Problema 1: Electricidad

Amanda Marrero Santos

Loraine Monteagudo García

Carlos Rafael Ortega Lezcano

Universidad de La Habana

2018-2019

Resumen

En este informe se abordan distintos enfoques para la resolución de un problema que busca minimizar el costo total de un conjunto de objetos que va aumentando a medida que avanza el tiempo, donde el incremento de este último no es constante, sino que depende de las distancias entre dichos objetos. Primero, se prueba con un enfoque de fuerza bruta para analizar las características del problema, obteniendo todas las posibles soluciones y retornando de estas la de menor costo, teniendo una complejidad de $O(n * n!)$. Después de percatarnos de varias propiedades interesantes del problema, se desarrolla una optimización del algoritmo anterior generando menos permutaciones a analizar. Luego, se lleva a cabo una solución recursiva en la que se exploran todos los posibles casos tomando 2 posibles desiciones con un costo de $O(n * 2^n)$. Por último, con el próposito de disminuir la complejidad de los algoritmos, se desarrollan varias soluciones greedy que buscan minimizar el costo de los objetos. Después de varios intentos, se llega a la conclusión de que este enfoque puede no ser el indicado para la resolución.

I. Análisis del Problema

Según el enunciado del problema podemos determinar lo siguiente:
Dadas n lámparas los algoritmos recibirán como entrada los siguientes elementos:

- Lista d con las distancias de las lámparas: las lámparas se encuentran en línea recta, así que se fijará una primera lámpara y para el resto se representará la distancia con respecto a ella. Es decir, será:

d =	d_1	d_2	d_3	...	d_{n-1}	d_n

  Donde $d_0 = 0$ y $\forall i \;\; d_{i+1} > d_i$
- Lista c con el consumo de cada lámpara:

c =	c_1	c_2	c_3	...	c_{n-1}	c_n

- Posición inicial en donde se empieza

$$\text{start = i, con } 0 < i \leq n$$

Aclaremos: cada lámpara se identificará con un entero $i = \overline{1, n}$, este representará el orden en que se encuentran ubicadas las lámparas en línea recta, es decir, la primera lámpara será aquella que tiene $i = 1$, y la última $i = n$. Además, d_i representará la distancia de la lámpara i con respecto a la lámpara 1 y c_i el consumo de esta lámpara.

Teniendo en cuenta que por cada movimiento todas las lámparas encendidas consumirán electricidad, se debe generar como salida:

- El orden en que se apagan las lámparas para minimizar el total de consumo.

$$l = \boxed{l_1 \mid l_2 \mid l_3 \mid ... \mid l_{n-1} \mid l_n}$$

Es decir, se deberá retornar una lista $\{l_1, l_2, ..., l_n\}$, donde los l_i, con $i = \overline{1, n}$, representan las posiciones que ocupan las lámparas ordenadas con respecto a la primera lámpara.

Teniendo en cuenta estas nociones previas del problema se realizaron diversas aproximaciones de la solución que se expondrán a continuación

II. Fuerza Bruta

Primero se desarrolló un algoritmo de fuerza bruta para tener una noción de como se comportaban las soluciones, y para irnos familiarizando con el problema.

Como salida se debe dar una ordenación de las lámparas, y por ese orden se determinará cuál es la permutación que minimiza el consumo total. Así que nuestro algoritmo de búsqueda exhaustiva genera todas las posibles permutaciones, luego se comprueba cuál es el consumo que las lámparas tendrán respeto a esa ordenación y como resultado final se retorna aquella que menor consumo total tenga.

Se tuvieron en cuenta varios detalles de implementación, como que las permutaciones deben comenzar por la posición inicial, por lo que se fija el primer elemento de la ordenación.

