Informe
Matemática Discreta II
Nearest Opposity Parity
Problema:1272E

Abel Molina Sánchez
Grupo 2-11
Ciencias de la Computación
Universidad de La Habana

18 de noviembre de 2020

1. Problema

E. Nearest Opposity Parity

You are given an array a consisting of n intergers. In one move, you can jump from the position $i - a_i$ (if $1 \leq i - a_i$) or to the position $i + a_i$(if $i + a_i \leq n$).
For each position i from 1 to n you want to know the minimum the number of moves required to reach any position j such that a_j has the opposite parity from a_i (i.e. if a_i is odd then a_j as to the be even and vice versa).

Input:
The first line of the input contains one integer n ($1 \leq 2 * 10^5$) - the number of element in a.
The second line of the input contains n integers $a1, a2, ..., an$ ($1 \leq a_i \leq n$), where a_i is the 1-th element of a.

Output:
Print n integers $d1, d2, ...dn$, where d_i is the minimum the number of moves requiered to reach any position j such that a_j has the opposite parity from a_i or -1 if it is impossible to reach such a position.

Interpretación y consideraciones del problema

Considero el arreglo de entrada como un grafo dirigido no ponderado $G(V, E)$, $(u, v) \in E$ par ordenado, donde cada vértice $v \in V[G]$ se caracteriza por cumplir o no la propiedad de ser par, digamos definida por la función $Par(x) : N \to \{0, 1\}$ que indica si es par o no de acuerdo a su congruencia módulo(2); pero en sí, son dos grupos bien definidos por esta propiedad ya que $\forall n \in N$, 2 divide a n o 2 no divide a n. Luego cada vértice tendría como adyacentes, o una arista incidente en ellos, a aquellos que se encuentran en las posiciones $i+ai$ e $i-ai$ siempre que sea posible.

Luego se tendría un grafo $G(V, E)$ dirigido, donde se busca encontrar para cada vértice la mínima distancia hacia otro cualquiera que sea evaluado de forma distinta que él por la propiedad $Par(x)$.

Ideas de solución

Primera idea: conociendo que el algoritmo $BFS(v)$ sobre un grafo G, para un vértice $v \in V[G]$ calcula la mínima distancia desde él hasta el resto de los vértices del grafo, la primera idea de solución pasa por ejecutar el algoritmo de $BFS(v)$ para cada vértice v que forma parte del grafo del problema. Se realizaría comprobando si se encuentra a alguno que sea evaluado contrario a él por la función $Par(x)$ que define la propiedad de ser o no par. Luego, esta solución tendría una complejidad temporal $O(|V|^2 + |V||E|)$ que es mejorable dados los planteamientos del problema.

Idea de la solución presentada:

Considerando los dos grupos definidos de vértices por la función $Par(x)$, llamaremos rojos(r)

y azules(a) a estos dos grupos. La solución pasará por encontrar la menor distancia hasta cada vértice azul partiendo la búsqueda desde los vértices rojos y viceversa. Luego para adaptar la solución a las condiciones iniciales del problema, considero el grafo $G^- = (V, E^-)$, donde por cada par ordenado $(u, v) \in E[G]$, $(v, u) \in E^-[G^-]$ (cambiar la orientación del arco). Entonces cada vértice i va a ser incidido por los vértice $i - a_i$ e $i + a_i$ (siempre que sea posible).

Luego, la idea pasa por agregar un nuevo vértice a G^-, llamémosle blanco(b), conectado a cada vértice rojo de G^-, o sea, existirán las aristas $(b, r_i) \forall r_i \in V[G^-]$(siendo r_i los vértices rojos de G^-). Luego se ejecutaría el algoritmo $BFS()$ partiendo desde b y por cada vértice azul guardar la menor distancia hallada desde b hasta ellos. Luego repetir el proceso cambiando azules por rojos y restando 1 a cada distancia obtenida se tendrá la solución completa del problema.

Definiciones y Demostración:

Definición: Un grafo dirigido es un par $G = (V, E)$ donde V es un conjunto de vértices y E es un conjunto de pares ordenados (u, v).

Definición: En un grafo $G = (V, E)$ dirigido, a los pares ordenados $(u, v) \in E[G]$ se les llama arco.

