Cómo hacer un TP de AlgoIII

y no reentregar en el intento

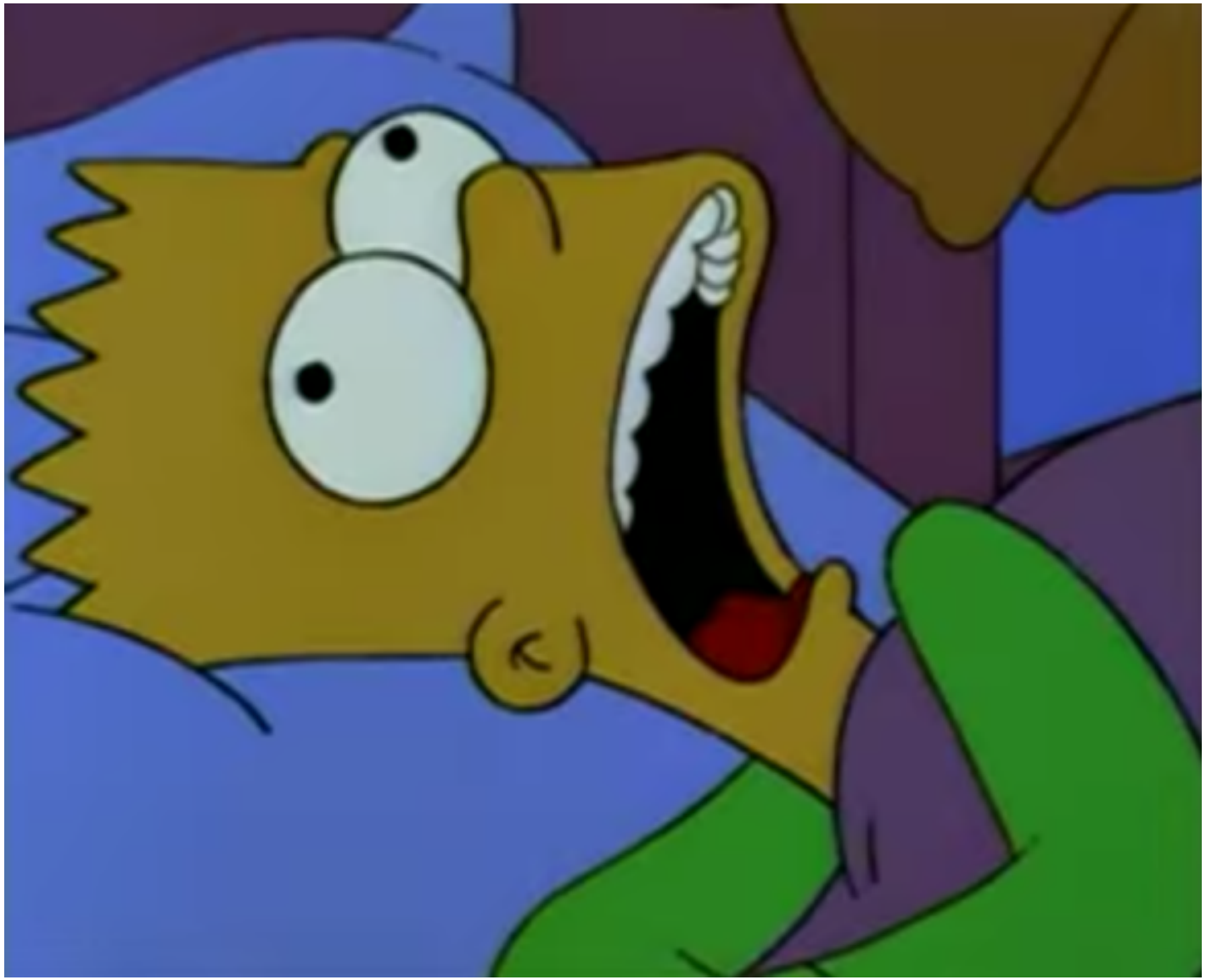
DC - FCEN - UBA

15 de agosto de 2016

Bart, no quiero asustarte ...

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.



Bart, no quiero asustarte ...

