Algoritmos golosos I

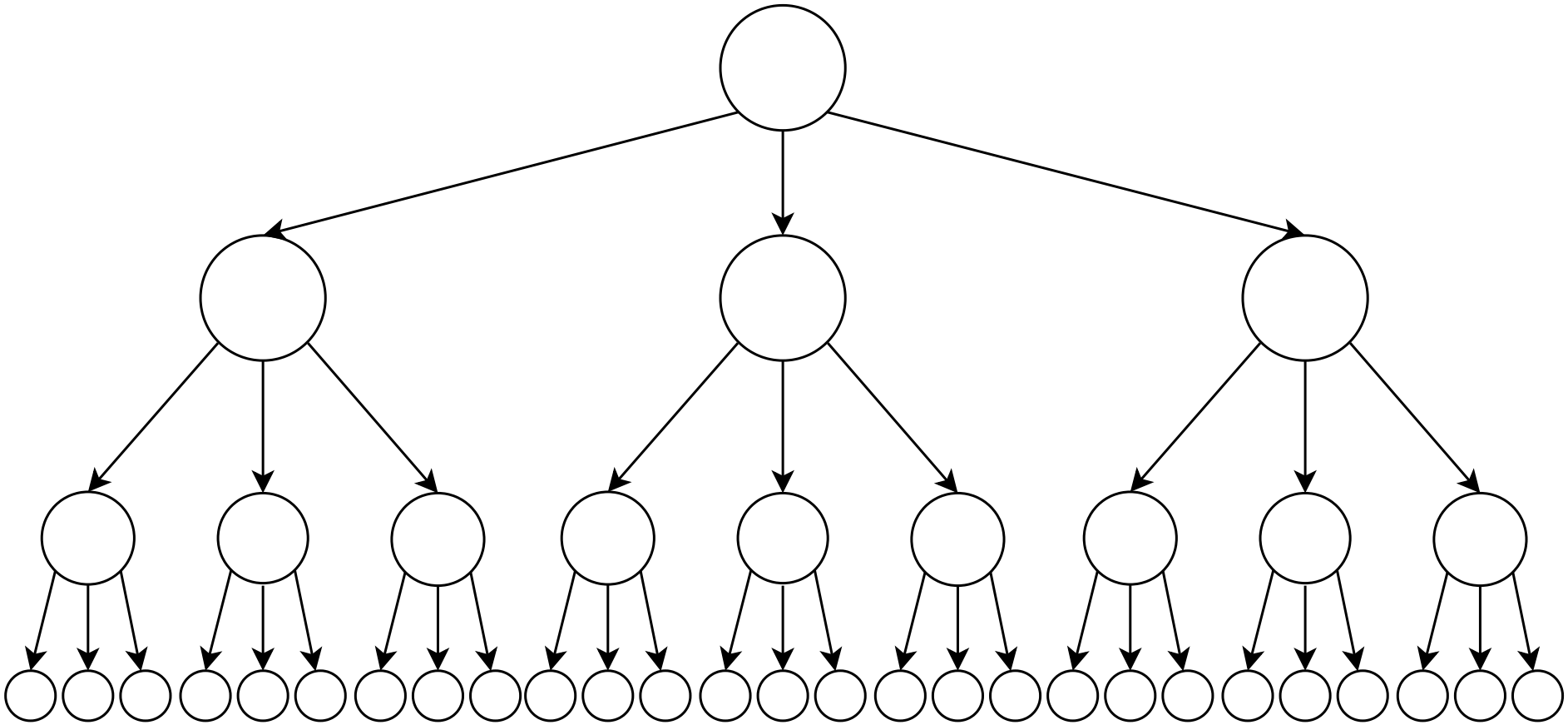
Algoritmos y Estructuras de Datos III
Clase práctica

