Cómo redactar tus soluciones para el curso de TI2

Diego Caro

Agosto 2018

Aquí están las instrucciones de cómo escribir las soluciones de los ejercicios semanales, y que es lo que se espera que se entregue en ellas. Este documento es una adaptación de **How to write up homework solutions**[1] del curso CS 170 de la Universidad de California, Berkeley.

Cada vez que se le pregunte por cómo resolver algún problema programando en Python, su solución debe incluir los siguientes cuatro elementos:

1. La **presentación** del problema. Debe indicar en un par de líneas cuál es el problema que se debe resolver, cuáles son sus restricciones, y cuál es la información con la que se cuenta disponible. Si le parece conveniente, incluya una figura o una notación que ayude al lector a comprender lo que se desea resolver.

2. La **idea principal** de su solución. Debería ser corto y conciso, a lo más dos párrafos. No se deben indicar todos los detalles de la solución, del código, o explicar por qué está correcta su solución. Debe ser información suficiente para que un compañero de clase entienda la estrategia que utilizó para resolver el problema. Esta es la parte más importante de su informe. Si hace un buen trabajo aquí, el lector estará más receptivo y olvidará errores que se puedan presentar.

3. El **código** de su solución. El propósito del código es comunicar claramente cómo programó su solución. Debe ser presentado utilizando Python, y solo debe contener la función/método que contiene la solución al problema. No incluyas el código usado para leer los datos de entrada, tests unitarios o la función main.

4. La **validación** de su solución. En esta sección debes indicar por qué tu código resuelve el problema. Tus validaciones deben ser rigurosas y convincentes, considerando no sólo casos triviales, si no también casos límites. Lo más importante es demostrar que tu código resuelve correctamente el problema, sin importar como se escoja la entrada.

Pero... ¿Cuánto detalle es necesario? Aquí va una guía sencilla. Supone que tienes un set de pruebas unitarias (Una prueba unitaria es una forma de comprobar el correcto funcionamiento de un método, clase, o función, más info en https://es.wikipedia.org/wiki/Prueba_unitaria) que evalúa tu solución. Ahora pregúntate ti mismo si estas validaciones cubren todos los casos posibles. Si parece que no es obvia la respuesta, entonces debes volver a pensar los casos menos triviales y cuidadosamente revalidar su solución. Si le cuentas tus validaciones a alguien (por ejemplo, al ayudante o a alguien que ya haya pasado el curso) y la otra persona debe razonar por largo tiempo sobre si cubre todos los casos posibles, entonces quizás deberías reescribir la validación con mayor claridad para dichos casos especiales.

Importante: debes denotar **claramente** tus respuestas en cada una de estas categorías. En otras palabras, debes tener una sección "Idea principal", que en efecto indique cómo funciona tu solución. Deja un espacio en blanco por cada sección para distinguir fácilmente las 4 secciones de tu respuesta. Si el problema original se divide en sub-problemas, debe indicar para cada sub-problema cada una de estas secciones por separado.

Errores típicos al entregar soluciones

1. Indicar la idea principal de manera poco clara. Cuando se omite la idea principal, el lector debe inferir desde tu código cuál es la intención detrás, lo que les hace perder tiempo y volverse más gruñones (y además, se vuelve más quisquilloso con los errores o typos en el documento). Si incluyes la idea principal, estás haciendo más fácil la lectura y haciendo que los lectores digan "Wow, el/ella entendió el concepto", y luego es más probable obtener todo el puntaje en la

[1] Disponible en http://www-inst.eecs.berkeley.edu/~cs170/fa14/hws/instruct.pdf.

pregunta.

2. Hacer una validación poco informativa. Indicar de manera informal o imprecisa la validación de tu solución podría ser indicio de que el algoritmo no resuelve el problema, o de que no se sabe por qué funciona, o bien, funciona para sólo unos casos y no todos.
3. Añadir dos hojas llenas de código en Python, en vez de escribir solo la sección que interesa para entender el algoritmo. Recuerda que el código final debes entregarlo en un archivo aparte o subirlo a github.
4. Entregar el informe en Word. Para este curso todos los informes se deben entregar usando Markdown en formato PDF. Así podemos aprovechar el coloreo de sintáxis y un formato de dos columnas.