Para calcular el costo llevamos una variable t que cuenta cuanto tiempo ha transcurrido desde el momento inicial en segundos, y un contador c que lleva el consumo total. Como el movimiento es a una velocidad de $1m/s$, entonces, se sabe que el tiempo transcurrido para ir de una posición d_1 a otra d_2 es la distancia que hay entre estas dos posiciones, es decir, $abs(d_1 - d_2)$. Luego, según la ordenación, la variable t se actualiza de acuerdo a la distancia que existe entre la lámpara l_{i-1} y l_i de la permutación en el momento i. En total $t = \sum_{i=1}^{n} abs(d_{i-1} - d_i)$, donde d_{i-1} y d_i indican las distancias de las lámparas que ocupan la posición $i - 1$ e i respectivamente en la permutación.

Además, una lámpara l_i que consume c_i consumirá esa cantidad de energía por todo el tiempo t_i que estuvo encendida, es decir, tendrá como consumo total $c_i * t_i$, nótese entonces que $t_i = t$, ya que como se mencionó anteriormente, t cuenta el tiempo transcurrido hasta el momento i. Por lo tanto, el consumo total c será:

$$c = \sum_{i=1}^{n} c_i * t = \sum_{i=1}^{n} c_i * abs(d_{i-1} - d_i)$$

I. Complejidad

Calcular el consumo total de las ordenaciones tendrá, por lo tanto, complejidad $O(n)$, donde n es la cantidad de lámparas. La complejidad del algoritmo está determinada por la cantidad de permutaciones que se deben generar, que es $n!$. Por cada una de estas permutaciones se debe calcular el consumo, por lo que la complejidad del algoritmo de fuerza bruta será $n * n!$

II. Correctitud

La correctitud del algoritmo es muy simple ya que este realiza una búsqueda exhaustiva entre todas las posibles soluciones y se comprueba cuál de ellas es la solución quedándonos con el mínimo. Se computarán más permutaciones que no serán solución, pero como se agotan todas las posibilidades, entonces entre ellas estará el óptimo buscado.

III. Fuerza Bruta Mejorada

El problema fundamental del algoritmo anterior es que no está teniendo en cuenta todas las características del problema. Se generan todas las permutaciones sin restricciones, sin considerar que al ir de la lámpara 1 a la 5 se pasa por las lámparas 2, 3 y 4 y no cuesta nada apagarlas; esto siempre va a ser mejor que ir hasta la 5 apagando solo esta. Demostremos esto en el Lema 1.

Lema 1

Sea s una permutación, tal que: $s = \{l_1, l_2, ..., l_n\} : \exists l_i, l_{i+1}$ tal que $l_j - l_i > 0$, entonces existe una permutación s' donde aparece la subsecuencia $\{l_i, l_i + 1, l_i + 2, ..., l_{i+1} - 1, l_{i+1}\}$ que tendrá menor costo que s.

El consumo para todas aquellas lámparas que estén entre l_i y l_{i+1} será lo que se demora en ir de la lámpara l_i a la l_{i+1}:

$$cs_{[l_i, l_{i+1}]} = (d_{l_{i+1}} - d_{l_i}) \sum_{p=l_i}^{l_{i+1}} c_p$$

Donde $cs_{[l_i, l_{i+1}]}$ denota el consumo total de las lámparas que se encuentran en la secuencia de los números entre $[l_i, l_{i+1}]$, d_{l_i} la distancia con respecto al origen de la lámpara que ocupa la posición l_i y c_p el consumo de la lámpara p.

Para el caso de la secuencia s' el costo en el tiempo de ir a l_i a l_{i+1} para todas aquellas lámparas que están entre $[l_i, li + 1]$ será:

$$cs'_{[l_i, l_{i+1}]} = \sum_{p=l_i}^{l_{i+1}} (d_{l_{i+1}} - d_p) c_p$$

La secuencia de distancias es creciente, por lo que: $d_{l_{i+1}} - d_{l_i} > d_{l_{i+1}} - d_p : \quad \forall l_i < p < l_{i+1}$. Luego,

$$cs'_{[l_i, l_{i+1}]} = \sum_{p=l_i}^{l_{i+1}} (d_{l_{i+1}} - d_p) c_p < (d_{l_{i+1}} - d_{l_i}) \sum_{p=l_i}^{l_{i+1}} c_p = cs_{[l_i, l_{i+1}]}$$

Por lo tanto, s' tiene menor costo que s, lo que demuestra el lema.