Definición: En un grafo $G = (V, E)$ dirigido, un vértice $v \in V[G]$ es adyacente a otro vértice $u \in V[G]$ si el arco $(u, v) \in E[G]$.

Definición: En un grafo $G = (V, E)$ dirigido, se dice que un arco incide sobre $v \in V[G]$ cuando v es el segundo vértice del par ordenado que define ul arco (ej: (u,v)).

Definición: En un grafo $G = (V, E)$ dirigido, se dice que v es adyacente a u, si el arco $(u, v) \in E$

Definición: Una lista de adyacencia es una estructura que se utiliza para representar un grafo. La lista de adyacencia para un grafo $G = (V, E)$ dirigido, será una lista donde cada vértice contiene el conjunto de los vértices que son adyacentes a él.

Definición: Se llama camino dirigido en un grafo $G = (V, E)$ dirigido a la secuencia de arcos $A_1A_2A_3..A_n$, tal que el primer vértice de A_{i+i} coincide con el segundo vértice de A_i $\forall i$, $i = 2, 3..., n - 1$. También se puede representar el camino dirigido con la secuencia de vértices ordenadas según los arcos. En este problema simplemente le llamaremos camino al camino dirigido.

Notación: En este problema se denotará como $cd(u, v)$ al camino dirigido de u a v, con $(u, v) \in V[G]$.

Definición: Sea $G = (V, E)$ un grafo dirigido, se define como grafo inverso de G, al grafo dirigido $G^- = (V, E^-)$ tal que $(u, v) \in E[G]$, $(v, u) \in E^-[G^-]$.

Propiedad: $G = (G^-)^-$
Demostración: Directo de la defínción grafo inverso.

Lema 1: Para cada $cd(v,u)$ en G^- existe el $cd(u,v)$ en G y viceversa.
Demostración:
Inducción en la cantidad de vértices del camino.

Caso base: para 2, un camino de 2 vértice es un arco, por definición, si $(u,v) \in E$, $(v,u) \in E^-$.

Asumimos por hipótesis que se cumple para caminos con k-1 vértices.

Sea $v_1...v_{k-1}, v_k$ un $cd(v_1, v_k)$, si quitamos el vértice v_k del grafo, con todos los arcos incidentes en él, nos queda el camino $v_1, v_2...v_{k-1}$ que por hipótesis de inducción cumple que existe $v_{k-1}, ..., v_1$ en G. Luego, agrego el vértice v_k con sus arcos incidentes y se tiene el arco (v_{k-1}, v_k) que por caso base se cumple que $(v_k, v_{k-1}) \in E$, luego existirá el camino $v_k, v_{k-1}, ..., v_1$ en G, luego se cumple para caminos de k vértices.
Luego por la Propiedad 1 y la inducción matématica se demuestra que el Lema 1 se cumple.

Definición: En un grafo $G = (V, E)$ dirigido, se llama distancia de un camino p, de u a v, a la cantidad de arcos presentes en p.

Definición: En un grafo $G = (V, E)$, se llama $\delta(u, v)$a la distancia mínima de u a v en el G. Esto, es, $\delta(u, v)$ seá igual a la distancia de los caminos de distancia mínima de $u \to v$.

Definición: Si en un grafo $G = (V, E)$, no existe un camino p de $u \to v$, se denota como $\delta(u, v) = \infty$.

Notación: Sea p un camino en un grafo $G = (V, E)$, decimos que $d(p)$ es la distancia de p.

Ahora para continuar con la demostración de la solución analicemos la correctitud del algoritmo BFS:

```
    BFS(s)
1       por cada vértice v en V[G]:
2            d[v] = INF
3       d[s] = 0, Q = {}
4       Q.Enqueue(s)
5       mientras !Q.Empty:
6            u = Q.Dequeue
7            por cada vértice v in Ady[u]:
8                 si no se tiene d[v] < INF:
9                      d[v] = d[u]+1
10                     Q.Enqueue(v)
```

`Ady: Lista de adyacencia para el grafo G=(V,E)`

Notación: Sea s el vértice con el que se inicializa el algoritmo BFS, será $d[u]$ la distancia de

s a v computada por el BFS.

Definición: Se dice que v es descubierto cuando v es agregado a la cola.

Definición: Se dice que u es explorado cuando es extraído de la cola y se analizan los vértices adyacentes a él.
Definición: Se dice que u descubre a v, o v es descubierto por u, cuando v es descubierto durante el análisis de la lista de adyacencia de u.