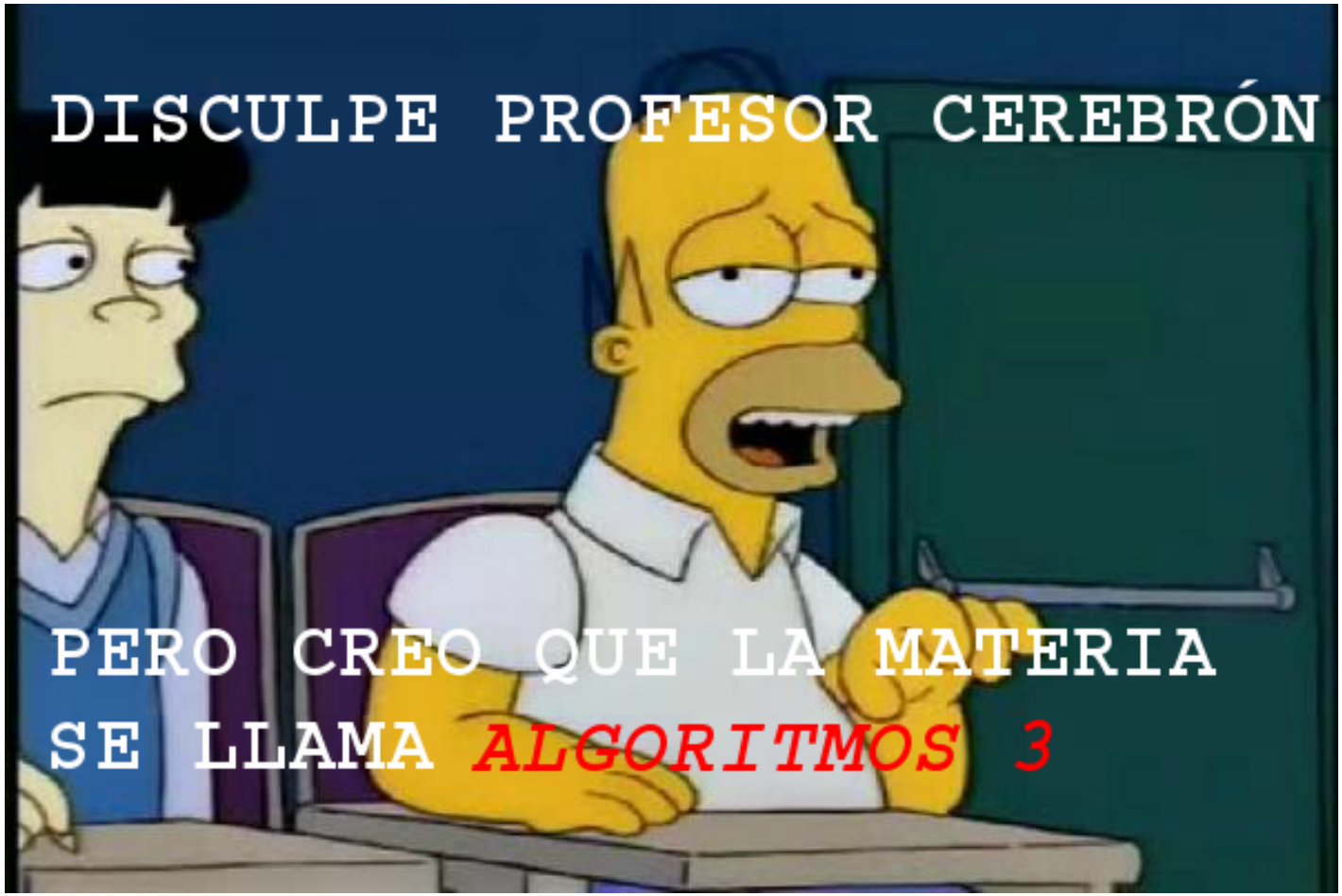
... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.



Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.
- *explicar* la solución propuesta.

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.
- *explicar* la solución propuesta.
- *demostrar* que el algoritmo propuesto es efectivamente una solución al problema (*i.e.* correctitud[1]).

[1] `http://dle.rae.es/?w=correctitud`

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.
- *explicar* la solución propuesta.
- *demostrar* que el algoritmo propuesto es efectivamente una solución al problema (*i.e.* correctitud[1]).
- *demostrar* que la solución cumple con una cota de complejidad dada.

[1] `http://dle.rae.es/?w=correctitud`

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

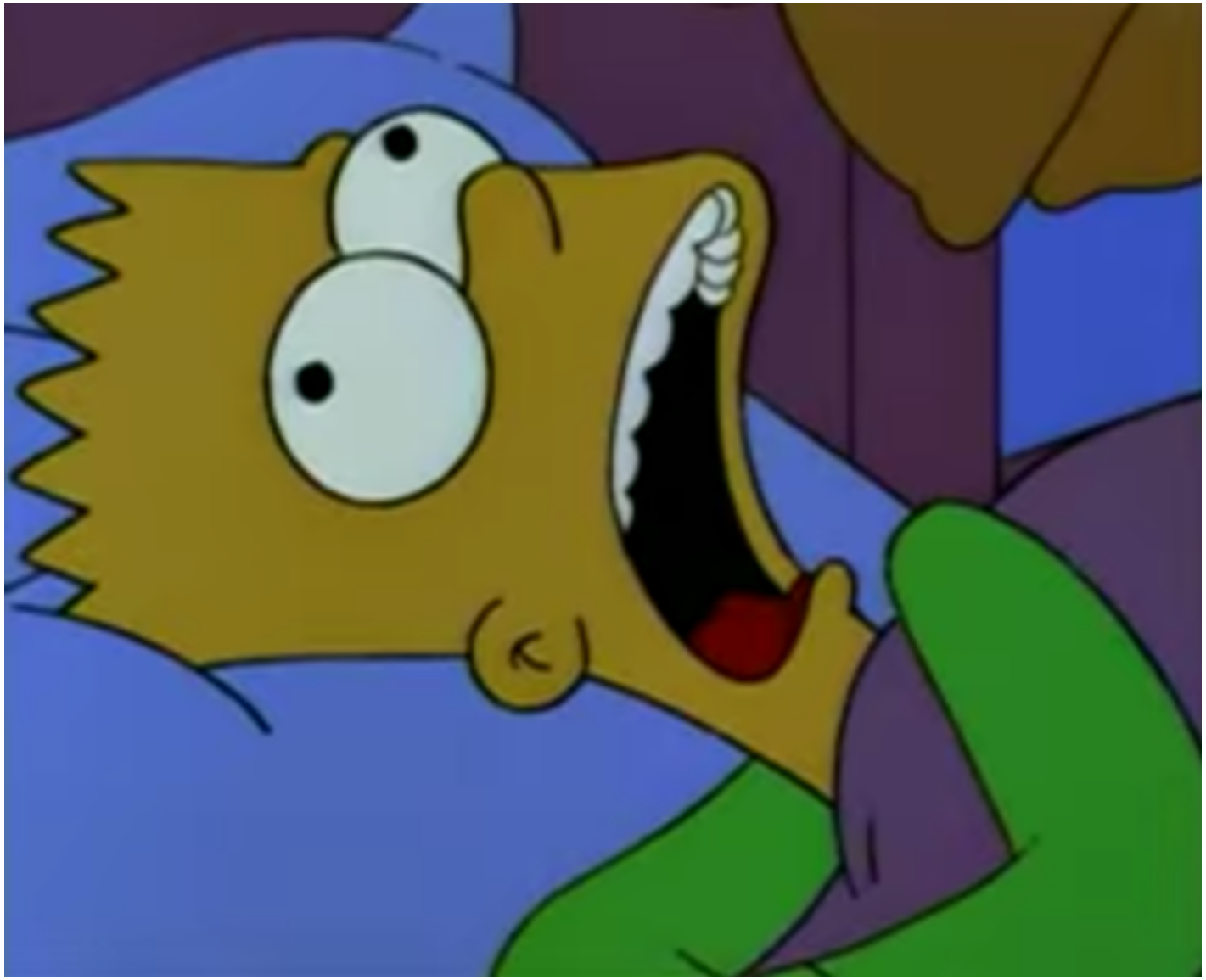
Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.
- *explicar* la solución propuesta.
- *demostrar* que el algoritmo propuesto es efectivamente una solución al problema (*i.e.* correctitud[1]).
- *demostrar* que la solución cumple con una cota de complejidad dada.
- poner a prueba la solución y *mostrar* cómo se comporta (*i.e.* experimentación)