Una demostración análoga se puede hacer para el caso en que $l_i - l_{i+1} > 1$, es decir:

Lema 2

Sea s una permutación, tal que: $s = \{l_1, l_2, ..., l_n\} : \exists l_i, l_{i+1}$ tal que $l_i - l_{i+1} > 0$, entonces existe una permutación s' donde aparece la subsecuencia $\{l_i, l_i - 1, l_i - 2, ..., l_{i+1} + 1, l_{i+1}\}$ que tendrá menor costo que s:

En este caso la subsecuencia que se formará será decreciente, por lo que las distancias entre estos elementos también será decreciente, entonces: $d_{l_i} - d_{l_{i+1}} > d_p - d_{l_{i+1}} : \quad \forall l_i < p < l_{i+1}$. Luego, al igual que en el lema anterior se tendrá que:

$$cs'_{[l_{i+1},l_i]} = \sum_{p=l_{i+1}}^{l_i} (d_p - d_{l_{i+1}})c_p < (d_{l_i} - d_{l_{i+1}}) \sum_{p=l_{i-1}}^{l_{i+1}} c_p = cs_{[l_{i+1},l_i]}$$

Entonces este lema se cumple ya que $cs' < cs$, por lo que la secuencia s' tiene menor costo que s.

Demostremos que el óptimo es una secuencia en la que entre todos los números consecutivos l_{i-1}, l_i están previamente los elementos entre $[l_{i-1}, l_i]$, dependiendo de quien sea mayor, apoyándonos en los lemas anteriores con el siguiente teorema:

Lema 3

Sea $s = \{l_1, l_2, ..., l_{i-1}, l_i, ..., l_n\}$ una permutación óptima, probemos que se cumple que si $|l_{i-1} - l_i| > 1$ entonces en $[l_1, l_{i-2}]$ están todos los números del intervalo $[l_{i-1}, l_i]$, o $[l_i, l_{i-1}]$ acorde al signo de $l_{i-1} - l_i$.

Supongamos que existe $l_k \notin [l_1, l_{i-2}]$ y que está entre l_{i-1} y l_i, usando el **Lema 1** o **Lema 2** acorde al intervalo, es posible obtener s' tal que $c_{s'} < c_s$. Luego, s no era óptimo, resultando que la secuencia óptima cumplirá lo planteado

Por lo tanto, una posible mejora al algoritmo anterior sería generar solo aquellas permutaciones que no tengan estos saltos. Ya que por el **Lema 3**, el óptimo tendrá esta forma.

Para explicar nuestro procedimiento primero debemos aclarar las siguientes notaciones: se denota i como la lámpara que se va a añadir a la permutación, $i + 1$ e $i - 1$ serán, por lo tanto, las lámparas que se encontrarán a la derecha y a la izquierda de la lámpara i; *last* será la última lámpara que se añadió a la ordenación y $mark_j$ determinará si la lámpara j se añadió o no a la ordenación. De manera tal que si $mark_j = True$ entonces esto significa que en el momento de añadir la lámpara i la lámpara j estará encendida.

Teniendo claro estas notaciones en nuestro algoritmo añadimos solo las permutaciones que cumplan con los siguientes casos:

1. $i < last$ y $mark_{i+1}$: En este caso la última lámpara que se añadió estaba después en la enumeración original de las lámparas con respecto a la lámpara actual. Entonces, el movimiento se hará hacia la izquierda, por lo tanto, la lámpara siguiente a ella, es decir, aquella que ocupa la posición $i + 1$ debe de estar encendida para que la i pueda ser añadida a la permutación en el momento actual.