Tomás Delgado

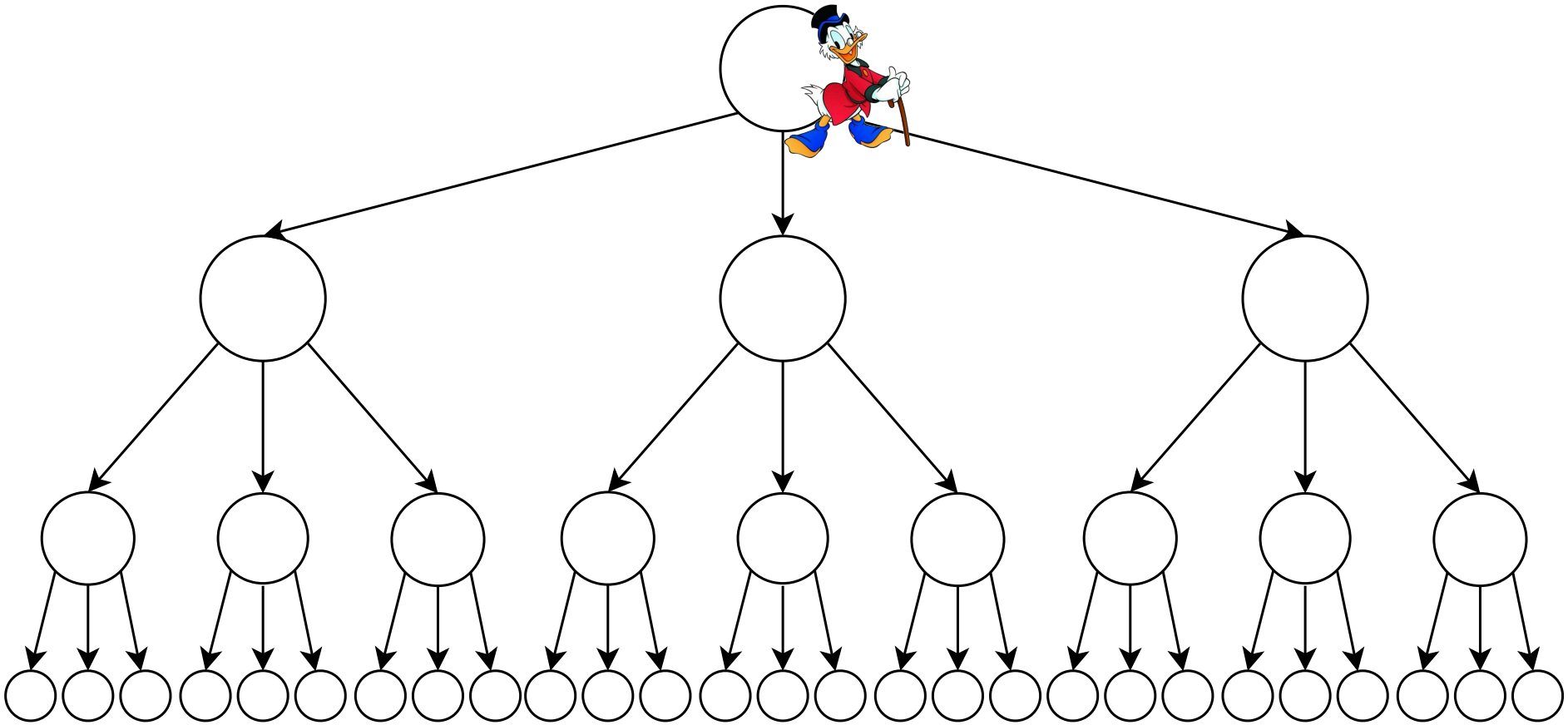
6 de abril de 2023

¿Qué es un algoritmo goloso (o *greedy*)?

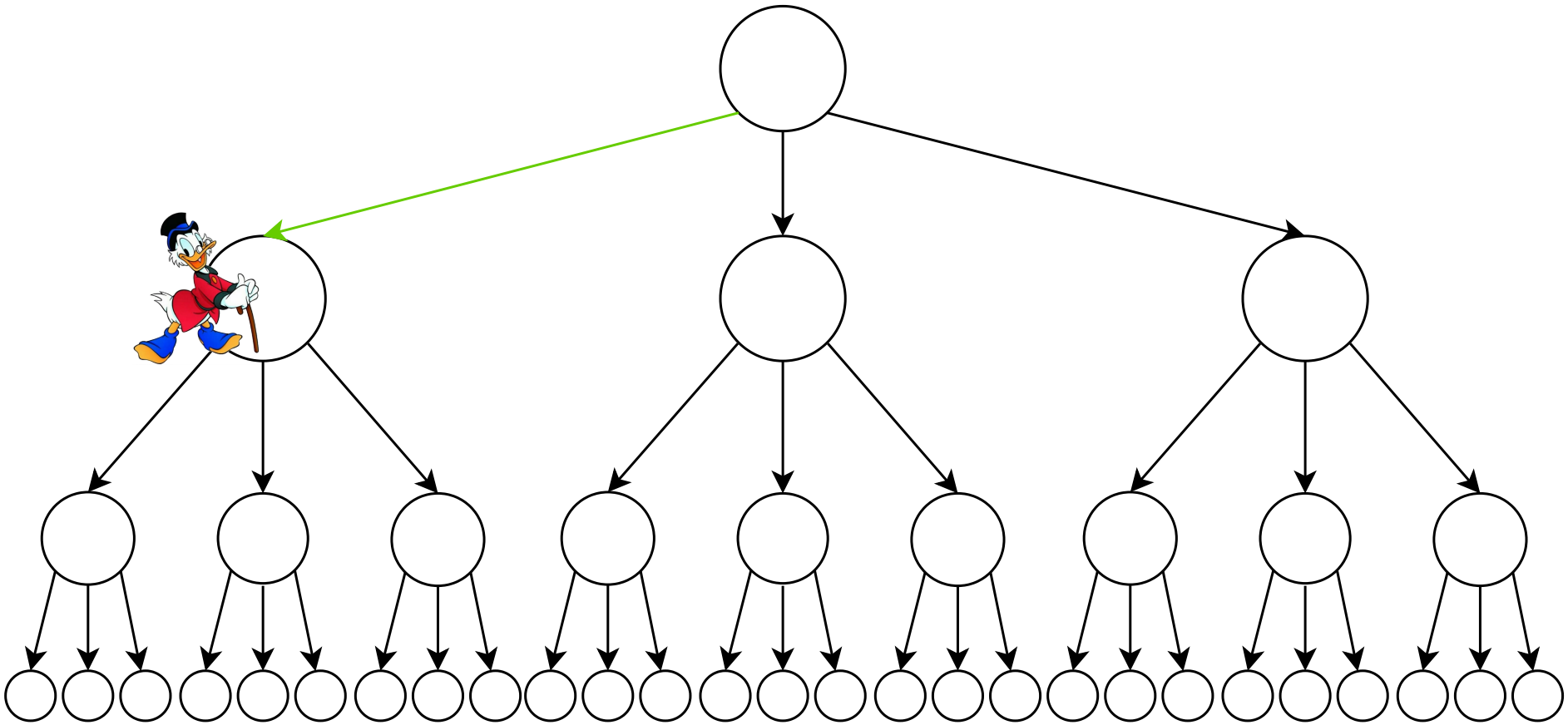
Backtracking (otra vez)