[1] `http://dle.rae.es/?w=correctitud`

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.



Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.
- *explicar* la solución propuesta.
- *demostrar* que el algoritmo propuesto es efectivamente una solución al problema (*i.e.* correctitud[1]).
- *demostrar* que la solución cumple con una cota de complejidad dada.
- poner a prueba la solución y *mostrar* cómo se comporta (*i.e.* experimentación)

[1] `http://dle.rae.es/?w=correctitud`

Bart, no quiero asustarte ...

... pero tal vez los TPs de Algo3 son con informe.

Básicamente, los TPs de esta materia (y de muchas otras que vendrán) son el informe.
Sí, estamos en una materia de algoritmos. Pero justamente, dado que son nuestro objeto de estudio, y vamos (dijo el mosquito) a estudiarlos en profundidad, no alcanza con programarlos. Además hay que:

- *enunciar* (in)formalmente el problema que resuelve el algoritmo.
- *explicar* la solución propuesta.
- *demostrar* que el algoritmo propuesto es efectivamente una solución al problema (*i.e.* correctitud[1]).
- *demostrar* que la solución cumple con una cota de complejidad dada.
- poner a prueba la solución y *mostrar* cómo se comporta (*i.e.* experimentación)

Y mucho más, que veremos a continuación.

[1] `http://dle.rae.es/?w=correctitud`

Introducción

"Let's start at the very beginning, a very good place to start."[2]

En la introducción, se espera introducir (brillante!) al lector en el problema que se va a resolver, dando por sentado que ni siquiera leyó el enunciado. Debe contener:

1. el enunciado informal del problema (si existe), brevemente.
2. el enunciado formal del problema.
3. ejemplos, casos interesantes, explicaciones *en palabras* que ayuden a comprender qué problema se está tratando de resolver.
4. no mucho más.

[2] `https://youtu.be/5Ugr0eO9SFM`

Introducción

"Let's start at the very beginning, a very good place to start." [2]

En la introducción, se espera introducir (brillante!) al lector en el problema que se va a resolver, dando por sentado que ni siquiera leyó el enunciado. Debe contener:

1. el enunciado informal del problema (si existe), brevemente.
2. el enunciado formal del problema.
3. ejemplos, casos interesantes, explicaciones *en palabras* que ayuden a comprender qué problema se está tratando de resolver.
4. no mucho más.

El objetivo de esta sección es que quede claro de qué se va a tratar el resto del informe, con lo cual, deben tener en cuenta, respectivamente:

1. que la informalidad sea accesible pero precisa.
2. que la formalidad evite ambigüedades pero sin complicar la lectura.
3. que los ejemplos sean concisos, fáciles de entender y muestren claramente lo que les parece relevante del problema.
4. que sea breve.

[2] `https://youtu.be/5Ugr0e09SFM`

Enunciado de la vida real

Gokú se está enfrentando a N androides y necesita destruirlos con la menor cantidad de Kamehamehas posibles. Los enemigos de Gokú se encuentran en posiciones (X_i, Y_i) y los Kamehameha recorren una semirrecta desde donde Gokú lo lance, en cualquier dirección que Gokú lo decida. ¿Cuántos Kamehamehas necesita Gokú para desturir a todos los androides del doctor Maki Gero?

Introducción Informal ☹

El problema consiste optimizar la cantidad de Kamehameha's que Gokú debe lanzar para eliminar todos los androides (ubicados en posiciones (X_i, Y_i)).

Introducción Informal ☹

El problema consiste optimizar la cantidad de Kamehameha's que Gokú debe lanzar para eliminar todos los androides (ubicados en posiciones (X_i, Y_i)).

Cosas que no nos gustan:

- "Androides", "Kamehamehas", "Gokú"? WTF?!
- No aporta nada decir que están ubicados en (X_i, Y_i), aunque el latex quede más lindo.
- Si no leí el enunciado, no hay chance de que entienda de qué están hablando.