2. $i > last$ y $mark_{i-1}$: En el otro caso, la lámpara anterior está antes en la enumeración con respecto a la lámpara a analizar, por lo que el movimiento se realizará a la derecha, por lo tanto, la lámpara que está a la izquierda de la actual, es decir, aquella que ocupa la posición $i - 1$ debe de estar encendida para que la i pueda ser añadida a la permutación en el momento actual, en otro caso existirá un salto y esto no se considerará una permutación válida.

Luego de obtener las permutaciones generadas comprobamos cuál es el costo de estas secuencias con el mismo algoritmo explicado en el algoritmo anterior, y el óptimo devuelto es el menor.

I. Correctitud del Algoritmo

Para demostrar la correctitud del algoritmo se debe probar que para añadir un elemento a la permutación solo basta con comprobar si el elemento anterior a él o el siguiente estuvo marcado, es decir, ya se añadió a la permutación, para formar una partición en la que todos sus elementos anteriores o siguientes estuvieron en la permutación, dependiendo si se comprobó que el anterior o el siguiente estuvo marcado en el array de marcas $mark_i$. Observemos que por el **Lema 3** el óptimo es una secuencia que cumple con estas características, por lo que entre las permutaciones generadas se encontrará el óptimo, lo que demuestra la correctitud del algoritmo al quedarnos con aquella que tenga menos consumo.

Demostremos que dada una secuencia $l = \{l_1, l_2, ..., l_k\}$ de tamaño k en la que si $l_k - l_{k-1} > 0$ y $mark_{l_k-1} = True$ implica que todos los elementos entre $[l_{k-1}, l_k]$ están en la secuencia o si $l_{k-1} - l_k > 0$ y $mark_{l_k+1} = True$ entonces todos los elementos entre $[l_k, l_{k-1}]$ fueron añadidas anteriormente a la permutación por inducción en k que es la cantidad de pasos de nuestro algoritmo.

Como caso base, empecemos con una secuencia de tamaño 2, solo $mark_s = True$, donde s es la posición inicial. Por lo tanto, solo son válidas para añadir a la permutación $s-1$ y $s+1$ ya que son las únicas posiciones que tienen un elemento adyacente marcado. Si se añade $s-1$, entonces el único número entre $(s-1, s]$ es s, que ya fue añadida a la permutación por lo que la secuencia $\{s, s-1\}$ es válida. En otro caso, si se añade $s+1$ entonces el único número a entre $[s, s+1)$ es nuevamente s, que ya se encontraba en la permutación, por lo que la secuencia $\{s, s+1\}$ también será válida.

Supongamos que en el paso $2 < m < k$ tenemos una secuencia $l = \{l_1, l_2, ..., l_m\}$ de tamaño m en la que si $l_m - l_{m-1} > 0$ y $mark_{l_m-1} = True$ implica que todos los elementos entre $[l_{m-1}, l_m]$ están en la secuencia o si $l_{m-1} - l_m > 0$ y $mark_{l_m+1} = True$ entonces todos los elementos entre $[l_m, l_{m-1}]$ fueron añadidas anteriormente a la permutación. Demostremos que en el paso k se obtiene una secuencia $l = \{l_1, l_2, , ..., l_m, ..., l_k\}$ si $l_k - l_{k+1} > 0$ y $mark_{l_k-1} = True$ entonces todos los elementos entre $[l_{k-1}, l_k]$ están en la secuencia y si $l_{k-1} - l_k > 0$ y $mark_{l_k+1} = True$ entonces todos los elementos entre $[l_k, l_{k-1}]$ fueron añadidas anteriormente a la permutación.

Si $l_k - l_{k-1} > 0$ y $mark_{l_k-1} = True$, entonces el elemento $l_k - 1$ fue añadido anteriormente en un paso $m < k$, además, el elemento l_{k-1} fue también previamente seleccionado, por lo que cumple la hipótesis, y todo elemento entre l_{k-2}, l_{k-1} está en la permutación, lo mismo ocurrirá para los elementos entre $l_{k-2}, l_{k-3}, ..., l_k - 1$, por lo que por transitividad, en la permutación estarán todos los elementos entre $[l_{k-1}, l_k - 1]$. Además al añadir l_k, como este es el sucesor de $l_k - 1$, también estarán todos los elementos entre $[l_{k-1}, l_k]$.

