

---

# Fundamentos de Análisis y Diseño de Algoritmos

*Mini-Proyecto*

---

## **Integrantes:**

KAROL MARIANA MAYORQUIN HERRERA - 1860682 3743  
JENIFFER GUARIN ARISTIZABAL - 1860661 3743  
EDINSON ORLANDO DORADO DORADO - 1941966 3743  
SANTIAGO RAMIREZ OSPINA - 1841391 3743

## **Profesor:**

JESUS ALEXANDER ARANDA BUENO

FACULTAD DE INGENIERÍA  
ESCUELA DE INGENIERÍA DE SISTEMAS & COMPUTACIÓN

Marzo 22, 2022

## **1. Resumen y estructura del informe**

Lo primero que se presenta en el informe es el pseudocódigo que busca entre todas las posibilidades, una solución óptima, este algoritmo es recursivo, y usa fuerza bruta, o una técnica ciega. Este algoritmo será de utilidad al hacer el análisis de los 2 algoritmos siguientes. Un algoritmo que usa una técnica voraz, y que al compararlo con el primer algoritmo vamos a poder ver cuál es la relación de soluciones óptimas que encuentra el algoritmo voraz comparado con el algoritmo que usa fuerza bruta. El último algoritmo que se muestra y analiza usa la técnica de programación dinámica, que en teoría, (en teoría debido a que esto se va a probar en el análisis de dicho algoritmo), encuentra la mejor solución al problema, al igual que el primer algoritmo, pero de una forma mucho más eficiente.

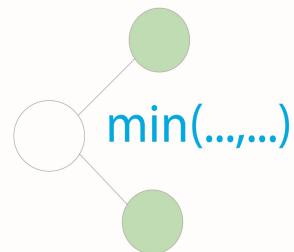
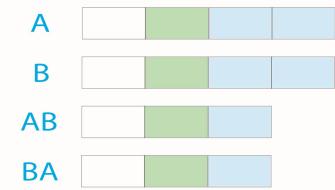
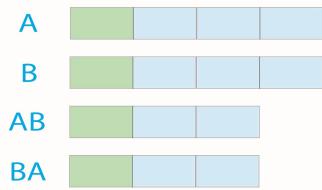
## 2. Solución usando fuerza bruta

Una forma segura de encontrar la solución óptima para algún problema, es procesar todas las soluciones. La forma que utiliza el algoritmo mostrado abajo es comparar en cada llamado recursivo los dos caminos posibles que esa instancia del problema tiene. Esto lo hace, por supuesto, llamándose a sí mismo con esas dos instancias, ambas, más pequeñas que el problema original. Esto se hace hasta que se llega a un estado trivial, que es cuando el problema queda de un solo elemento en el array *a* y *b* (así mismo el array *ab* y *ba* quedan vacíos).

Como para solucionar cada instancia del problema, se crean dos más, el número de operaciones que se deben hacer son  $2^n$  aproximadamente. Lo cual tiene una complejidad  $\mathbf{O}(2^n)$ .

Cabe recordar, que el objetivo con este primer algoritmo es solamente con fines de apoyar el análisis de los dos algoritmos siguientes.

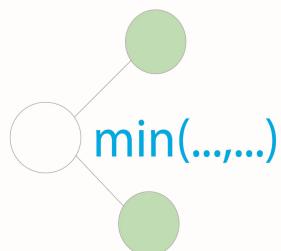
```
1: procedure ALGORITMOIEGO(n, a, b, ab, ba, counter, lineas, lineaActual)
2:   if n == 1 then
3:     if lineaActual == 'a' then
4:       return counter + b[len(b) - n], lineas + [b[len(b) - n]]
5:     else
6:       return counter + b[len(b) - n], lineas + [b[len(b) - n]]
7:     end if
8:   else
9:     if lineaActual == 'a' then
10:      return
11:        min(
12:          algoritmoCiego(n-1, a, b, ab, ba, counter + a[len(a) - n], lineas+[a[len(a) - n]], 'a'),
13:          algoritmoCiego(n-1, a, b, ab, ba, counter + ab[len(ab)-n+1] + a[len(a) - n], lineas + [a[len(a) - n]] + [10*ab[len(ab)-n+1]], 'b'))
14:      else
15:        return
16:          min(
17:            algoritmoCiego(n-1, a, b, ab, ba, counter + ba[len(ba)-n+1] + b[len(b) - n], lineas + [b[len(b) - n]] + [10*ba[len(ba)-n+1]], 'a'),
18:            algoritmoCiego(n-1, a, b, ab, ba, counter + b[len(b) - n], lineas + [b[len(b) - n]], 'b'))
19:      end if
20:    end if
21:  end procedure
```