Lema 1(BFS): $\forall$ vértice v descubierto durante la ejecución del algoritmo $BFS(s)$, $d[v] < \infty$

Demostración:
Inducción en la cantidad de iteraciones
Caso base para la iteración inicial, $d[s] = 0 < \infty$ Asumamos por hipótesis que se cumple para todo vértice descubierto en las iteraciónes anteriores a $k+1$.
Sea j un vértice descubierto durante la iteración $k+1$, sea i el vértice que descubre a j, i tiene que haber sido explorado durante la iteración $k+1$, luego i fue descubierto en una iteración anterior a la $k+1$, y por tanto $d[i] = c < \infty$, luego, como i descubre a j, $d[j] = d[i] + 1 < \infty$.

Lema 2(BFS): Todo vértice v alcanzable por s es descubierto una sola vez.

Demostración: Supongo que hay al menos un vértice que es descubierto más de una vez durante la ejecución del algoritmo. Sea x el único o al menos el primero en esa condición. Sea la segunda vez que se descubre x, si se descubre es porque cumple la condición de la línea 8, lo cual es una contradicción con el Lema1(BFS). Por tanto se cumple que cada vértice alcanzable desde s se descubre una sola vez en el algoritmo.

Lema 3(BFS)[1]:
Suponer que durante la ejecución del BFS, la cola contiene los vértices $< v_1, v_2, ..., v_r >$ donde v_i es el primero y v_r el último. Se cumple que $d[v_r] \leq d[v_1]+1$ y que $d[v_i] \leq d[v_{i+1}]$ para $i = 1, 2, ..., r-1$.

Demostración:
Inducción sobre las de operaciones en la cola.
Caso base para cuando solo s está en la cola se cumple $d[s] \leq d[s] + 1$, v_i coincide con v_r.
Hipótesis: si la cola contiene los vértices $v_1, v_2, v_3...v_r$, v_1 el primero, v_r el último, $d[v_r] \leq d[v_1] + 1$ y $d[v_i] \leq d[v_{i+1}]$ para $i = 1, 2, ..., r-1$.

Demostrar que se cumple cuando se realiza la operación de extraer y agregar a la cola vértices.

Si se extrae el primer vértice de la cola, v_1, luego queda v_2 como primero dentro de la cola, por hipótesis se cumple que $d[v_1] \leq d[v_2]$
luego
$d[v_1] + 1 \leq d[v_2] + 1$,

[1]Lema 22.3 Introduction to Algorithms 3rd Edition,p.599

y por hipótesis se tiene que
$d[v_r] \leq d[v_1] + 1$.
Uniendo ambas, se llega a que
$d[v_r] \leq d[v_1] + 1 \leq d[v_2] + 1$,
$d[v_r] \leq d[v_r] + 1$,
luego se cumple tras realizar una extracción del primero de la cola.

Luego, cuando se agrega un vértice v a la cola, éste se convierte en el último, $v_r + 1$. Este vértice tiene que haber sido descubierto por un vértice u que está siendo explorado, por lo cual, u fue extraído de la cola. Por tanto, sea v_1 el primero de la cola luego de la extracción de u, por hipótesis $d[v_1] \geq d[u]$. Como v fue descubierto por u, $d[v_r] = d[u] + 1$.

Entonces se tiene que:
$d[v_{r+1}] = d[v] = d[u] + 1 \leq d[v_1] + 1$
Como u era el primero en la cola antes de su extracción, se tiene que $d[v_r] \leq d[u] + 1$ por hipótesis.
Luego
$d[v_r] \leq d[u] + 1 = d[v] = d[v_{r+1}]$, por tanto,
$d[v_r] \leq d[v_{r+1}]$.
Por lo tanto el Lema se cumple luego de extraer o agregar vértices en la cola.

Lema 4: Si v es descubierto después de u, se cumple que $d[u] \leq d[v]$.

Demotración: por el Lema2(BFS) se garantiza que cada vértice es descubierto una única vez, luego consideramos todos los vértices descubiertos entre u y v durante la ejecución. sean $u, v_1, v_2, ...v_k, v$ por el Lema3(BFS) se tiene que $d[u] \leq d[v_1] \leq d[v_2], ..., d[v_k] \leq d[v]$.