Introducción Informal ☺

En este problema, Gokú debe destruir una cantidad de androides lanzando Kamehameha's. Un Kamehameha se lanza desde cualquier punto en una dirección, avanza en línea recta (en un único sentido) y destruye a todos los enemigos que encuentre a su paso. El objetivo es encontrar la manera de eliminar todos los androides lanzando la mínima cantidad de Kamehameha's posible.

Introducción Informal ☺

En este problema, Gokú debe destruir una cantidad de androides lanzando Kamehameha's. Un Kamehameha se lanza desde cualquier punto en una dirección, avanza en línea recta (en un único sentido) y destruye a todos los enemigos que encuentre a su paso. El objetivo es encontrar la manera de eliminar todos los androides lanzando la mínima cantidad de Kamehameha's posible.

Cosas que nos gustan

- Se explica el problema en el contexto en el que fue enunciado.
- Además, se explica qué significa cada cosa que es propia del problema.
- Se dice, de manera concreta y clara, *qué* problema en concreto se está buscando resolver.

Introducción Formal ☹

Formalmente, dado un conjunto de pares $P = \{(X_1, Y_1), ..., (X_n, Y_n)\}$ buscamos un conjunto $L = \{\langle (p1_{x_1}, p1_{y_1}), (p2_{x_1}, p2_{y_1})\rangle, ..., \langle (p1_{x_k}, p1_{y_k}), (p2_{y_k}, p2_{y_k})\rangle\}$ tal que $\forall (x, y) \in P\ \exists i\ 1 \leq i \leq k$ tal que (x, y) está en la recta determinada por $(p1_{x_i}, p1_{y_i})$ y $(p2_{y_i}, p2_{y_i})$, y además, para todo otro conjunto L' que cumpla esto, se tiene $\#L \leq \#L'$.

Introducción Formal ☹

Formalmente, dado un conjunto de pares $P = \{(X_1, Y_1), ..., (X_n, Y_n)\}$ buscamos un conjunto $L = \{\langle (p1_{x_1}, p1_{y_1}), (p2_{x_1}, p2_{y_1})\rangle, ..., \langle (p1_{x_k}, p1_{y_k}), (p2_{y_k}, p2_{y_k})\rangle\}$ tal que $\forall (x, y) \in P\ \exists i\ 1 \leq i \leq k$ tal que (x, y) está en la recta determinada por $(p1_{x_i}, p1_{y_i})$ y $(p2_{y_i}, p2_{y_i})$, y además, para todo otro conjunto L' que cumpla esto, se tiene $\#L \leq \#L'$.

Antiguo Proverbio Chino

Cuando pongas $ procura que tus símbolos sean más sabios que el texto plano.

Introducción Formal ☺

Formalmente, esto es equivalente a, dado un conjunto de puntos en el plano, encontrar la manera de cubrirlos utilizando la menor cantidad de rectas posible.

Introducción Formal ☺

Formalmente, esto es equivalente a, dado un conjunto de puntos en el plano, encontrar la manera de cubrirlos utilizando la menor cantidad de rectas posible.

Cosas que nos gustan

- La explicación es lo más abstracta posible (no habla de androides y gokúes, sino de puntos y rectas), pero sin perder sentido.
- Es formal porque utiliza conceptos formales, aunque esté expresada en lenguaje natural.
- Es muy clara y para nada ambigua.

Esto NO quiere decir que estemos en contra de los símbolos y a favor del lenguaje natural, pero en este caso eran innecesarios. Como dijo el General: *"Todo en su justa medida y armoniosamente"*.

Naturalmente, la explicación formal debe corresponderse con la informal, pero deben decir cómo y por qué.

Ejemplo ☺:

EXPLICACIÓN INFORMAL ☺ ... podemos pensar a los androides como un punto en el plano y a los Kamehameha's como rectas con origen en un punto, con lo cual, formalmente ... EXPLICACIÓN FORMAL ☺ .

- En este caso, puede parecer trivial la conexión, pero de todas maneras debe estar explícita.
- En problemas con modelos un poco más complejos (como un grafo, tal vez...) esta etapa no es para nada trivial y debe estar adecuadamente explicado y justificado cómo se corresponde el problema concreto con el modelo abstracto y la solución sobre éste. Ante cualquier duda, consulte a su ayudante amigo del labo.

Explicación de la solución

Esta sección incluye una presentación de la solución, la solución en sí y las correspondientes demostraciones de correctitud y complejidad que avalen la solución planteada.

- La presentación de la solución debería servir a modo introductorio y se espera que esté escrita en lenguaje natural y explique, sin entrar en demasiado detalle, cómo funciona el algoritmo propuesto.
- La solución en sí, en general se presenta utilizando pseudocódigo. Más sobre esto más adelante.
- La rigurosidad de la demostración de correctitud dependerá del ejercicio. Traten siempre de ser lo más precisos posible, pero si se complica mucho escribir la demo, consulten.
- Las demostraciones de complejidad pueden ser más *habladas*, pero siempre y cuando sean rigurosos y no estén escondiendo costos ni pasando por alto argumentos que no se deducen directamente del pseudocódigo. Traten siempre de demostrar la cota más ajustada que puedan, aún si es menor que la pedida.