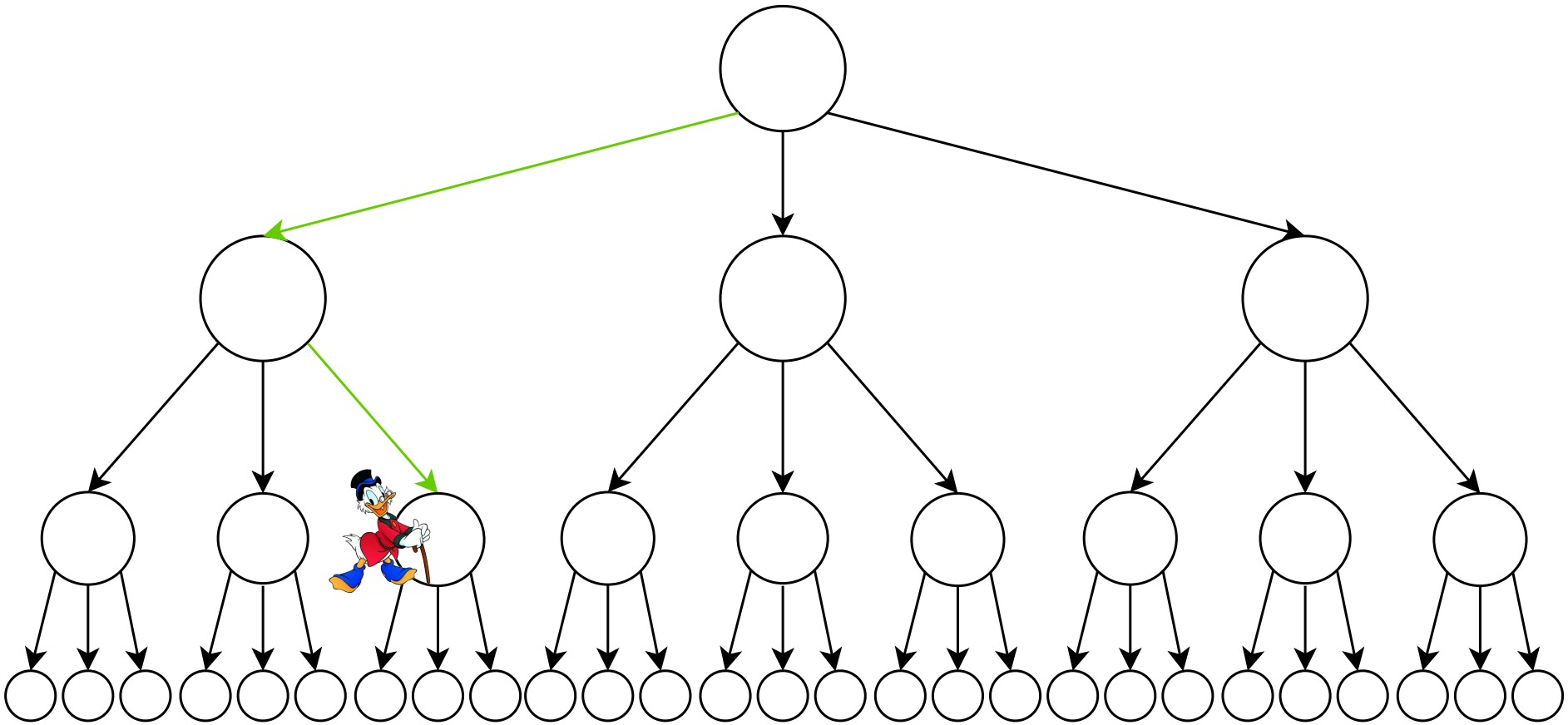
Pero lo resolvemos golosamente



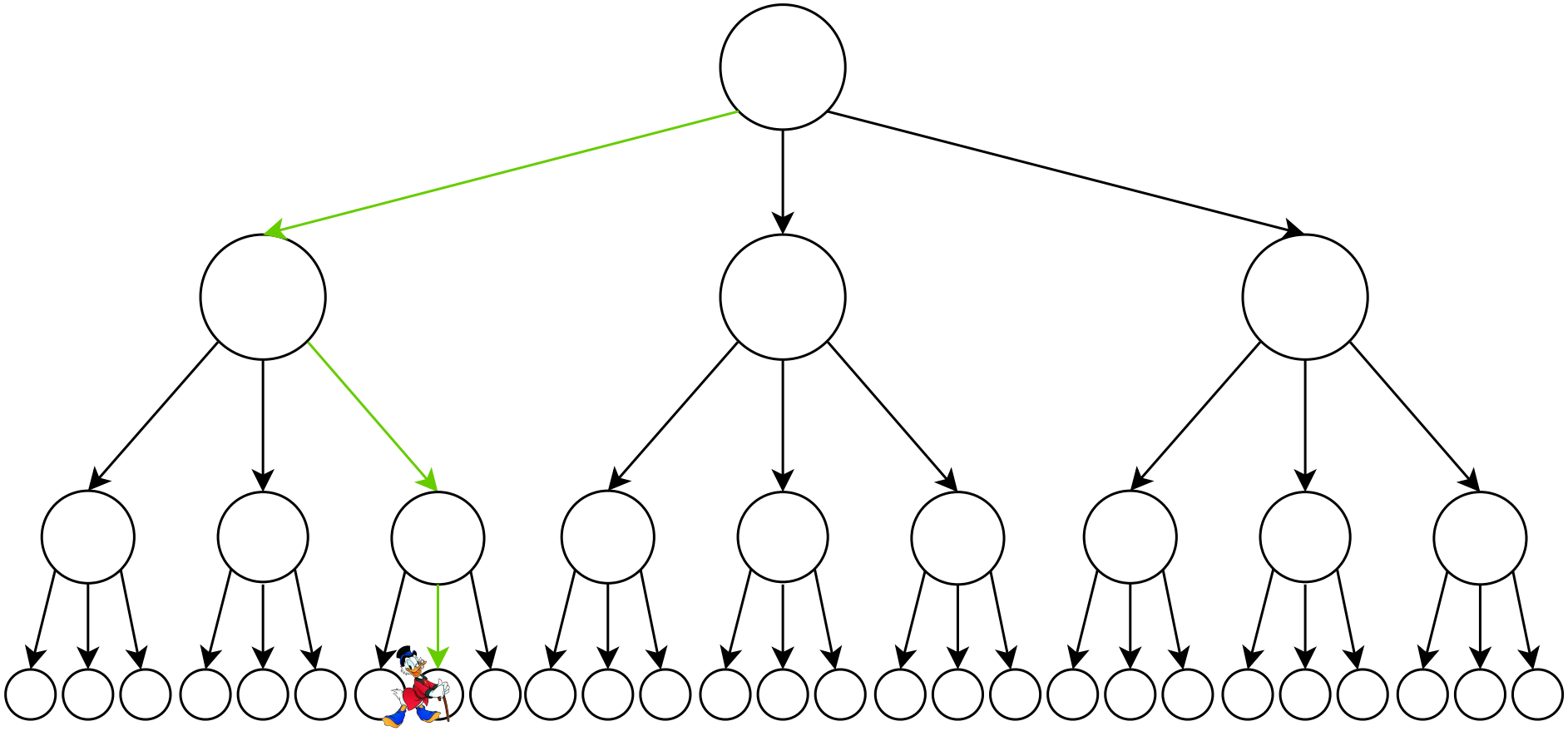
Pero lo resolvemos golosamente



Pero lo resolvemos golosamente



Pero lo resolvemos golosamente



Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión?

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión? Tratemos de que no pase.

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión? Tratemos de que no pase.
- Más en general, un greedy va a ser correcto si siempre llega a una solución.

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión? Tratemos de que no pase.
- Más en general, un greedy va a ser correcto si siempre llega a una solución.
- Para algunos problemas existen soluciones greedy correctas, y para otros no (ej. hace falta BT o DP).

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión? Tratemos de que no pase.
- Más en general, un greedy va a ser correcto si siempre llega a una solución.
- Para algunos problemas existen soluciones greedy correctas, y para otros no (ej. hace falta BT o DP).
- Si no es correcto, igualmente quizás sirva como heurística.

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión? Tratemos de que no pase.
- Más en general, un greedy va a ser correcto si siempre llega a una solución.
- Para algunos problemas existen soluciones greedy correctas, y para otros no (ej. hace falta BT o DP).
- Si no es correcto, igualmente quizás sirva como heurística.
- Suelen ser fáciles de implementar.

Idea general

- En un algoritmo greedy, construimos una solución tomando decisiones locales, sin analizar alternativas.
- ¿Qué pasa si nos equivocamos en una decisión? Tratemos de que no pase.
- Más en general, un greedy va a ser correcto si siempre llega a una solución.
- Para algunos problemas existen soluciones greedy correctas, y para otros no (ej. hace falta BT o DP).
- Si no es correcto, igualmente quizás sirva como heurística.
- Suelen ser fáciles de implementar.
- Lo difícil es convencernos de que funcionan, y en particular demostrarlo.