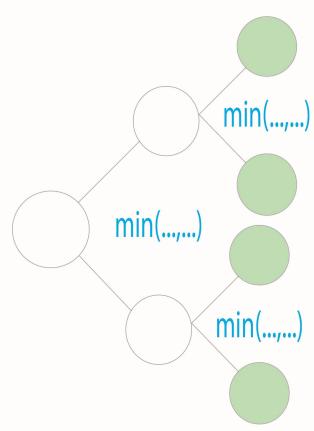
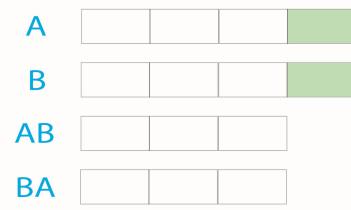
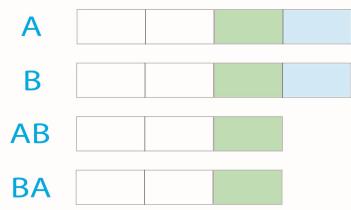


$\min(\dots,\dots)$

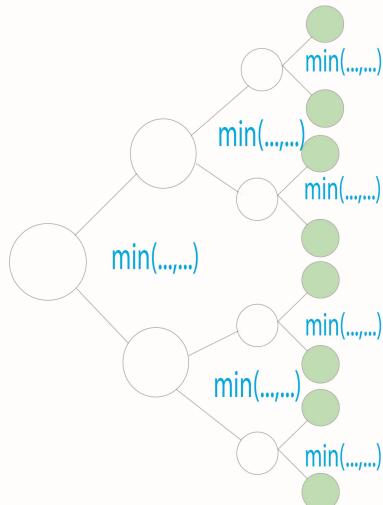


$\min(\dots,\dots)$

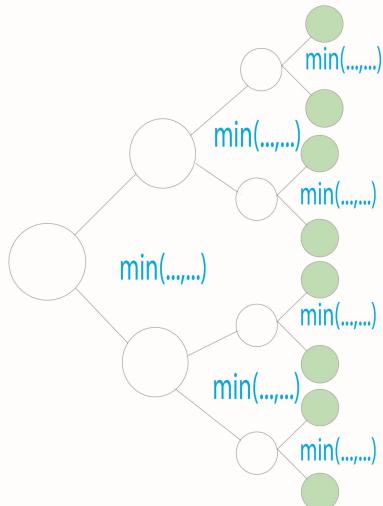
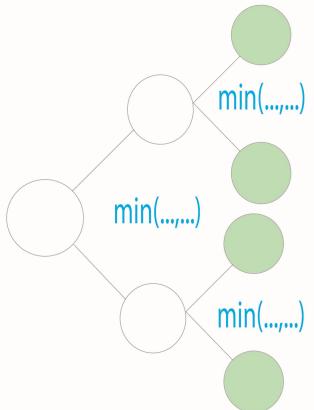