Si $l_k - l_{k-1} > 0$ y $mark_{l_k+1} = True$, entonces al igual que en el caso anterior el elemento $l_k - 1$ fue añadido anteriormente en un paso $m < k$, y lo mismo ocurre con el elemento l_{k+1}, por lo que aplicando la inducción todo elemento entre l_{k-2} y l_{k-1} está en la secuencia, lo mismo ocurrirá aplicando inducción para los elementos entre $l_{k-2}, l_{k-3}, ..., l_k + 1$, por lo que por transitividad, en la permutación estarán todos los elementos entre $[l_k + 1, l_{k-1}]$ y, por lo tanto, estarán todos los elementos entre $[l_k, l_{k-1}]$.

Luego, para ambos casos se cumple lo explicado, por lo que el planteamiento es cierto para toda secuencia de tamaño k por el principio de inducción matemática.

II. Complejidad

Sea desea determinar el número de permutaciones sobre $A = \{0, 1, ..., n\}$ que cumplen:

- Inician en un número s tal que: $s \in [0, n]$

- Se cumple un orden en su formación, según lo enunciado en el **Lema 3**

Lema 4: Sean n objetos iguales y k categorías distintas, la cantidad de formas de distribuir los n objetos entre las k categorías es:

$$\binom{n+k-1}{k-1}$$

Primeramente interpretemos nuestro problema para poder usar el **Lema 4**. En este caso fijemos a partir de s los conjuntos $S = \{0, 1, 2, 3, ..., s-1\}$ y $E = \{s+1, s+2, ..., n-1, n\}$ tal que tenemos s elementos en el primero y $n-s$ elementos en el segundo, para que una permutación sea válida, según lo visto en el **Lema 3**, tenemos que garantizar que todos los elementos están ya colocados para cada paso k, podemos ver como las permutaciones válidas están constituidas por bloques de números de tamaño entre 0 y s del conjunto S intercalados entre los números del conjunto E, en cada distribución es necesario que los números del bloque estén ordenados decrecientemente y que la unión de los bloques de como resultado una secuencia ordenada decreciente, mientras que los números de E se encuentran ordenados de forma ascendente, basado en lo anterior distribuir los números de S entre los números de E es equivalente al **Lema 4**, tomemos los números de S como los objetos iguales y $n-s$ las categorías distintas, donde se cuentan los pares $[s, s+1], ..., [n-1, n]$ y la posibilidad de poner números después de n, resultando en

$$\binom{s-n-s-1}{n-s-1} = \binom{n-1}{n-s-1}$$

Los s objetos iguales que distribuimos ahora se sustituyen por la secuencia $\{s-1, s-2, ..., 1, 0\}$, y se completa la permutación agregando s al inicio, obteniéndose una permutación válida.

IV. Algoritmo de Búsqueda Exhaustiva

El otro método de resolución del problema se basa en la idea de que lo mejor partiendo de una posición es lo mejor entre elegir desplazarse hacia la izquierda o hacia la derecha.

El enfoque utilizado surge teniendo en cuenta que los subproblemas generados al desplazarse hacia la derecha o la izquierda, son de la misma naturaleza que el problema original, aunque con una posición de menos, ya que al tomar una decisión elimino la posición actual, puesto que esta ya no brinda información necesaria para la resolución del nuevo problema.

Es importante destacar los casos en que se alcanza una esquina, donde siempre nos desplazamos en dirección contraria, es decir si llega a la esquina izquierda me desplazo hacia la derecha y si llegue a la esquina derecha me desplazo hacia la izquierda.