Lema 5: Después de la ejecución del $BFS(s)$, $\forall v \in V[G]$, $d[v] \geq \delta(s, v)$.

Demotración: Después de la ejecución del BFS $d[v]$ representa la distancia de un camino de u a v, luego por definición $\delta, \delta(s, v)$ es la distancia del menor camino de s a v. Por lo tanto $d[v] \geq \delta(s, v)$.

Lema 6: Sea p un camino de distancia mínima de $s \to v$ y la arista u, v forma parte del camino, entonces $p - (u, v)$ es un camino de distancia mínima de $s \to u$.

Demostración: Supongo que no se cumple, luego $\exists q$ un camino de $s \to u$ tal que: $d(q) < d(p-(u, v))$, como p es un camino mínimo de $s \to v$, $d(p) = \delta(s, u)$, luego $d(p - (u, v)) = \delta(s, v) - 1$ (por la arista u,v)
Se tiene que:

$d(q) < \delta(s, v) - 1$, luego
$d(q) + 1 < \delta(s, v)$ (1)
Ahora, agrego (u, v) a q y hay dos casos.
Si $(u, v) \in q$, existe un camino menor que p de $s - v$, que contradice que p sea mínimo.
$(u, v) \notin q$ si lo agrego, sigue siendo, por (1), un camino de $s \to v$ menor que p, lo que contradice que p sea mínimo. Luego $p - (u, v)$ es un camino mínimo de $s \to u$.

Corolario1: Sea p un camino mínimo de $s \to v$ y la arista (u, v), forma parte de p, entonces $\delta(s, v) = \delta(s, u) + 1$.
Demostración: Por el Lema 6, si p es un camino mínimo de $s \to v$, entonces $p - (v, u)$ es un camino mínimo de $s \to u$. Como la distancia del camino mínimo es igual $\delta(s, u)$,luego

$\delta(s, u) = d(p - (u, v))$
$\delta(s, u) = \delta(s, v) - 1$
$\delta(s, v) = \delta(s, u) + 1$

Teorema 1: luego de ejecutado el algoritmo $BFS(s)$, $d[v] = \delta(s, v)\ \forall v \in G$.

Demotración: Supongo que existe al menos un vértice v tal que $d[v] \neq \delta(s, v)$.
Si no existe camino entre s y v en G, entonces v no es descubierto durante la ejecución y por la inicialización se tiene $d[v] = \delta(s, v)$.
Luego analicemos cuando existe un camino entre s y v en G.

Sea v el vértice más cercano tal que se cumple esa condición. Luego, por Lema 5 se tiene que $d[v] > \delta(s, v)$

Sea p un camino mínimo de $s \to v$. Sea (u, v) el último arco de p, $p = (s, s_1, s_2...uv)$. Como p es un camino mínimo de $s \to v$, $d(p) = \delta(s, v)$ y por el Corolario 1, $\delta(s, v) = \delta(s, u) + 1$

Luego como v es el más cercano que cumple la condición $d[v] > \delta(s, v)$, se tiene que $d[u] = \delta(s, u)$. Entonces se tiene:
$d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1$
$d[v] > d[u] + 1$ (2)

Ahora analizando el estado de v en el momento en que u es explorado:
Caso 1: v no ha sido descubierto:

Como v no ha sido descubierto y v es adyacente a u, v será descubierto por u, por lo tanto $d[v] = d[u] + 1$ lo cual es una contradicción con (2).

Caso 2: v fue explorado.

Si v fue explorado, como u está siendo explorado, v tuvo que ser explorado antes que u, por lo cual fue descubierto antes que u y por el Lema 4, $d[v] \leq d[u]$, lo cual es una contradicción con (2).

Caso 3: v fue descubierto pero no explorado (v está en la cola).

Sea w el vértice que descubre a v, luego v tuvo que ser descubierto antes que u fuera explorado, porque u está siendo explorado, si v no hubiera sido descubierto, u descubriría a v, por tanto w fue explorado antes que u sea explorado, por tanto w fue descubierto antes que u fuera descubierto y por Lema 4:
$d[w] \leq d[u]$, sumando 1 a cada miembro

$d[w] + 1 \leq d[u] + 1$.
Como w descubrió a v,
$d[v] = d[w] + 1$, por tanto

$d[v] \leq d[u] + 1$, lo cual es una contradicción con (2).