Ejercicio 1: Representantes ordenados

Problema

Dada una lista de grupos de trabajos prácticos $g_1, g_2, \ldots, g_n$, queremos saber si es posible elegir un representante de cada grupo de tal forma que nos queden ordenados alfabéticamente.

Ejercicio 1: Representantes ordenados

Problema

Dada una lista de grupos de trabajos prácticos $g_1, g_2, \ldots, g_n$, queremos saber si es posible elegir un representante de cada grupo de tal forma que nos queden ordenados alfabéticamente.

Ejemplo 1

[[Andrea, Juan], [Pedro, Juana, Roberta], [Sandro]
Rta: Se puede, por ejemplo tomando Andrea, Juana, Sandro.

Ejemplo 2

[[Andrea, Juan], [Pedro, Juana, Roberta], [Celia]]
Rta: No se puede.

Definir subproblemas

Definir subproblemas

Propuesta:

- $S_{i,s}$ = "decidir si existe una solución para los grupos $g_i, \ldots, g_n$ tal que el primer representante es mayor a s".
- el problema completo sería $S_{1,}$""

Definir subproblemas

Propuesta:

- $S_{i,s}$ = "decidir si existe una solución para los grupos $g_i, \ldots, g_n$ tal que el primer representante es mayor a s".
- el problema completo sería $S_{1,""}$
- Para resolver $S_{i,s}$ elegimos un representante r_i para g_i y pasamos a resolver S_{i+1,r_i}.

Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?

Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?
- Sí! Tomamos el más chico posible (alfabéticamente).

Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?
- Sí! Tomamos el más chico posible (alfabéticamente).
- $r_i = min\{x \in g_i : r_{i-1} < x\}$ ($r_0 = ""$)

Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?
- Sí! Tomamos el más chico posible (alfabéticamente).
- $r_i = min\{x \in g_i : r_{i-1} < x\}$ ($r_0 = ""$)
- Y podemos llamar $R_i = r_1, ..., r_i$ a nuestras soluciones parciales.

Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?
- Sí! Tomamos el más chico posible (alfabéticamente).
- $r_i = min\{x \in g_i : r_{i-1} < x\}$ ($r_0 = ""$)
- Y podemos llamar $R_i = r_1, ..., r_i$ a nuestras soluciones parciales.
- Existen siempre?

Demostración de correctitud

Funciona siempre esta estrategia?

Demostración de correctitud

Funciona siempre esta estrategia?

Podrías hacerlo mejor...

Demostración de correctitud

Funciona siempre esta estrategia?

Podrías hacerlo mejor...

Teorema

La estrategia greedy R_n va a estar bien definida (siempre encuentra un representante) si y solo si existe una solución para los grupos dados.

Queremos ver que si R_n está bien definida, existe una solución.

$\Rightarrow$)

Queremos ver que si R_n está bien definida, existe una solución.
Por hipótesis sabemos que existen $r_1, \ldots, r_n$, que por definición son $r_1 < \ldots < r_n$ y $r_i \in g_i$. Listo, esa es una solución.

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$\Leftarrow)$

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$$P(i) = \text{Si existe solución e } i \leq n, \text{ entonces } R_i \text{ está bien definida}$$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto, así que R_1 está bien definida.

$\Leftarrow)$

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$$P(i) = \text{Si existe solución e } i \leq n, \text{ entonces } R_i \text{ está bien definida}$$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto, así que R_1 está bien definida.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i+1 \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.

$\Leftarrow)$

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$$P(i) = \text{Si existe solución e } i \leq n, \text{ entonces } R_i \text{ está bien definida}$$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto, así que R_1 está bien definida.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i+1 \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto.

$\Leftarrow)$

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$$P(i) = \text{Si existe solución e } i \leq n, \text{ entonces } R_i \text{ está bien definida}$$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto, así que R_1 está bien definida.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i+1 \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto. s_{i+1} está porque $s_{i+1} > s_i$ y $s_i \geq r_i$.

$\Leftarrow)$

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$$P(i) = \text{Si existe solución e } i \leq n, \text{ entonces } R_i \text{ está bien definida}$$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto, así que R_1 está bien definida.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i+1 \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto. s_{i+1} está porque $s_{i+1} > s_i$ y $s_i \geq r_i$.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.
Veamos por inducción que las R_i van a estar bien definidas.

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto, así que R_1 está bien definida. Y además $r_1 \leq s_1$ por ser el mínimo.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto. s_{i+1} está porque $s_{i+1} > s_i$ y $s_i \geq r_i$ por HI.
Además $r_{i+1} \leq s_{i+1}$ por ser el mínimo del conjunto.

Implementación

```cpp
bool representantes(vector<vector<string> >& grupos){
    string anterior = "";
    for(vector<string>& grupo : grupos){

        string r = "-1";
        for(const string& e : grupo){
            if(e > anterior) {
                if(r == "-1") {
                    r = e;
                } else {
                    r = min(r, e);
                }
            }
        }
        if(r == "-1") return false;

        anterior = r;
    }
    return true;
}
```

- Complejidad?

