



Introducción al Pensamiento Computacional - Semana 6



© Todos los derechos reservados Universidad Rafael Landívar URL.

DESEMPEÑOS ESPERADOS

- ≡ Desempeños esperados

DESARROLLO DE CONOCIMIENTOS

- ≡ Evaluación de soluciones
- ≡ Recursos complementarios

APLICANDO LO APRENDIDO

- ≡ Actividad 1
- ≡ Actividad 2
- ≡ Actividad 3

Actividad 4

DE LA TEORÍA A LA PRÁCTICA Y REFLEXIÓN

Activación de conocimientos previos

CRITERIOS DE EVALUACIÓN

Rúbrica de evaluación

Diario de experiencias de laboratorio

FUENTES DE REFERENCIAS

Referencias

CRÉDITOS

Créditos

Desempeños esperados



El estudiante:

1

Discute las medidas clave de calidad de la efectividad de las soluciones, incluyendo correcciones con muestras empíricas; eficiencia, examinando complejidades de tiempo y espacio; elegancia y maximización de la efectividad, así como simplicidad y usabilidad.

2

Explica la importancia de las compensaciones de las medidas de calidad.

Evaluación de soluciones



¡Felicitaciones por llegar hasta este punto! Se ha analizado el problema, se descompuso en partes, se diseñó una solución, se implementó la solución, se probó y se debuggeó. Ya se tiene algo que funciona, así que hemos terminado, ¿verdad?

La verdad es que no.

El trabajo no ha terminado porque ya se tiene una solución. Antes de realmente finalizar, se debe analizar si la solución desarrollada es una buena solución. Se debe evaluar su calidad.

Hay muchos aspectos de calidad. Evaluar una solución involucra realizarse preguntas básicas, como:

- ¿Es correcta? ¿En realidad resuelve el problema que debía resolver?
- ¿Es eficiente? ¿Utiliza los recursos adecuadamente?
- ¿Es elegante? ¿Es simple y efectiva?
- ¿Es amigable con el usuario? ¿Provee una forma satisfactoria para que las personas la utilicen?

En este módulo se exploran las respuestas a estas preguntas.



Es la mejor pregunta que se debe realizar. En otras palabras, ¿se resolvió realmente el problema que se debía resolver? Si no, se debe iniciar nuevamente, independientemente de qué tan atractiva, rápida e interesante sea la solución. Si no resuelve el problema, no es aceptable.

El enfoque correcto es **asumir** que la solución es **incorrecta** hasta que se pruebe que es correcta. Para comprobarlo se pueden aplicar muchos métodos, incluso matemáticos, pero para fines prácticos, se pueden realizar **pruebas empíricas**. Para ello se realiza un plan de pruebas, como se vio en el módulo anterior. A continuación, un plan de pruebas para la máquina que opera con monedas.

Prueba No.	Aspecto a probar	Descripción de la prueba.	Salida esperada	Salida actual
1.	Depósito de monedas	Insertar una moneda de 50 centavos.	La moneda es aceptada. 0.5 aparece en la pantalla.	Se comportó como se esperaba.
2.	Depósito de monedas	Insertar una moneda extranjera.	La moneda es rechazada. Desplegar el mensaje "La moneda no es reconocida".	Falló. La moneda fue rechazada, pero no desplegó el mensaje.
3.	Teclado	Ingresar una selección válida del No. 15.	Desplegar el mensaje: "Ud. ha elegido chocolate".	Se comportó como se esperaba.
4.	Teclado	Ingresar una selección inválida del No. 99.	Desplegar el mensaje "Selección inválida, pruebe de nuevo".	Se comportó como se esperaba.

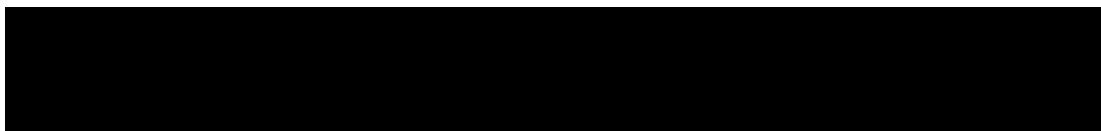
Las pruebas empíricas buscan demostrar que la solución es incorrecta, es decir, hacerla fallar. Es fácil, con tan solo encontrar un error. Pasar las pruebas de calidad no implica un resultado binario, sino una gama de niveles de opciones exitosas y otra de opciones fallidas. Estos niveles de una solución correcta se listan a continuación, ordenados del menos al más convincente.