El consumo de cada lámpara se va acumulando a través de los llamados recursivos utilizando una idea similar a la que se expone en el algoritmo de Fuerza Bruta; se lleva un acumulador t que cuenta el tiempo transcurrido y se retornará c, que lleva el consumo total de las lámparas. Luego, como ya se explicó anteriormente, se busca cuál es la mejor opción para mejorar c al moverse hacia la izquierda o la derecha, es decir, cuál es el consumo menor dependiendo del camino que se tome. En el caso base, se retornará por lo tanto el consumo que se lleva acumulando.

I. Complejidad

La complejidad de este algoritmo es de la forma $T(n) = 2T(n-1) + O(n)$ que es $O(n * 2^n)$, donde $2T(n-1)$ se debe a la división constante del problema actual en dos nuevos subproblemas

que trabajan con una entrada de un elemento menos; y el $O(n)$ se debe al costo de que cada vez que se elimina un elemento, es necesario reorganizar la lista haciendo un corrimiento de la información hacia una posición anterior a la suya.

II. Correctitud

Este algoritmo se apoya en el **Lema 3** para su correctitud. Es decir, este generará permutaciones que cumplan con las características de las secuencias enunciadas en dicho lema, por lo que la respuesta óptima estará entre las permutaciones generadas. A todas estas ordenaciones se les determinará el costo y se retornará aquel que tenga costo menor, por lo que se asegurará la veracidad de la respuesta.

En este algoritmo cuando se decide ir hacia la derecha o hacia la izquierda es posible llevar un par (i, j) que represente la posición exacta hacia la cual se realizará el movimiento, evitando el hecho de remover elementos de la lista. Con esta técnica se mejora el costo del algoritmo ya que la operación de complejidad n que era requerida en cada paso ya no es necesaria, por lo que la expresión $T(n)$ será $T(n) = 2T(n-1) + O(1)$, obteniéndose una complejidad de $O(2^n)$. Observar que con esta definición el caso base será cuando las posiciones (i, j) son iguales a los extremos de la lista que se recorre.

V. Algoritmos Greedy

Luego de ver los distintos resultados en los algoritmos anteriores, dada la complejidad obtenida con estas soluciones se intentaron varios enfoques greedy.

En todas estas implementaciones, se tuvieron en cuenta los **Lema 1** y **Lema 2**, por lo que si estas generaban una secuencia $\{l_1, l_2, ..., l_n\}$, donde existe l_i tal que $l_i - l_{i+1} < 1$ o $l_{i+1} - l_i > 1$, entonces todas las lámparas que estén encendidas entre $[l_i, l_{i+1}]$ se apagarán. Ya que, recordemos, de manera intuitiva, siempre es mejor apagar las lámparas que se encuentren entre $[l_i, l_{i+1}]$, porque estas se encuentran en el recorrido y no cuesta nada apagarlas.

I. Consumo

Un intento de greedy básico es ir siempre a la lámpara que más consumía y apagarla primero. Este es un enfoque bastante intuitivo ya que se podría pensar que es mejor apagar primero la lámpara que más consume, ya que mientras más tiempo pase encendida el costo total aumentará. Esta lámpara se podría considerar que es prioridad apagarla.

Sin embargo, no fue difícil de encontrar un contraejemplo que evidenciara la falta de correctitud del algoritmo. Este contraejemplo fue comprobado con el algoritmo mejorado de fuerza bruta explicado anteriormente:

start = 5

d =	0	3	17	35	55	67	86	101

c =	7	5	5	19	16	12	4	17

El greedy devolvió lo siguiente:

l =	5	4	3	6	7	2	1	0

Teniendo un consumo total de 5305
Mientras que la respuesta óptima consume 2043 y esta sería:

l =	5	4	2	1	0	6	7

Pensamos que el principal fallo de este enfoque es que se centra solo en el consumo máximo y no toma en cuenta la distancia que se debe recorrer para ir de una lámpara a otra. Mientras mayor sea la distancia, más tiempo transcurrirá y más estarán consumiendo el resto de las lámparas, por lo que se debería tener en cuenta la distancia, y en eso nos apoyamos para desarrollar el otro enfoque greedy.