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

```
res ← a[0]
for i = 1; i < a.size(); ++i do
    if a[i] < res then
        res ← a[i]
    end if
end for
return res
```

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

$res \leftarrow a[0]$
for $i = 1, \ldots, |a| - 1$ **do**
    if $a[i] < res$ **then**
        $res \leftarrow a[i]$
    end if
end for
return res

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

$res \leftarrow a[0]$
for $i = 1, \ldots, |a| - 1$ **do**
  if $a[i] < res$ **then**
    $res \leftarrow a[i]$
  end if
end for
return *res*

return $\min_{0 \leq i < |a|} a[i]$

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

```
res ← a[0]
for i = 1, ..., |a| − 1 do
    if a[i] < res then
        res ← a[i]
    end if
end for
return res
```

return mín a

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

```
res ← a[0]
for i = 1, ..., |a| − 1 do
    if a[i] < res then
        res ← a[i]
    end if
end for
return res
```

return mín a

Pseudocódigo

pseudo significa falso y *código* significa código.

Utilizamos falso código como una herramienta para exponer algoritmos porque es mucho más cómodo de escribir que el castellano pero no para mostrar el código tal cual lo programaron. Al contrario, el pseudocódigo permite escribir algunas abstracciones que pueden hacer que la explicación del algoritmo sea mucho más clara.

```
res ← a[0]
for i = 1, ..., |a| − 1 do
    if a[i] < res then
        res ← a[i]
    end if
end for
return res
```

return mín a

La idea es aprovechar que no tenemos que escribir código que compile para mostrar lo mejor posible el algoritmo, concentrándonos en sus partes importantes y escondiendo cosas poco interesantes o detalles de implementación.

Pseudocódigo

Otro ejemplo

```
p ← new Pila()
visitados ← new Array < Bool > (n)
for i = 0; i < n; + + i do
    visitados[i] ← False
end for
p.push(vertices[0])
visitados[0] ← True
while p.size() > 0 do
    v ← p.pop()
    for w in v.adyacentes do
        if visitados[w.id] == False then
            p.push(w)
            visitados[w.id] ← True
        end if
    end for
end while
```

Pseudocódigo

Otro ejemplo

```
p ← Pila()
visitados ← Array < Bool > (n)
for i = 0; i < n; + + i do
    visitados[i] ← False
end for
p.push(vertices[0])
visitados[0] ← True
while p.size() > 0 do
    v ← p.pop()
    for w in v.adyacentes do
        if visitados[w.id] == False then
            p.push(w)
            visitados[w.id] ← True
        end if
    end for
end while
```

Pseudocódigo

Otro ejemplo

```
p ← Pila()
visitados[n] ← {False, ..., False}
p.push(vertices[0])
visitados[0] ← True
while p.size() > 0 do
    v ← p.pop()
    for w in v.adyacentes do
        if visitados[w.id] == False then
            p.push(w)
            visitados[w.id] ← True
        end if
    end for
end while
```

Pseudocódigo

Otro ejemplo

$p \leftarrow Pila()$
$visitados[n] \leftarrow \{False, ..., False\}$
$p.push(vertices[0])$
$visitados[0] \leftarrow True$
while $p.size() > 0$ **do**
    $v \leftarrow p.pop()$
    for w in $v.adyacentes$ **do**
        if $\neg visitados[w.id]$ **then**
            $p.push(w)$
            $visitados[w.id] \leftarrow True$
        end if
    end for
end while

Pseudocódigo

Otro ejemplo

$p \leftarrow Pila()$
$visitados[n] \leftarrow \{False, ..., False\}$
$p.push(vertices[0])$
$visitados[0] \leftarrow True$
while $p.size() > 0$ **do**
    $v \leftarrow p.pop()$
    for w in $v.adyacentes$ **do**
        if $\neg visitados[w.id]$ **then**
            $p.push(w)$
            $visitados[w.id] \leftarrow True$
        end if
    end for
end while

Pseudocódigo

Otro ejemplo

$p \leftarrow$ pila vacia
Apilar un primer vértice
Marcarlo como visitado
while p no está vacía **do**
    $v \leftarrow p.pop()$
    for cada vecino w de v **do**
        if w no está visitado **then**
            Apilar w en p
            Marcar w como visitado
        end if
    end for
end while

Pseudocódigo

Otro ejemplo

$p \leftarrow$ pila vacia
Apilar un primer vértice
Marcarlo como visitado
while p no está vacía **do**
    $v \leftarrow p.pop()$
    for cada vecino w de v **do**
        if w no está visitado **then**
            Apilar w en p
            Marcar w como visitado
        end if
    end for
end while

Correctitud y Complejidad

- Correctitud:
  - En esta sección se espera una demostración formal de que el algoritmo es correcto, es decir, una argumentación (probablemente sobre el pseudocódigo que presentaron) ordenada y suficientemente justificada que permita concluir que el resultado que devuelve el algoritmo es una solución al problema.
  - Las ideas o las intuiciones de por qué funciona deberían aparecer en la parte de explicación, no acá.
  - Ojo con el síndrome de *MiAlgoritmoEsCorrectoPorqueSoyCapo*.

Correctitud y Complejidad

- Complejidad:
  - En esta sección se espera que propongan una cota de complejidad y demuestren que su algoritmo la cumple, haciendo un análisis adecuado:
    - Si un ciclo itera $\mathcal{O}(n)$ veces y su cuerpo es $\mathcal{O}(n)$, no necesariamente el costo de todo eso es $\mathcal{O}(n^2)$.
  - También deberán identificar mejores y peores casos.
    - Ojo cuando la complejidad depende de más de un parámetro: $\mathcal{O}(|a| * p)$, *donde p es la cantidad de números pares que hay en a. Un peor caso es cuando a contiene sólo números pares.*