Implementación

```
bool representantes(vector<vector<string> >& grupos){
    string anterior = "";
    for(vector<string>& grupo : grupos){

        string r = "-1";
        for(const string& e : grupo){
            if(e > anterior) {
                if(r == "-1") {
                    r = e;
                } else {
                    r = min(r, e);
                }
            }
        }
        if(r == "-1") return false;

        anterior = r;
    }
    return true;
}
```

- Complejidad? $O(n)$ con n la cantidad de estudiantes.

Versión recursiva

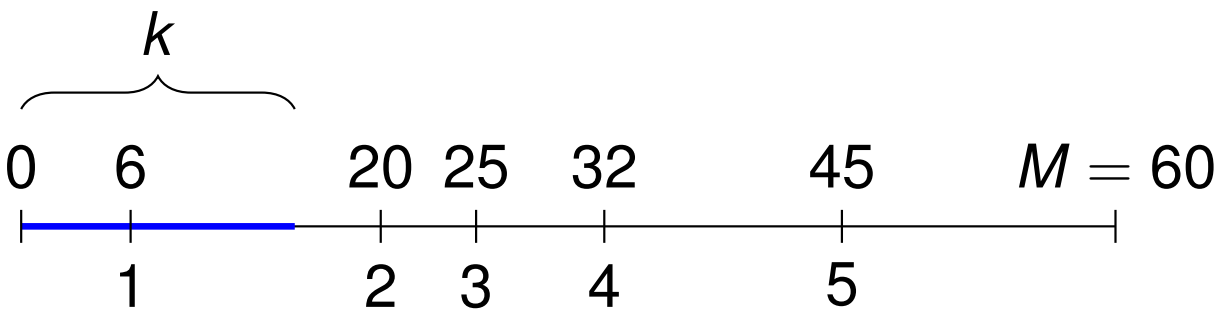
```cpp
bool representantesBT(int i, const string& anterior, vector<vector<string> >& grupos){
    if(i == grupos.size()) return true;

    for(string& e : grupos[i]){
        if(e > anterior && representantesBT(i+1, e, grupos)) return true;
    }

    return false;
}
```

```cpp
bool representantesGreedyRec(int i, const string& anterior, vector<vector<string> >& grupos){
    if(i == grupos.size()) return true;

    string r = "-1";
    for(const string& e : grupos[i]){
        if(e > anterior) {
            if(r == "-1") {
                r = e;
            } else {
                r = min(r, e);
            }
        }
    }

    if(r == "-1") return false;

    return representantesGreedyRec(i+1, r, grupos);
}
```

Ejercicio 2: Dale que quiero llegar

Enunciado

Estamos viajando en auto desde el km 0 al km M de una ruta. En el camino hay n estaciones en los kms $x_1 < \ldots < x_n$. Con un tanque lleno podemos hacer hasta k kilómetros. Suponiendo que siempre que paramos llenamos el tanque, y que la distancia entre dos estaciones contiguas nunca es mayor a k. ¿En qué estaciones de servicio nos conviene parar para minimizar la cantidad de paradas?



Ejercicio 2: Subproblemas

Ejercicio 2: Subproblemas

Subproblemas

$S_{i,c}$ = "Arranco en la estación i con c nafta en el tanque."

El problema completo es $S_{0,k}$

Ejercicio 2: Elección greedy

Ejercicio 2: Elección greedy

Estrategia:

- Paramos solo cuando no nos alcanza la nafta para llegar a la estación siguiente.
- Es decir, en $S_{i,c}$ cargamos nafta en i sii $x_{i+1} - x_i > c$

Ejercicio 2: Demostración

Tenemos que ver dos cosas:

1. La solución que generamos es válida.
2. No existen soluciones con menos paradas.

Ejercicio 2: Demostración (Validez)

(1) La solución que generamos es válida:

Ejercicio 2: Demostración (Validez)

(1) La solución que generamos es válida:

- Veamos que nunca nos quedamos sin nafta.

Ejercicio 2: Demostración (Validez)

(1) La solución que generamos es válida:

- Veamos que nunca nos quedamos sin nafta.
- A la primera estación llegamos porque arrancamos con el tanque lleno y $x_1 \leq k$.

Ejercicio 2: Demostración (Validez)

(1) La solución que generamos es válida:

- Veamos que nunca nos quedamos sin nafta.
- A la primera estación llegamos porque arrancamos con el tanque lleno y $x_1 \leq k$.
- Si llegamos a una estación, llegamos a la siguiente porque si no tuviéramos suficiente nafta cargaríamos (y $x_{i+1} - x_i \leq k$).

Ejercicio 2: Demostración (Optimalidad)

(2) No existen soluciones con menos paradas:

Ejercicio 2: Demostración (Optimalidad)

(2) No existen soluciones con menos paradas:

- Veamos que en cada paso i nuestra solución con paradas $e_1 \dots e_r$ hasta i se puede extender a una solución óptima $e_1 \dots e_r e_{r+1} \dots e_s$ (con $e_{r+1} \geq x_i$).

Ejercicio 2: Demostración (Optimalidad)

(2) No existen soluciones con menos paradas:

- Veamos que en cada paso i nuestra solución con paradas $e_1 \dots e_r$ hasta i se puede extender a una solución óptima $e_1 \dots e_r e_{r+1} \dots e_s$ (con $e_{r+1} \geq x_i$).
- Por inducción en i.