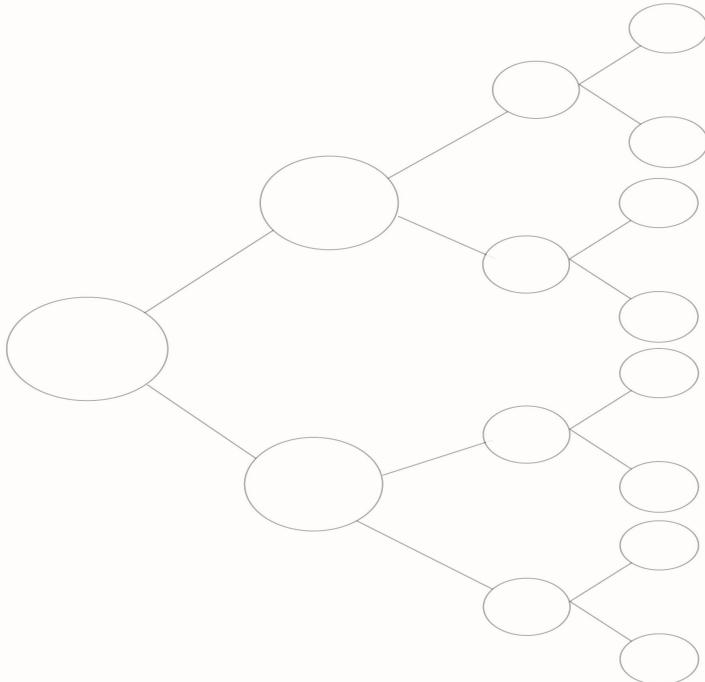
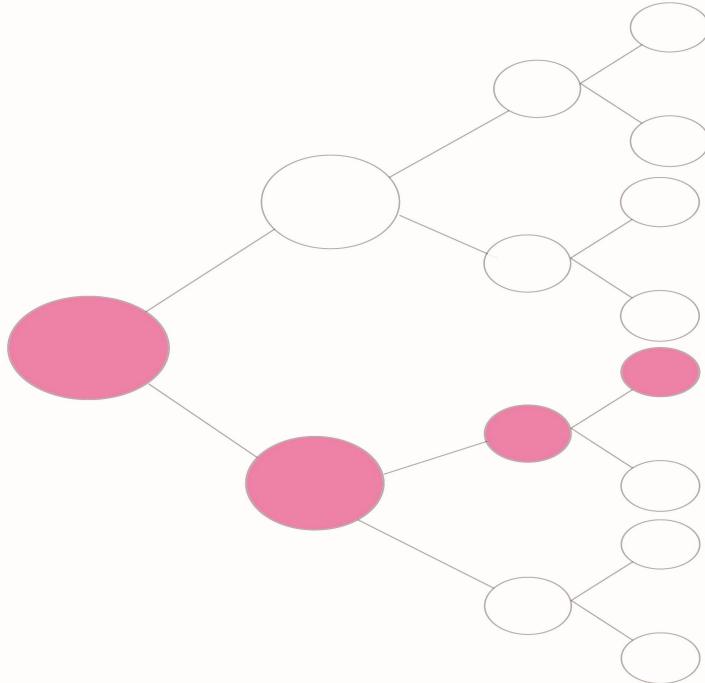
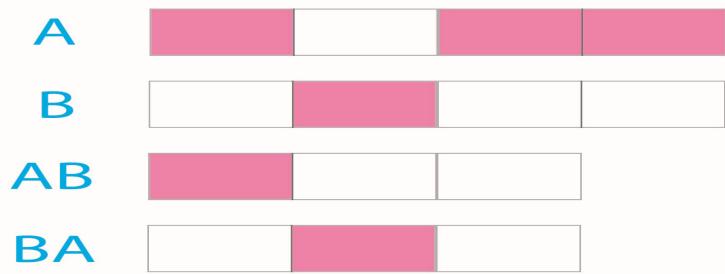


$\text{min}(\dots,\dots)$



$\text{min}(\dots,\dots)$





### 3. Solución usando Programación Dinámica

#### 3.1. Descripción de la idea

Se hará uso de las tablas hash para indicar cada punto de ensamblaje con su respectiva posición. La llave estará conformada por: el carril (línea a ó línea b) y la ubicación (posición del punto de ensamblaje). El contenido estará dado por: líneas (camino más óptimo que hay hasta un punto de ensamblaje) y counter (tiempo total del camino más óptimo que hay hasta un punto de ensamblaje).

Se crea una función que indique por cual linea (a ó b) se va a empezar la producción (actividad 1), es decir, determina la línea que tiene menor valor en la posición 0.

Se crea la función principal con los atributos: lineaInicial (indica en que linea se inicia la actividad 1), hash (tabla hash), a (linea a), b (linea b), ab (cambios de "a" a "b"), ba (cambios de "b" a "a"). Además se crea la variable estacion que indicará la posición del punto de ensamblaje que se está analizando, la cual se irá iterando dependiendo de los n puntos de ensamblaje. Se harán dos condiciones principales cuando (estación = 0) y cuando (estación > 0).

En la condición (estación = 0) se determina donde empieza la actividad 1; si (lineaInicial = a), se añade a la tabla hash "b" en la posición 0 con infinito ( $\infty$ ) y si (lineaInicial = b), se añade a la tabla hash "a" en la posición 0 con infinito ( $\infty$ ).

En la segunda condición (estación > 0), se crea la variable estacionDeAtras que será igual a estacion-1 y permitirá "mirar hacia atrás" comparando en cada posición el menor valor que hay entre a y b. Para escoger el camino y guardarlo en la tabla hash se tiene en cuenta el valor del punto de ensamblaje siguiente en a y b, como también el valor de cada cambio (ab o ba).

En un primer if se analiza los posibles caminos optimos que hay si se toma los puntos de ensamblaje que hay en "a", entonces ubicados en un punto de ensamblaje en a miramos hacia atrás y comparamos el punto que esta justo antes del que se está analizando, si es menor que la suma del punto en "b" de la estacionDeAtras y el cambio ba, si es así, en la tabla hash se almacena la linea "a" y el valor que tiene el punto de ensamblaje que está justo antes del punto que se está analizando en a.

En el segundo if de igual manera se analiza los posibles caminos optimos que hay si se toma los puntos de ensamblaje que hay en "b". Ubicados en un punto de ensamblaje en "b" miramos hacia atrás y comparamos si tal punto es menor que la suma del punto en "a" de la estacionDeAtras y el cambio ab, si es así, en la tabla hash se almacena la linea "b" y el valor que tiene el punto de ensamblaje que está justo antes del punto que se está analizando.