- 1 La solución no contiene errores de sintaxis ni operaciones inválidas que puedan ser detectadas.
- 2 La solución arroja las respuestas correctas a algunas pruebas realizadas.
- 3 La solución arroja la respuesta correcta cuando se introducen datos aleatorios.
- 4 La solución arroja la respuesta correcta cuando se introducen datos deliberadamente incorrectos para dificultar el proceso.
- 5 La solución arroja la respuesta correcta para todos los datos que son posibles en relación con la especificación del problema original.
- 6 Para todas las entradas posibles, la solución arroja la respuesta correcta o respuestas razonables.

El reducir el número de pruebas fallidas no implica que la solución sea correcta. Tampoco la presencia de un error implica que la solución sea incorrecta. La corrección se refiere a que la solución resuelve el problema.

No importa si es lenta, ineficiente, muy complicada o a alguien no le gusta, estos factores no afectan que sea correcta.

Para ampliar más sobre la notación O (Big O), se puede consultar:





Complete Beginner's Guide to Big O Notation

Learn the basics of Big O Notation and Time Complexity in this crash course video. Learn how to evaluate and discuss the performance of different solutions

...

VER EN YOUTUBE >

YOUTUBE



Introduction to Big O Notation and Time Complexity (Data Structures & Algorithms #7)

Big O notation and time complexity, explained. Check out Brilliant.org (<https://brilliant.org/CSDojo/>), a website for learning math and computer science conce...

VER EN YOUTUBE >

Pruebe probar el rendimiento con esta herramienta:

RITHMSCHOOL

Big O Introduction

MÁS INFORMACIÓN RITHMSCHOOL >





```
2146:     scope:scope, element, attr, ngSwitchController) {  
2147:       var previousElements = attr.ngSwitch || attr.on,  
2148:           selectedTranscludes = [],  
2149:           selectedElements = [],  
2150:           previousElements = [],  
2151:           selectedScopes = [];  
2152:       scope.$watch(expr, function ngSwitchWatchAction(value) {  
2153:         var i, ii;  
2154:         for (i = 0, ii = previousElements.length; i < ii; ++i) {  
2155:           previousElements[i].remove();  
2156:         }  
2157:         previousElements.length = 0;  
2158:         for (i = 0, ii = selectedScopes.length; i < ii; ++i) {  
2159:           var selected = selectedElements[i];  
2160:           selectedScopes[i].$destroy();  
2161:           previousElements[i] = selected;  
2162:           $animate.leave(selected, function() {  
2163:             previousElements.splice(i, 1);  
2164:           });  
2165:         }  
2166:         selectedElements.length = 0;  
2167:         selectedScopes.length = 0;  
2168:       if ((selectedTranscludes = ngSwitchController.cases["!" + value] || ngSwitchC  
2169:           scope.$eval(attr.change);  
2170:           forEach(selectedTranscludes, function(selectedTransclude) {  
2171:             var selectedScope = scope.$new();  
2172:             selectedScopes.push(selectedScope);  
2173:           });  
2174:         }  
2175:       }  
2176:     }  
2177:   }  
2178: };
```

Cada algoritmo requiere una cantidad de recursos para funcionar. Diferentes algoritmos pueden resolver una solución con diferentes niveles de eficiencia. La eficiencia se mide en términos de tiempo y espacio.

TIEMPO:

ESPACIO:

La duración de un algoritmo en su tiempo de ejecución, desde el inicio hasta su fin. Se puede medir en el número de pasos que tome durante su ejecución.

TIEMPO:

ESPACIO:

La cantidad de memoria que requiera un algoritmo para funcionar.

Cada categoría describe cómo los algoritmos se desempeñan en el peor escenario. En otras palabras, la máxima cantidad de recursos que un algoritmo utiliza. La siguiente tabla muestra algunos ejemplos de categorías de complejidad de clases y su eficiencia en términos de tiempo.