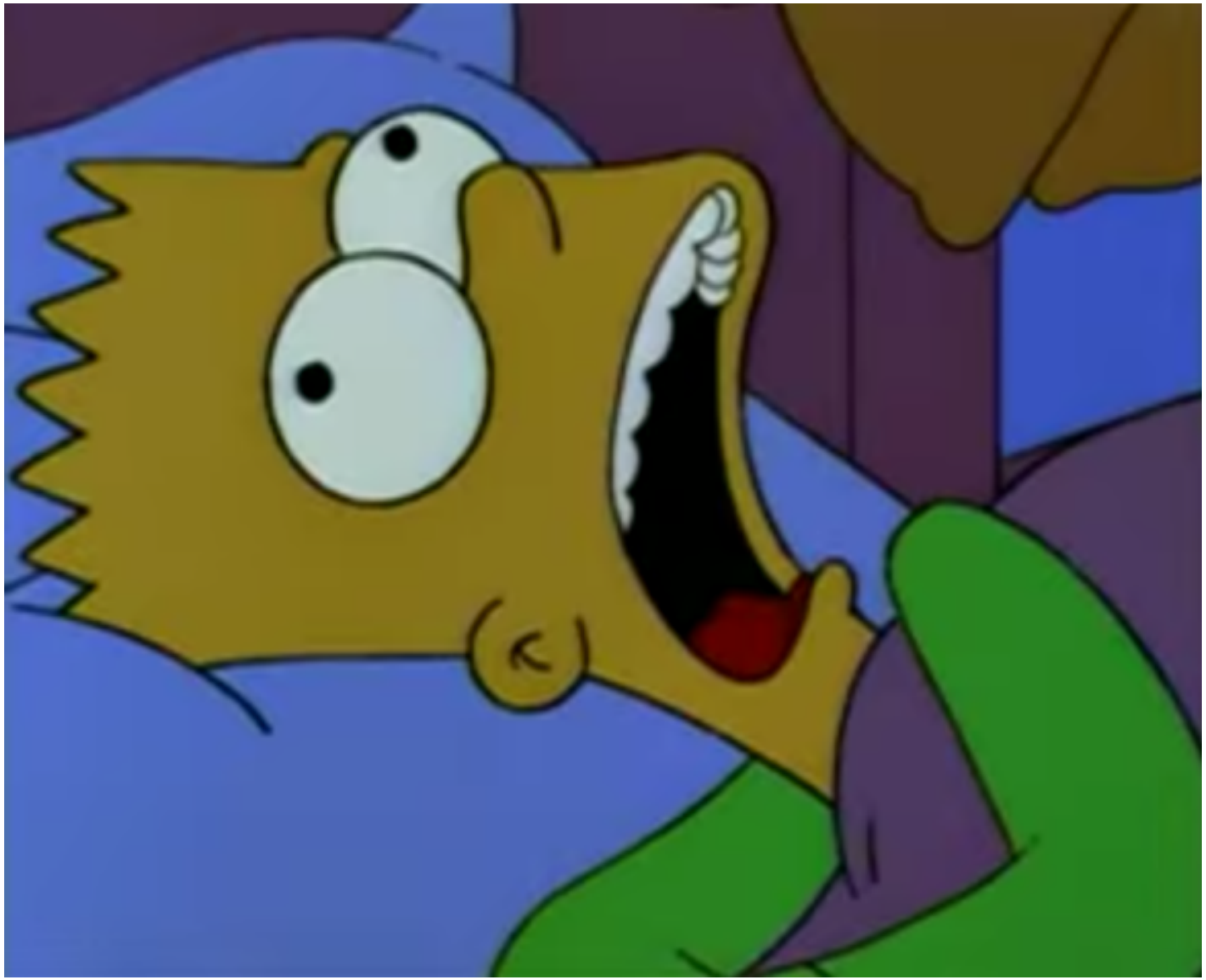
Correctitud y Complejidad

- Complejidad:
  - En esta sección se espera que propongan una cota de complejidad y demuestren que su algoritmo la cumple, haciendo un análisis adecuado:
    - Si un ciclo itera $\mathcal{O}(n)$ veces y su cuerpo es $\mathcal{O}(n)$, no necesariamente el costo de todo eso es $\mathcal{O}(n^2)$.
  - También deberán identificar mejores y peores casos.
    - Ojo cuando la complejidad depende de más de un parámetro:
      $\mathcal{O}(|a| * p)$, *donde p es la cantidad de números pares que hay en a.*
      Un peor caso es cuando a contiene sólo números pares. NO!
      Es un caso distinto.

Correctitud y Complejidad

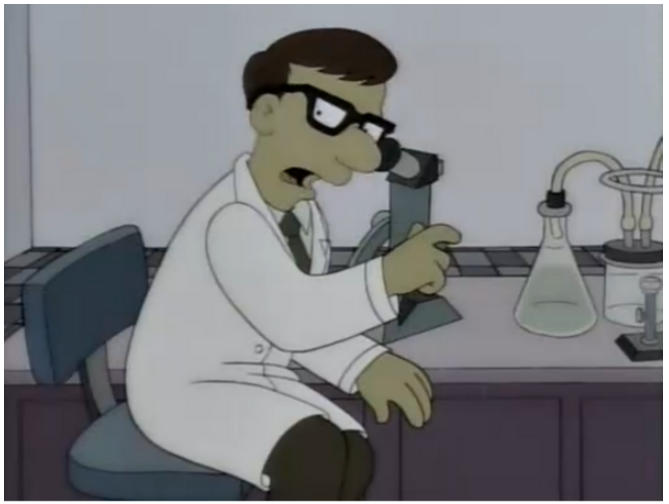
- Complejidad:
  - En esta sección se espera que propongan una cota de complejidad y demuestren que su algoritmo la cumple, haciendo un análisis adecuado:
    - Si un ciclo itera $\mathcal{O}(n)$ veces y su cuerpo es $\mathcal{O}(n)$, no necesariamente el costo de todo eso es $\mathcal{O}(n^2)$.
  - También deberán identificar mejores y peores casos.
    - Ojo cuando la complejidad depende de más de un parámetro: $\mathcal{O}(|a| * p)$, *donde p es la cantidad de números pares que hay en a. Un peor caso es cuando a contiene sólo números pares.* NO! Es un caso distinto.
  - Si usan código ajeno (*i.e.* ~~librerías~~ bibliotecas) deberán incluir referencias claras y confiables para la complejidad de las operaciones utilizadas.