II. Razón entre consumo y distancia

El otro intento greedy siguió intentando priorizar aquellas lámparas que tuviesen mayor consumo, pero además se intentó multar aquellas que estuviesen a una mayor distancia de la posición actual, por lo que se consideró la razón entre el consumo y la distancia.

Es decir, para todo paso i se calcula cuál es el mejor movimiento próximo, y para ello se considera la razón entre el costo de todas las lámparas encendidas restantes y la distancia que estas ocupan con respecto a la lámpara l_i, que es la posición actual.

Por lo que para toda lámpara l_j encendida se considera la razón:

$$r_j = \frac{c_j}{abs(d_i - d_j)}$$

Y según estos resultados se elige la que tenga mayor razón, es decir, aquella cuyo costo con relación a la distancia sea mayor.

Este enfoque dio muchos mejores resultados que el anterior. Los contraejemplos que evidencian la falta de correctitud del algoritmo fueron en un principio difíciles de encontrar, pero este fue rápidamente declarado erróneo por el probador desarrollado.

Dado este ejemplo:

start = 4

d =	0	7	11	26	31	44

c =	1	12	15	15	2	19

El greedy retornó como resultado con un costo de 2139:

l =	4	3	5	2	1	0

Mientras que el óptimo con un consumo de 1927 fue:

l =	4	3	2	1	5	0

La cercanía entre las dos soluciones nos hizo pensar en la posibilidad de un enfoque similar un poco más elaborado.

III. Razón acumulada

Lo que no tiene en cuenta el algoritmo anterior es la pérdida ocasionada al determinar ir en una dirección y desechar la otra. Ya que, por ejemplo, de decidir ir hacia la derecha se tardará más en ir a todas las lámparas que se encuentran a la izquierda, por lo que se debe tener en cuenta el tiempo de más que estas estarán consumiendo.

Por lo tanto, este enfoque intenta determinar más que las lámparas hacia donde ir, la dirección hacia la cual vale la pena ir.

Para esto se calcula la suma acumulada de las razones entre el consumo y la distancia dada una posición i. Es decir, para todo paso i, se calcula para todo $j > i$:

$$acc_j = acc_{j-1} + \frac{c_j}{abs(d_i - d_j)}$$

Y luego, para todo $i > j$ se calcula:

$$acc_j = acc_{j+1} + \frac{c_j}{abs(d_i - d_j)}$$

Por último, se comprueba cuál valor es mayor, acc_0 o acc_{n-1}, si es el primero entonces se decide ir hacia la izquierda ya que es hacia esa dirección donde está el mayor consumo de energía con respecto a la distancia que se tiene, en otro caso se elige ir hacia la derecha ya que por esta misma razón en esta dirección habrá mayor consumo.

Aunque este resultaba un enfoque prometedor, en el probador programado se encontró un caso en el que se obtiene la respuesta errónea comparada con el algoritmo de fuerza bruta:

Para el siguiente caso:

start = 1

d =	0	4	13	27

c =	8	13	20	8

El algoritmo greedy da como resultado:

l =	1	2	0	3

Con costo 748, pero el óptimo tiene consumo 620 y es:

l =	1	0	2	3

IV. Conclusiones sobre el enfoque Greedy

Después de desarrollar todos los algoritmos Greedy previamente expuestos, llegamos a una conclusión analizando el último ejemplo, observemos un momento las particularidades de este caso.

Después de la posición inicial 1 se decide ir hacia la primera lámpara, es decir, aquella que ocupa la posición 0, a pesar de que esta es una de las lámparas que menos consumen (8). En este

ejemplo, en el óptimo no se elige cual es el mejor paso inmendiato, sino se elige el mejor paso a largo plazo. En el momento actual, este paso no parece ser el mejor y la respuesta óptima de este ejemplo no tiene un enfoque greedy, lo que nos hizo pensar que quizás este enfoque no funcionará para la resolución del ejercicio.