Ejemplos de las principales complejidades de clase

Nombre	Tiempo de ejecución	Comentarios
Constante	$O(1)$	No necesariamente toma una unidad de tiempo, pero sí toma una cantidad fija de unidades de tiempo. Un ejemplo es determinar si un número es par o impar.
Logarítmica	$O(\log N)$	Según aumenta la cantidad de datos de entrada, el tiempo se incrementa cada vez de manera más pequeña. Las estrategias de divide y vencerás se comportan así.

Nombre	Tiempo de ejecución	Comentarios
Lineal	$O(N)$	Es el mejor rendimiento posible para cualquier algoritmo que se ejecute secuencialmente.
Cuadrático	$O(N^2)$	El tiempo de ejecución de los algoritmos en esta clase se cuadriplica cuando las entradas se duplican.
Factorial	$O(N!)$	Los algoritmos en esta clase son fuerza bruta y se consideran demasiado lentos para usos prácticos.

Una **complejidad de clase** es un perfil que indica cómo se comportarán los algoritmos que pertenecen a ellas.



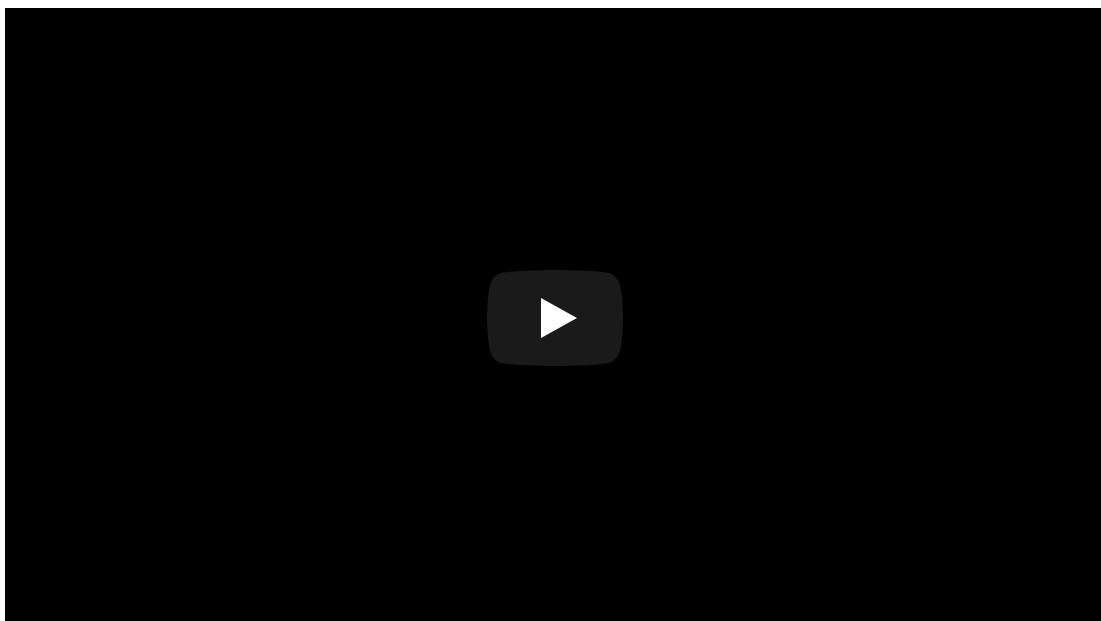
¿Es elegante?



La elegancia se refiere a la maximización entre la efectividad y la simplicidad al mismo tiempo.

Para un ingeniero, la elegancia se alcanza al desempeñarse en pocas partes y simples. Un ejemplo de esto es el reto que resolvió Gauss ante la solicitud de sumar los números del 1 al 100.

 YOUTUBE



Sumas de Gauss. Cómo sumar enteros consecutivos rápido

Obtención de la fórmula de Gauss para sumar enteros consecutivos

VER EN YOUTUBE >

La solución obvia era:

$$\text{Total} = 0$$

Para cada número, n , entre 1 y 100, hacer:

$$\text{Total} = \text{total} + n$$

La solución más elegante de Gauss fue:

$$\text{Total} = 50 \times 101$$

La solución de Gauss obtiene la misma respuesta pero utiliza menos espacio (una variable en lugar de dos) y menos tiempo (un paso en lugar de 100). En términos de complejidad, la solución obvia tiene un rendimiento de $O(N)$ y la de Gauss $O(1)$.