Ejercicio 2: Demostración (Optimalidad)

(2) No existen soluciones con menos paradas:

- Veamos que en cada paso i nuestra solución con paradas $e_1 \dots e_r$ hasta i se puede extender a una solución óptima $e_1 \dots e_r e_{r+1} \dots e_s$ (con $e_{r+1} \geq x_i$).
- Por inducción en i.
- $i = 0$: Todavía no paramos así que basta extender la secuencia vacía a una solución óptima cualquiera.

Ejercicio 2: Demostración (Optimalidad)

Paso inductivo ($i > 0$).

- Supongamos que nuestra solución para i se podía extender. Es decir, habíamos parado en $e_1 \dots e_r$ antes de llegar a i y existía una extensión $e_{r+1} \dots e_s$.

Ejercicio 2: Demostración (Optimalidad)

Paso inductivo ($i > 0$).

- Supongamos que nuestra solución para i se podía extender. Es decir, habíamos parado en $e_1 \dots e_r$ antes de llegar a i y existía una extensión $e_{r+1} \dots e_s$.
- Veamos que en el paso $i + 1$, luego de la decisión tomada en x_i, esto sigue siendo posible.

Ejercicio 2: Demostración (Optimalidad)

Paso inductivo ($i > 0$).

- Supongamos que nuestra solución para i se podía extender. Es decir, habíamos parado en $e_1 \dots e_r$ antes de llegar a i y existía una extensión $e_{r+1} \dots e_s$.
- Veamos que en el paso $i + 1$, luego de la decisión tomada en x_i, esto sigue siendo posible.
- Tenemos 4 casos, según si nuestro algoritmo para en x_i y según si $e_{r+1} = x_{i+1}$.

Ejercicio 2: Demostración (optimalidad)

Caso 1: Nuestro greedy no para en x_i y la extensión tampoco.

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.

Ejercicio 2: Demostración (optimalidad)

Caso 1: Nuestro greedy no para en x_i y la extensión tampoco.

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Se puede extender con $e_{r+1} \dots e_s$ por HI.

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i + 1$ es $e_1 \dots e_r$.

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Si bien es cierto que $e_1 \dots e_r$ se puede extender con $e_{r+1} \dots e_s$, la HI que queremos probar nos pide que $e_{r+1} \geq x_{i+1}$ (estamos probando $P(i+1)$).

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Si bien es cierto que $e_1 \dots e_r$ se puede extender con $e_{r+1} \dots e_s$, la HI que queremos probar nos pide que $e_{r+1} \geq x_{i+1}$ (estamos probando $P(i+1)$).
- Como nuestro greedy no paró, entonces $x_{i+1} - e_r \leq k$.

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Si bien es cierto que $e_1 \dots e_r$ se puede extender con $e_{r+1} \dots e_s$, la HI que queremos probar nos pide que $e_{r+1} \geq x_{i+1}$ (estamos probando $P(i+1)$).
- Como nuestro greedy no paró, entonces $x_{i+1} - e_r \leq k$.
- Si $e_{r+2} = x_{i+1}$, entonces la extensión $e_1 \dots e_r x_{i+1} e_{r+3} \dots e_n$ es válida y óptima. Absurdo porque es más chica que la que por HI es óptima.

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Si bien es cierto que $e_1 \dots e_r$ se puede extender con $e_{r+1} \dots e_s$, la HI que queremos probar nos pide que $e_{r+1} \geq x_{i+1}$ (estamos probando $P(i+1)$).
- Como nuestro greedy no paró, entonces $x_{i+1} - e_r \leq k$.
- Si $e_{r+2} = x_{i+1}$, entonces la extensión $e_1 \dots e_r x_{i+1} e_{r+3} \dots e_n$ es válida y óptima. Absurdo porque es más chica que la que por HI es óptima.
- Caso contrario, como $e_{r+2} > e_{r+1} = x_i$, debe ser $e_{r+2} > x_{i+1}$ porque x_{i+1} es la inmediata siguiente a x_i.

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Si bien es cierto que $e_1 \dots e_r$ se puede extender con $e_{r+1} \dots e_s$, la HI que queremos probar nos pide que $e_{r+1} \geq x_{i+1}$ (estamos probando $P(i+1)$).
- Como nuestro greedy no paró, entonces $x_{i+1} - e_r \leq k$.
- Si $e_{r+2} = x_{i+1}$, entonces la extensión $e_1 \dots e_r x_{i+1} e_{r+3} \dots e_n$ es válida y óptima. Absurdo porque es más chica que la que por HI es óptima.
- Caso contrario, como $e_{r+2} > e_{r+1} = x_i$, debe ser $e_{r+2} > x_{i+1}$ porque x_{i+1} es la inmediata siguiente a x_i.
- Luego $e_1 \dots e_r x_{i+1} e_{r+2} \dots e_s$ válida y óptima.