Experimentación



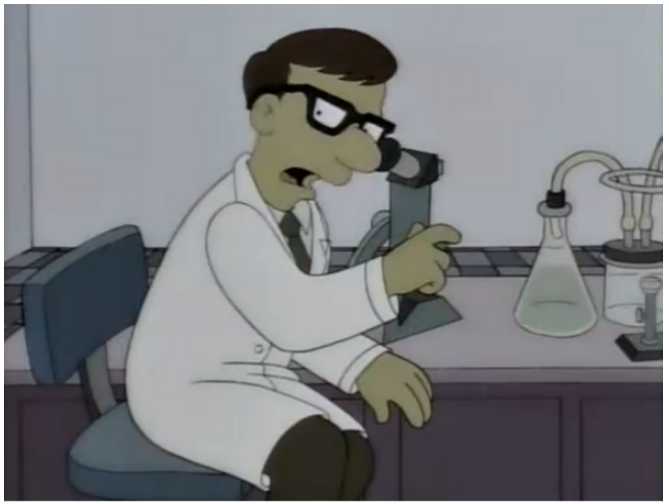
Experimentación

Para nosotros, la experimentación suele ser la parte más importante del TP dado que es la única instancia en la que se evalúa. Con lo cual, esperamos que tanto el diseño de los experimentos como sus resultados y análisis sean no sólo correctos, sino también interesantes (en el sentido de no triviales) y estén adecuadamente presentados y justificados.



Experimentación

Para nosotros, la experimentación suele ser la parte más importante del TP dado que es la única instancia en la que se evalúa. Con lo cual, esperamos que tanto el diseño de los experimentos como sus resultados y análisis sean no sólo correctos, sino también interesantes (en el sentido de no triviales) y estén adecuadamente presentados y justificados.



¡Mucho más sobre esto en las próximas clases!

☺☺☺☺☺☺☺

Experimentación

Experimentos

- En general, la experimentación tendrá que ver con el tiempo de corrida del algoritmo. Con lo cual, debería incluir:

Experimentación

Experimentos

- En general, la experimentación tendrá que ver con el tiempo de corrida del algoritmo. Con lo cual, debería incluir:
  - una experimentación básica que muestre que la cota planteada de complejidad tiene sentido.

Experimentación

Experimentos

- En general, la experimentación tendrá que ver con el tiempo de corrida del algoritmo. Con lo cual, debería incluir:
  - una experimentación básica que muestre que la cota planteada de complejidad tiene sentido.
  - algún experimento que muestre que el mejor caso es bueno y el peor caso es malo.

Experimentación

Experimentos

- En general, la experimentación tendrá que ver con el tiempo de corrida del algoritmo. Con lo cual, debería incluir:
  - una experimentación básica que muestre que la cota planteada de complejidad tiene sentido.
  - algún experimento que muestre que el mejor caso es bueno y el peor caso es malo.
  - otros experimentos que les parezcan relevantes y pongan de manifiesto alguna característica del algoritmo (que ya deberían haber comentado en algún lado).

Experimentación

Experimentos

- En general, la experimentación tendrá que ver con el tiempo de corrida del algoritmo. Con lo cual, debería incluir:
  - una experimentación básica que muestre que la cota planteada de complejidad tiene sentido.
  - algún experimento que muestre que el mejor caso es bueno y el peor caso es malo.
  - otros experimentos que les parezcan relevantes y pongan de manifiesto alguna característica del algoritmo (que ya deberían haber comentado en algún lado).
  - una explicación y justificación de cómo van a hacer el experimento (cómo van a generar las instancias de prueba, cuántas corridas de cada uno van a tomar, etc.).

Experimentación

Presentación de los resultados

- También es importante la adecuada presentación de los datos.

- También es importante la adecuada presentación de los datos.
  - Todos los gráficos llevan **título**, **rótulos en los ejes** (indicando unidades, si corresponde), **número** (que después se usa para referirse a él como *el gráfico número 7*) y **epígrafe** que diga, por lo menos, qué es lo que se está graficando.

- También es importante la adecuada presentación de los datos.
  - Todos los gráficos llevan **título**, **rótulos en los ejes** (indicando unidades, si corresponde), **número** (que después se usa para referirse a él como *el gráfico número 7*) y **epígrafe** que diga, por lo menos, qué es lo que se está graficando.
  - No siempre la mejor manera de presentar datos es mediante un gráfico (*i.e.* agarrar los datos de todos los experimentos, plotear un gráfico para cada uno y ponerlo en el informe con `\begin{figure}` no siempre es la solución.)

- También es importante la adecuada presentación de los datos.
  - Todos los gráficos llevan **título**, **rótulos en los ejes** (indicando unidades, si corresponde), **número** (que después se usa para referirse a él como *el gráfico número 7*) y **epígrafe** que diga, por lo menos, qué es lo que se está graficando.
  - No siempre la mejor manera de presentar datos es mediante un gráfico (*i.e.* agarrar los datos de todos los experimentos, plotear un gráfico para cada uno y ponerlo en el informe con `\begin{figure}` no siempre es la solución.)
  - Ojo cuando las cosas dependen de más de una variable.