“Lo más simple puede ser más difícil que lo complejo. Es necesario esforzarse para pensar de manera limpia para hacerlo simple”, Jobs (1998).

Parece paradójico, pero es más difícil pensar de manera simple que compleja, ya que se debe conocer la esencia de un problema e identificar los patrones.

¿Es amigable?

En muchos casos, los usuarios de las soluciones serán humanos, por lo que la solución debe atender a los humanos y sus formas particulares de ser. Las soluciones deben ser **amigables** para brindar a los usuarios una experiencia positiva.

"Los criterios de ser correcta, eficiente y elegante describen a la solución en sus propios términos. Sin embargo, el ser amigable mide qué tanto algo podrá ser utilizado por las personas para alcanzar sus objetivos. Las personas no son máquinas, tienen deseos y emociones, cometen errores y su paciencia es limitada. Las personas desean soluciones que sean fáciles de aprender, con el menor esfuerzo posible."

La usabilidad captura esos deseos y los formaliza en componentes para medir que tan fácil y satisfactoria es para las personas una solución (Nielsen, 2012). Estos factores son:

Que se pueda aprender

—

Que tan fácil es para los usuarios realizar tareas básicas la primera vez para encontrar la solución.

Eficiencia —

Una vez que los usuarios hayan aprendido la solución, ¿qué tan rápido pueden realizar tareas?

Ser recordada —

Cuando los usuarios regresen a la solución después de un período sin usarla, ¿Qué tan fácil pueden volver a alcanzar la eficiencia?

Errores —

¿Cuántos errores cometen los usuarios? ¿Qué tan severos son y qué tan rápido se pueden recuperar de estos errores?

Satisfacción —

¿Qué tan agradable de usar es el diseño?

Las heurísticas de usabilidad son principios generales que se pueden aplicar cuando se diseña una solución. Si se aplican cuidadosamente, tendrá menos retrabajo en esta etapa. Jakob Nielsen (1995) lista las heurísticas:

NN NIELSEN NORMAN GROUP

10 Usability Heuristics

nngroup.com

NN/g

for User Interface Design

10 Usability Heuristics for User Interface Design

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time. When users know the current system status, they learn the outcome of their prior interactions and determine next steps. Predictable interactions create trust in the product as well as the brand.

[MÁS INFORMACIÓN NIELSEN NORMAN GROUP >](#)

El evaluar la usabilidad es al final de cuentas sobre las reacciones de las personas reales. Al integrar personas a la evaluación, se realizan estudios de observación. Las personas representan al grupo objetivo y el evaluador observa cómo trabajan, anotando las funciones que logran realizar con éxito y las que les generan dificultad.

Ejemplos de tareas de usabilidad:

- 1 Autenticarse en el sistema.
- 2 Agregar su correo a su perfil.
- 3 Comprar un libro de cocina por GTQ 100.00.
- 4 Mostrar la lista de los pedidos realizados.



Luego que las personas realizan las pruebas, el evaluador les da un cuestionario para que califiquen su experiencia. Las preguntas suelen ser genéricas. Por ejemplo:

Califique en escala de 1 (totalmente en desacuerdo) a 5 (totalmente de acuerdo) los siguientes enunciados.



En general, disfrute usar este sistema.

1 of 6



Encontré este sistema fácil de usar.

2 of 6



Me tomó mucho tiempo saber cómo se usa el sistema.

3 of 6



Tuve problemas al encontrar las funciones en este sistema.

4 of 6



Había muchos pasos para hacer una función en este sistema.

5 of 6





El sistema pasaba por alto cuando yo cometía errores.

6 of 6

Se debe permitir a los usuarios agregar sus propias experiencias y enriquecer las respuestas.

Las pruebas de usabilidad no necesariamente deben realizarse cuando la solución está completa. Se puede crear un bosquejo del sistema en papel, a manera de prototipo y permitir que los usuarios provean feedback. La regla clave es: mientras más pronto encuentre problemas, será más fácil resolverlos.

