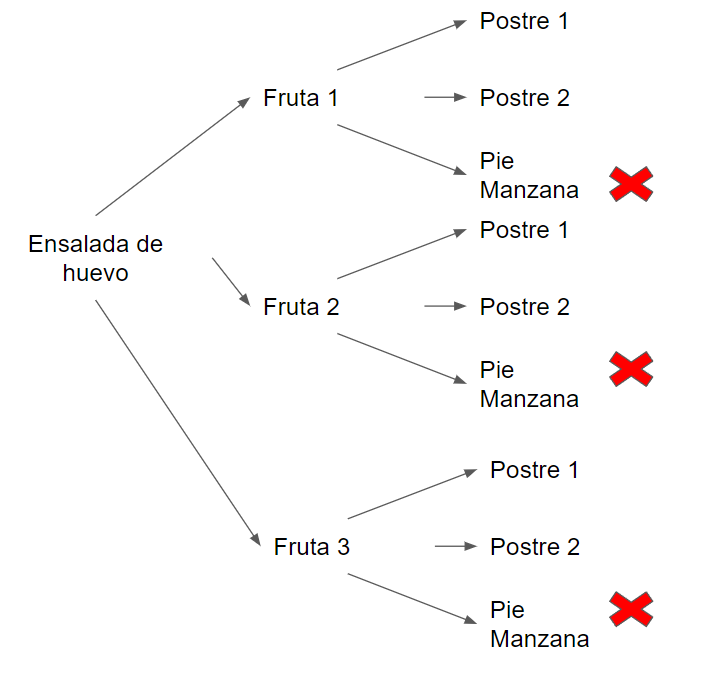


Si hacemos esto con los 4 emparedados nos da que **4\*3\*3 = 36.** Por lo que tenemos 36 combinaciones posibles de hacer almuerzo.

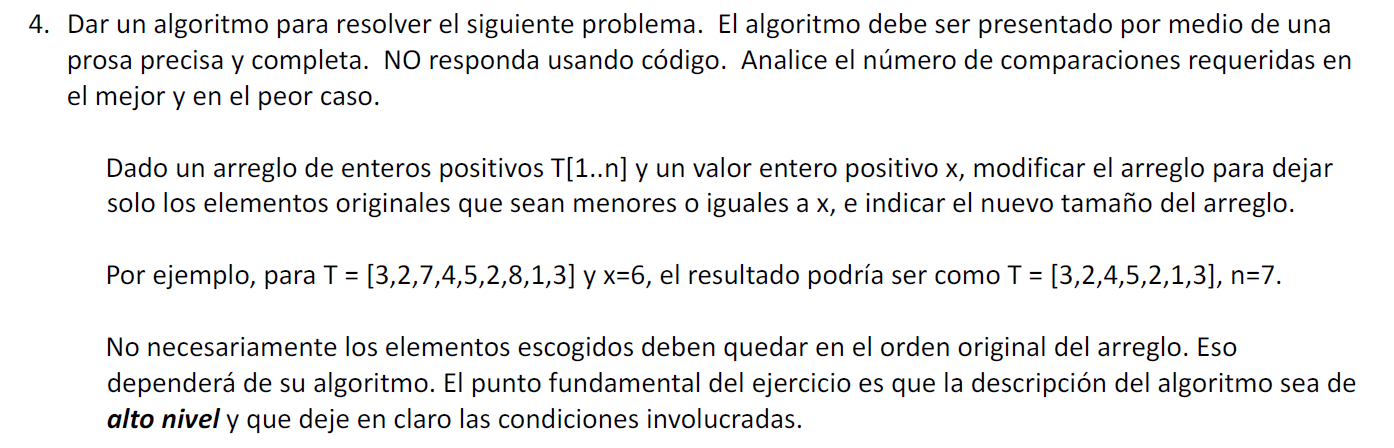


Supongamos que las combinaciones con la X roja son las no aceptables :





Como podemos ver **son 6 combinaciones en total no aceptables**, y quiere decir que en total Victor puede realizar **36-6 = 30** combinaciones aceptables.



Para resolver el siguiente problema nuestro algoritmo podría realizar los siguientes pasos siempre y cuando tengamos el arreglo **T[1…n]**

* Inicializamos los contadores **“i”**, **“c”, “ci”, “y”** en 0.
* Iteramos el arreglo elemento por elemento, con el índice **i** desde 1 hasta **n**
* Si el valor del arreglo en la posición **i** es mayor a **x** eliminamos el elemento en la posición **”i”** del arreglo e incrementamos **i + 1,** también incrementamos **c + 1**. De otro modo solamente incrementamos **i + 1.**
* Creamos un nuevo array de tamaño **c, F[1… c].**
* Iteramos de nuevo **T** desde **1** hasta **n** y todos los elementos distintos a un espacio en blanco los insertamos en el nuevo arreglo en la posición  **F[“ci+1”].**
* Finalmente el resultado es el arreglo **F.**

**Procedure algorithm(T[1…n], x)**

**i ← 0; c ← 0; ci ← 0; y ← 0**

**for i ← 1 to n-1 do:**

**If T[i] > x then delete T[i], i ← i+1, c ← c+1**

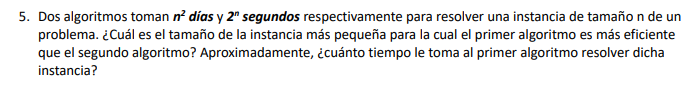
**Else i ← i+1**

**F = [1…c]**

**for i ← 1, ci ← 1 to n-1 do:**

**If T[i] == NULL | EMPTY then F[“ci”] = T[i], i ← i+1, ci ← ci+1**

**En el peor de los casos este algoritmo está en orden de O(2n).**

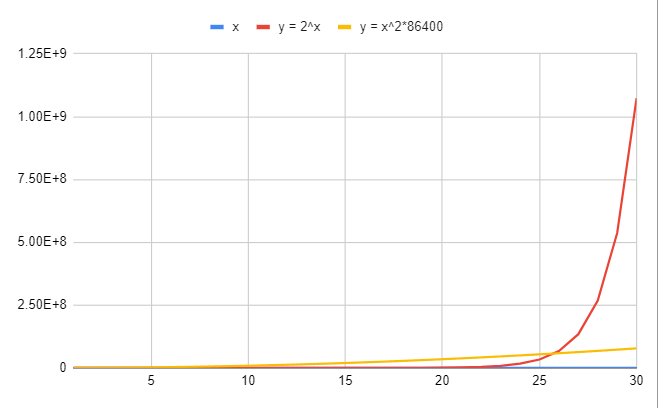


**A1 = n^2 días**

**A2 = 2^n segundos**

Convertimos A1 a notación de segundos eso quiere decir que **A1 = n^2 \* 86400.** Esto debido a que cada día cuenta con 86400 segundos.

Seguido graficamos las funciones :



Observamos que para **n=26**, se cumple que **A1** tardaría menos en resolverlo ya que tardaría **58406400** unidades de tiempo, mientras que **A2** tomaría **67108864.**

Finalmente podemos también saber que **A1** toma aproximadamente **676 días (58406400 / 86400)** en ser resuelto.