Luego queda demostrado que luego de la ejecución de BFS(s) $d[v] = \delta(s, v) \forall v \in V[G]$.

Teorema: La solución descrita resuelve correctamente el problema planteado. Sea n_i la solución esperada, luego de terminada la ejecución $n_i = \kappa_i \ \forall i, i = 1, ..., n$, siendo κ_i la solución del algoritmo para i.

Demostración:
Supongamos que $\exists i$, tal que $n_i \neq \kappa_i$, llamémosle u.
Sin pérdida de generalidad digamos que u es par, y que los pares son azules en la solución.
Si no existe un camino desde u hasta un nodo impar, por definición de grafo inverso, Lema 1 y Propiedad 1, no habrá un camino desde un impar hasta u, y por lo tanto no hará ningún vértice rojo a través del cual se alcance el vértice azul que representa u, luego, b solo posee aristas con los vértices rojos, luego no habrá camino desde b a un vértice rojo, por lo que por inicialización y la correctitud del algoritmo de BFS no se hallará un camino.
Analicemos el caso en el que existe un camino entre u y un impar.
Sea v_x el impar tal que el camino de u a él es mínimo, tenemos que su distancia será $\delta(u, v_x)$.
Sea r_x el vértice rojo tal que el camino minínimo p hallado en la ejecución del algoritmo es $p = b, r_x, ..., u$, luego $d(p) = \delta(b, u)$, como se termina restando 1 a la distancia obtenida
$\kappa_i = \delta_i - 1$,
pero por Corolario1 tenemos que
$\delta(b, u) - 1 = \delta(r_x, u) = \kappa_{i=u}$.
Una vez visto esto, hemos asumido que $\delta(u, v_x) \neq \delta(r_x, u)$:

Caso 1: $\delta(u, v_x) > \delta(r_x, u)$
Dado esto, por Lema 1 y Propiedad 1, se puede afirmar que existe un camino de $u \to r_x$,r_x impar en el grafo original tal que $\delta(u, r_x) < \delta(u, v_x)$, lo cual es una contradicción con el hecho de que v_x sea el impar más cercano.
Veamos el otro caso: $\delta(u, v_x) < \delta(r_x, u)$
Si esto se cumple por el Lema 1 y la propiedad 1 se tiene que va a existir un vértice rojo v_x en el grafo de la solución tal que $\delta(v_x, u) < \delta(r_x, u)$, pero esto es, como b tiene aristas que lo unen a cada nodo rojo y por el corolario1:
$\delta(b, u) < \delta(b, u)$, lo cual es una contradicción y también contradice el Teorema de correctitud del algoritmo BFS.

Una vez demostrada la correctitud de la solución, se pueden plantear los detalles de la implementación.
Se utiliza un único métdo BFS para calcular los caminos mínimos de pares a impares y de impares a pares. Para esto, recibe 2 listas con los vértices pares e impares, llamando azules a los que se le busca la distancia mínima. Como el vértice b tiene aristas con cada vértice rojo, al final habría

que restar 1 a cada distancia y en la primera iteración todos los vértices rojos serían agregados a la cola, por tanto, directamente inicializo la lista d de las distancias a los vértices rojos en 0 y los agrego a todos a la cola. Luego tengo directamente el valor de la distancia de los vértices azules, siendo b un vértice 'ficticio' en la implementación.

Complejidad temporal de la solución:
La complejidad temporal va a estar dada por el tiempo de ejecución del $BFS()$. Cada operación de Enqueue y Dequeue sobre la cola[2] se realiza en $O(1)$. A lo sumo se descubren $|V|$ vértices por tanto se realiza a lo sumo $O(|V|)$ operaciones sobre la cola. Luego por cada vértice se revisa a lo sumo una vez su lista de adyacencia, por lo cual se recorre a lo sumo todos los arcos del grafo, pero por las condiciones iniciales hay a lo sumo $2|V|$ arcos, ya que cada i puede ir a $i - a_i$ o $i + a_i$ a lo sumo, por lo que recorrerlos todos será $O(|V|)$, la inicializaciones son $O(|V|)$ para crear el arreglo d. Luego la complejidad temporal del algoritmo será la complejidad de recorrer toda la lista de adyacencia del grafo, por lo que será $O(|V|)$.

[2]Utilizando deque de Python, parte de la bibliografía