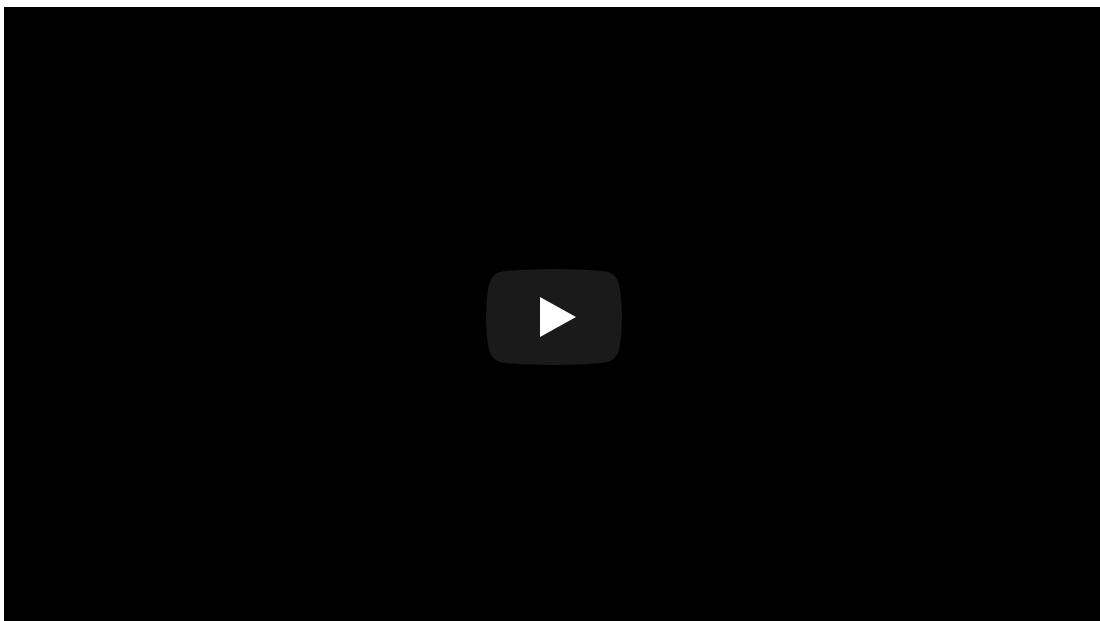


Comunicaciones

Compensaciones

Se puede abordar mucho sobre este tema, pero se simplificará a que cuando los datos contienen muchas repeticiones, pueden ser comprimidos para que ocupen menos espacio. Visualice el ejemplo de compensación entre tiempo y espacio.

 YOUTUBE



Time Space Tradeoff

what is our memory limit? How can save time at the expense of space?