Ejercicio 2: Demostración (optimalidad)

Caso 2: Nuestro greedy no para en x_i y la extensión sí ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r$.
- Si bien es cierto que $e_1 \dots e_r$ se puede extender con $e_{r+1} \dots e_s$, la HI que queremos probar nos pide que $e_{r+1} \geq x_{i+1}$ (estamos probando $P(i+1)$).
- Como nuestro greedy no paró, entonces $x_{i+1} - e_r \leq k$.
- Si $e_{r+2} = x_{i+1}$, entonces la extensión $e_1 \dots e_r x_{i+1} e_{r+3} \dots e_n$ es válida y óptima. Absurdo porque es más chica que la que por HI es óptima.
- Caso contrario, como $e_{r+2} > e_{r+1} = x_i$, debe ser $e_{r+2} > x_{i+1}$ porque x_{i+1} es la inmediata siguiente a x_i.
- Luego $e_1 \dots e_r x_{i+1} e_{r+2} \dots e_s$ válida y óptima.
- Notar que no nos quedamos sin nafta antes de llegar a e_{r+2} porque $e_{r+2} - x_{i+1} < e_{r+2} - e_{r+1} \leq k$.

Ejercicio 2: Demostración (optimalidad)

Caso 3: Nuestro greedy para en x_i y la extensión también ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i + 1$ es $e_1 \dots e_r x_i$.

Ejercicio 2: Demostración (optimalidad)

Caso 3: Nuestro greedy para en x_i y la extensión también ($e_{r+1} = x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i + 1$ es $e_1 \dots e_r x_i$.
- Como $e_{r+1} = x_i$, entonces la extensión $e_1 \dots e_r x_i e_{r+2} \dots e_s$ es exactamente igual a $e_1 \dots e_s$ y por lo tanto es válida y óptima.

Ejercicio 2: Demostración (optimalidad)

Caso 4: Nuestro greedy para en x_i y la extensión no ($e_{r+1} \neq x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r x_i$.

Ejercicio 2: Demostración (optimalidad)

Caso 4: Nuestro greedy para en x_i y la extensión no ($e_{r+1} \neq x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r x_i$.
- Como por HI $e_{r+1} \geq x_i$, entonces $e_{r+1} > x_i$. Luego $e_{r+1} \geq x_{i+1}$ porque x_{i+1} es la siguiente estación a x_i.

Ejercicio 2: Demostración (optimalidad)

Caso 4: Nuestro greedy para en x_i y la extensión no ($e_{r+1} \neq x_i$).

- En este caso la secuencia de paradas con la que llegamos a $i+1$ es $e_1 \dots e_r x_i$.
- Como por HI $e_{r+1} \geq x_i$, entonces $e_{r+1} > x_i$. Luego $e_{r+1} \geq x_{i+1}$ porque x_{i+1} es la siguiente estación a x_i.
- Como nosotros paramos en x_i, entonces $x_{i+1} - e_r > k$. Luego $e_{r+1} - e_r > k$ y por lo tanto se quedó sin nafta. Absurdo.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.
- Tiene que existir porque $M = b_s > a_s$.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.
- Tiene que existir porque $M = b_s > a_s$.
- El tema es que no cargar en a_{i_0-1} hace que se quede sin nafta.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.
- Tiene que existir porque $M = b_s > a_s$.
- El tema es que no cargar en a_{i_0-1} hace que se quede sin nafta.
- Digamos que la siguiente estación a a_{i_0} está en el km x.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.
- Tiene que existir porque $M = b_s > a_s$.
- El tema es que no cargar en a_{i_0-1} hace que se quede sin nafta.
- Digamos que la siguiente estación a a_{i_0} está en el km x.
- Como nuestro algoritmo cargó en a_{i_0}, debe ser $x - a_{i_0-1} > k$.

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.
- Tiene que existir porque $M = b_s > a_s$.
- El tema es que no cargar en a_{i_0-1} hace que se quede sin nafta.
- Digamos que la siguiente estación a a_{i_0} está en el km x.
- Como nuestro algoritmo cargó en a_{i_0}, debe ser $x - a_{i_0-1} > k$.
- Además, $b_{i_0} \geq x$ y $b_{i_0-1} \leq a_{i_0-1}$

Ejercicio 2: Bonus, por el absurdo, sin inducción.

(2) No existen soluciones mejores:

- Supongamos que nuestro algoritmo para en $a_1 \dots a_r$ y otra estrategia para en $b_1 \dots b_s$ con $s < r$.
- Veamos que se queda sin nafta.
- Tomemos $i_0 = \text{mín}\{i : 1 \leq i \leq n \text{ y } b_i > a_i\}$.
- Tiene que existir porque $M = b_s > a_s$.
- El tema es que no cargar en a_{i_0-1} hace que se quede sin nafta.
- Digamos que la siguiente estación a a_{i_0} está en el km x.
- Como nuestro algoritmo cargó en a_{i_0}, debe ser $x - a_{i_0-1} > k$.
- Además, $b_{i_0} \geq x$ y $b_{i_0-1} \leq a_{i_0-1}$
- Luego, $b_{i_0} - b_{i_0-1} \geq x - a_{i_0-1} > k$, se quedó sin nafta.

Fin!