# CI-5651 Proyecto #3

Programación Dinámica

Alejandro Flores Guillermo Palma

## Introducción

El Proyecto #3 de *CI-5651* está inspirado en los conjuntos de problemas de un *maratón de programación*. En este sentido, el proyecto consta de 5 problemas independientes, basados en problemas de la ACM-ICPC[1]. Cada problema cuenta con un enunciado que describe en detalle el problema que se quiere resolver, así como el formato que sigue la entrada y la salida del programa que lo solucione. La *entrada* describe la instancia del problema que se quiere resolver, y la *salida* describe los resultados obtenidos para dicha instancia. Al igual que en los maratones de programación, para la evaluación de las soluciones al Proyecto #1 se usará la entrada y salida *estándar*.

## Problemas

A continuación se lista el conjunto de problemas que deben ser solucionados como parte del Proyecto #3 de *CI-5651*. La descripción de todos los problemas se encuentran en el *juez online* conocido como SPOJ[2]. Se recomienda probar sus soluciones en dicho juez antes de la entrega.

- **ACMAKER** | `http://www.spoj.com/problems/ACMAKER/`
  Debe implementarse siguiendo una estrategia de Programación Dinámica *top-down*. Es necesario el uso de una matriz de *memoización* de 2 dimensiones.

- **BABY** | `http://www.spoj.com/problems/BABY/`
  Debe implementarse siguiendo una estrategia de Programación Dinámica *top-down*. Es necesario el uso de una matriz de *memoización* de 2 dimensiones, apoyado en el uso de máscaras de bits.

[1] `http://icpc.baylor.edu/`
[2] `http://www.spoj.com/`

- **BORW** | `http://www.spoj.com/problems/BORW/`
  Debe implementarse siguiendo una estrategia de Programación Dinámica *top-down*. Es necesario el uso de una matriz de *memoización* de 3 dimensiones.
- **MAXWOODS** | `http://www.spoj.com/problems/MAXWOODS/`
  Debe implementarse siguiendo una estrategia de Programación Dinámica *bottom-up*. Es necesario el uso de una matriz de *memoización* de 1 dimensión. Aunque el problema parece sugerir que una matriz de 2 dimensiones es necesaria, las características del problema permiten la reducción del espacio necesario a una dimensión.

## Entrega

La fecha **máxima** de entrega es el 13 de marzo de 2015. Debe entregarse un archivo .zip con las implementaciones y el informe, enviado por correo electrónico a `alejandroflores.af@gmail.com`.

### Implementación

- El algoritmo debe implementarse en alguno de los siguientes lenguajes de programación: C, C++ o Java.
- Deben usarse **solo** librerías estándar.
- El código debe compilar (y correr) en `www.ideone.com`.
  - Deben usar **solo** librerías estándar.
  - La solución a cada problema debe estar en **un solo archivo** con el código fuente. Su nombre debe ser "`p`" seguido del número del problema. Por ejemplo: `p1.cpp`
  - La entrada debe leerse desde la entrada estándar y la salida debe imprimirse en la salida estándar. El programa debe poder ejecutarse desde la terminal, usando la siguiente llamada:
    ```
    > ./p1 <entrada.txt >salida.txt
    > java p1 <entrada.txt >salida.txt
    ```
- El código va a ser revisado y evaluado (no solo la correctitud).
  - Debe estar debidamente documentado.

    - Debe cumplir con buenas prácticas de implementación (guías de estilo de `C/C++` y `Java`).
    - La limitación a un solo archivo por problema no es excusa para la falta de modularidad y orden en el código.
- El formato de entrada y salida debe seguirse de forma estricta. Un espacio de más en la salida puede significar una evaluación incorrecta.

## Informe

Junto con el código, debe incluirse un informe (en formato PDF) donde describan sus soluciones a cada problema. Por cada problema, debe describirse el diseño del algoritmo que fue implementado para su solución; es necesario describir la función de optimización empleada, la estrategia de programación dinámica seguida, así como un análisis de su complejidad en tiempo y espacio.