VER EN YOUTUBE >

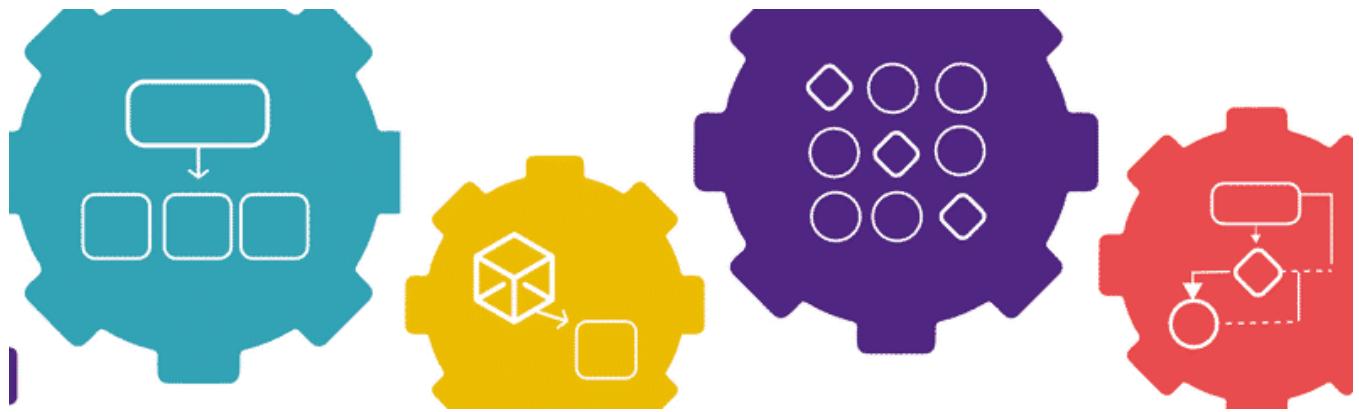


A manera de síntesis

La evaluación de una solución implica varios aspectos. Los más importantes son si la solución es correcta, es decir, si lo que hace resuelve el problema original.

La eficiencia mide el tiempo y el espacio utilizado en la solución para garantizar que resuelve el problema optimizando recursos. La elegancia determina si la solución maximiza la efectividad y la simplicidad.

En estos aspectos, las compensaciones optimizan aspectos como el almacenamiento, al realizar comprensión de datos repetitivos.



Recursos complementarios



Recursos complementarios

Notación Big O explicada

 YOUTUBE



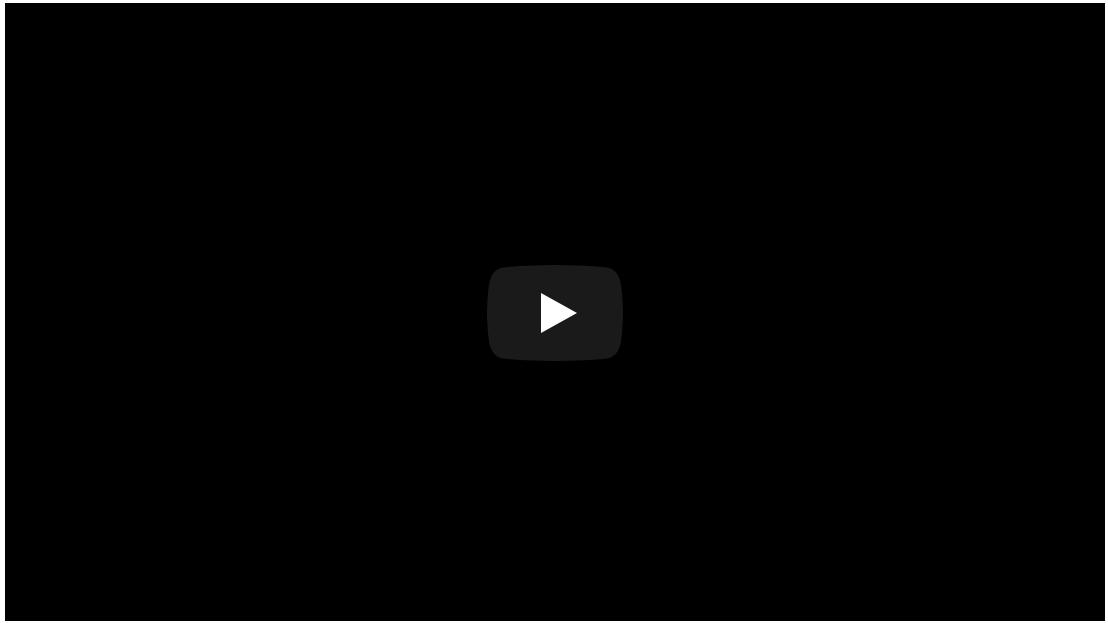
Notación Big O | Análisis de algoritmos de forma sencilla

Analizar nuestro código para determinar cómo se comporta con diferentes entradas no es tan difícil una vez aprendes la notación asintótica Big O.
Aprende cóm...

VER EN YOUTUBE >

Resolución y compresión

 YOUTUBE



¿Por qué la resolución es tan importante a la hora de imprimir?

<https://www.helloprint.es> En este video vamos a hablar sobre qué es la resolución y qué resolución establecer para sacar el máximo partido a tus archivos e i...

VER EN YOUTUBE >



Actividad 1



The background of the slide features a close-up photograph of two interlocking gears. One gear is a solid blue color, and the other is a solid yellow color. They are shown from a perspective angle, with light reflecting off their metallic surfaces.