Experimentación

Análisis

- Finalmente, se espera un análisis de los datos presentados.

Experimentación

Análisis

- Finalmente, se espera un análisis de los datos presentados.
  - En algunos casos no va a ser un análisis mucho más profundo que *"...efectivamente, nuestra hipótesis es correcta..."*, pero no siempre.

- Finalmente, se espera un análisis de los datos presentados.
  - En algunos casos no va a ser un análisis mucho más profundo que *"...efectivamente, nuestra hipótesis es correcta..."*, pero no siempre.
  - No abusen del *se ve* y ojo con las alucinaciones (*i.e.* decir que ven cosas en los gráficos que NO están ahí).

Experimentación

Análisis

- Finalmente, se espera un análisis de los datos presentados.
  - En algunos casos no va a ser un análisis mucho más profundo que *"...efectivamente, nuestra hipótesis es correcta..."*, pero no siempre.
  - No abusen del *se ve* y ojo con las alucinaciones (*i.e.* decir que ven cosas en los gráficos que NO están ahí).
  - Todo lo que digan en esta sección tiene que estar adecuadamente justificado a partir de los resultados de los experimentos que presentaron. Con lo cual, es tan importante que los datos estén bien presentados como que sean correctos.

Conclusiones

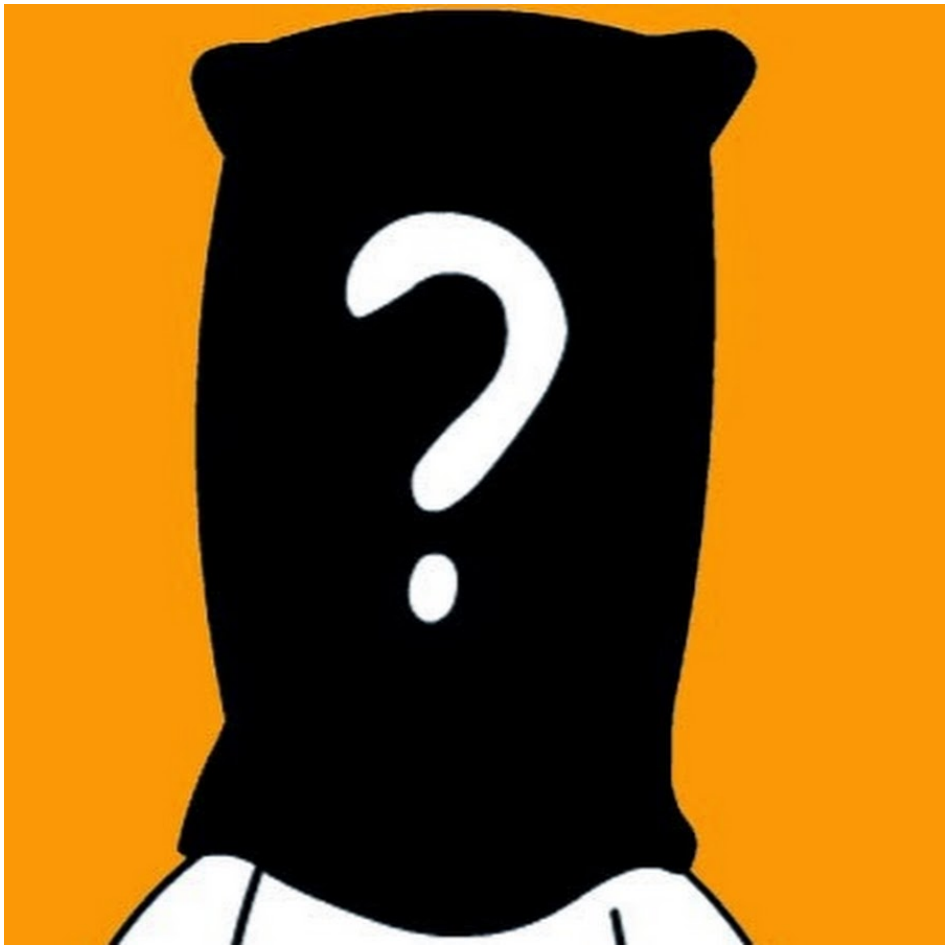
El informe es tanto una presentación y explicación de la solución, como su justificación con argumentos suficientes (demostraciones, mediciones, referencias, etc). Por lo tanto:

- debe ser completo.
- debe ser (casi) autocontenido. Es decir, si bien puede contener referencias a resultados conocidos, **no debería ser necesario para entenderlo leer el enunciado del TP ni mirar el código**.
- debe ser **claro en su redacción**, dado que es la manera que tienen ustedes de comunicarnos a nosotros su solución y sus argumentos.

Conclusiones

El informe es tanto una presentación y explicación de la solución, como su justificación con argumentos suficientes (demostraciones, mediciones, referencias, etc). Por lo tanto:

- debe ser completo.
- debe ser (casi) autocontenido. Es decir, si bien puede contener referencias a resultados conocidos, **no debería ser necesario para entenderlo leer el enunciado del TP ni mirar el código**.
- debe ser **claro en su redacción**, dado que es la manera que tienen ustedes de comunicarnos a nosotros su solución y sus argumentos.
- ¡NO LO DEJEN PARA ÚLTIMO MOMENTO!
- Aprovechen las consultas del labo para preguntar dudas del informe.



¿¿¿Preguntas???