Actividad 1



Instrucciones

Indique si los siguientes enunciados son verdaderos o falsos.

Las soluciones de software del mundo real comprueban que son correctas con pruebas empíricas, más que con pruebas matemáticas.

Verdadero



Falso

SUBMIT

Reducir el número de pruebas que fallan no mejora el hecho que la solución sea correcta.



Verdadero



Falso

SUBMIT

Una solución ineficiente no puede ser correcta.



Verdadero



Falso

SUBMIT

Una solución elegante maximiza la efectividad y la simplicidad al mismo tiempo.



Verdadero



Falso

SUBMIT

La evaluación de la usabilidad inicia con explicar cómo funciona el sistema a un ser humano.

Verdadero

Falso

SUBMIT

Actividad 2



Actividad 2



Instrucciones

Escriba las soluciones genéricas de la solución obvia y la de Gauss, considerando los números de 1 a N.

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

Actividad 3



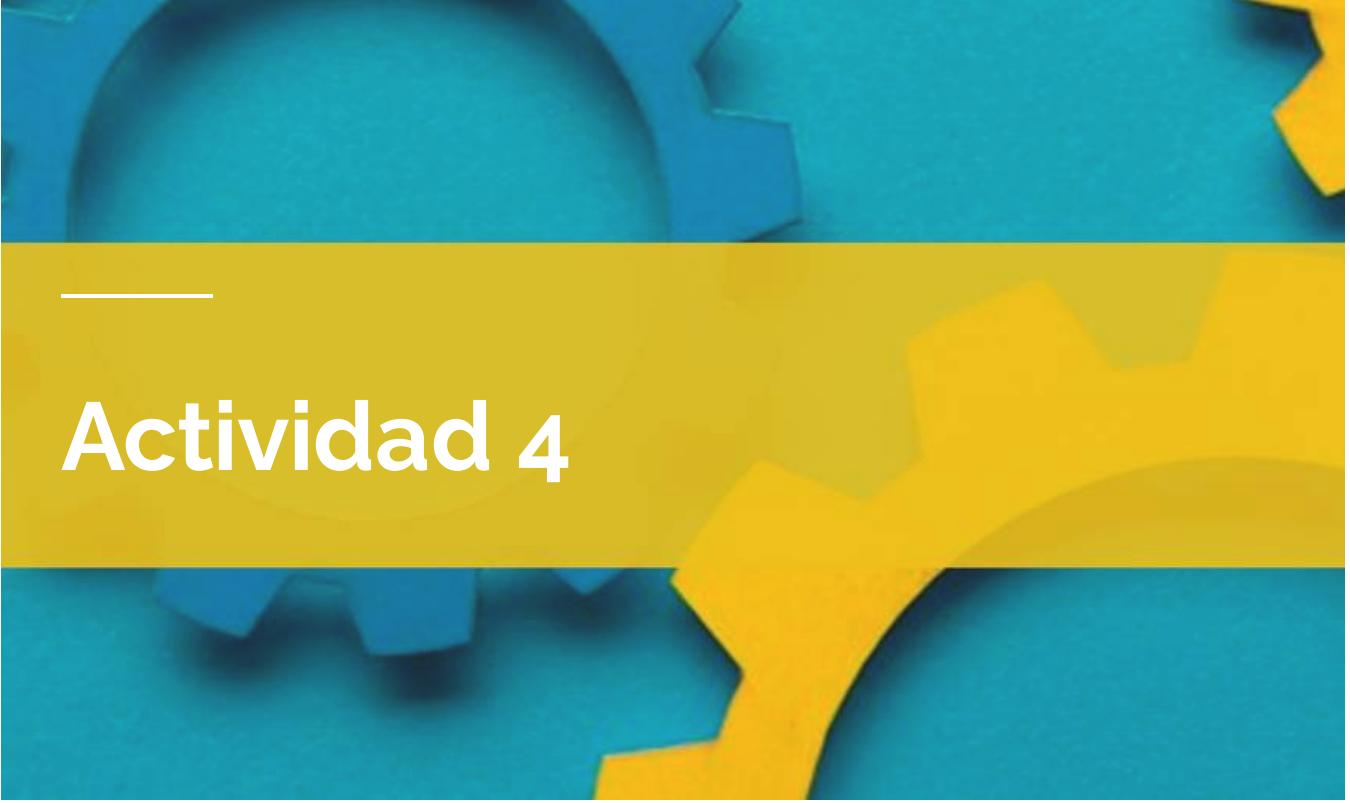
Actividad 3



Con los algoritmos de la actividad 2, ¿cuál de los dos es más eficiente en términos de complejidad y por qué?

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

Actividad 4



Actividad 4



Instrucciones

¿Cuáles son los nombres de los cinco componentes de usabilidad que se miden?

Diríjase a la plataforma para subir su actividad en el espacio correspondiente.

Activación de conocimientos previos



Recursos

Instrucciones:

Consultar con su profesor sobre los retos a realizar en el laboratorio y los materiales que se deben llevar.



Rúbrica de evaluación

Rúbrica





Descargue la siguiente rúbrica de evaluación.



Rúbrica de Evaluación.pdf

142.3 KB

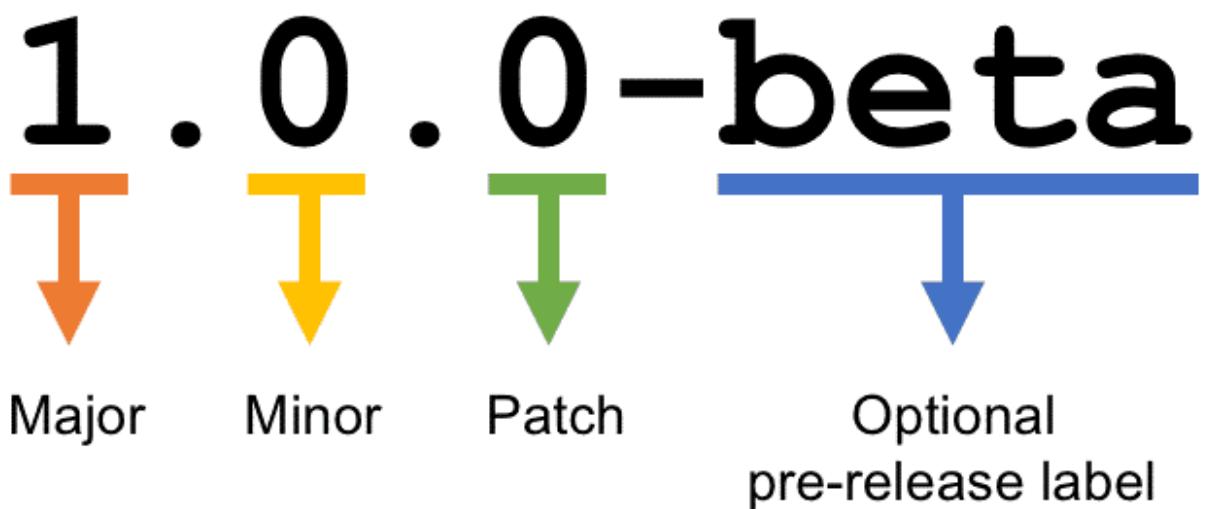


Diario de experiencias de laboratorio



Diario de experiencias del laboratorio

Cada práctica de laboratorio deberá publicarse en una carpeta en el diario de experiencias de laboratorio, bajo la versión 1.0.0.



Referencias

Referencias

Bordigon, F. e Iglesias, A. (2020). Introducción al pensamiento computacional. Universidad Pedagógica Nacional, Educar Sociedad del Estado y el Ministerio de Educación Argentina.

Beecher, K. (2017). Computational Thinking. A beginner's guide to problem-solving and programming.

ISTE (2018). Computational Thinking. Meets Students Learning. International Society for Technology in Education.

MIT (2022). Computational Thinking, a live online Julia/Pluto textbook. Julia: A Fresh Approach to Computing. computationalthinking.mit.edu. Massachusetts Institute of Technology.

Pourbahrami & Tritsy (2018). Computational Thinking: How Computer Science Is Revolutionizing Science and Engineering. ENGenious (15) 8-11.
<https://resolver.caltech.edu/CaltechCampusPubs:20181025-110029157>.

Créditos

