Tarea 2 de Programación 3

11 de octubre de 2024

En recuadros con este formato aparecerán aclaraciones que cumplen una función explicativa pero que no eran requeridos como parte de la solución.

Problema recorrido de ida y vuelta

Sea C una cuadrícula de tamaño $n \times n$, compuesta por n filas y n columnas. En esta cuadrícula, se buscan recorridos que conecten distintas celdas, donde los movimientos permitidos son únicamente en direcciones horizontales o verticales.

Cada celda de la cuadrícula puede tener uno de los siguientes tres estados: libre, ocupado o prohibido. Los recorridos solo pueden incluir celdas que no estén en estado prohibido. Las celdas se identifican por sus coordenadas, fila f y columna c, siendo la celda $(1,1)$ la ubicada en la esquina superior izquierda.

Para simplificar el tratamiento de los casos límite, se asume la existencia de filas 0 y $n+1$, así como columnas 0 y $n+1$, cuyas celdas son consideradas automáticamente como prohibidas.

(a) Se debe encontrar un recorrido donde los movimientos horizontales sean solo a la derecha y los verticales solo hacia abajo, desde la celda $(1,1)$ hasta la celda (n,n) y que pase por la mayor cantidad posible de celdas ocupadas o, si a causa de la posición de las celdas prohibidas ese recorrido no existe, indicarlo.

Se debe diseñar un algoritmo que resuelva el problema cuyo tiempo de ejecución sea $O(n^2)$. Para hacer esto, puede comenzar considerando el problema genérico en el que el recorrido empieza en la celda (f,c) y OPT(f,c) representa la máxima cantidad de celdas ocupadas por las que se puede pasar en un recorrido desde (f,c) hasta (n,n).

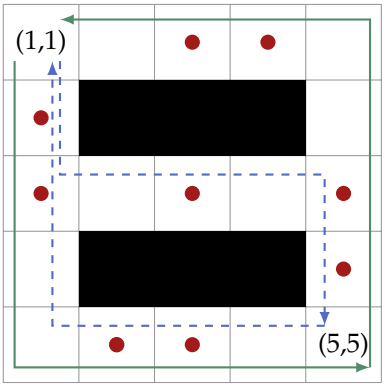
I. Formule una relación de recurrencia para OPT(f,c) y justifique su corrección.

II. Escriba un algoritmo que calcule OPT$(1,1)$. El algoritmo debe ser iterativo y aplicar la relación de recurrencia anterior.

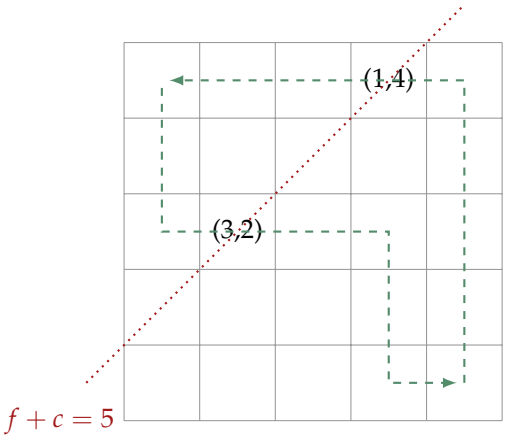
III. Escriba un algoritmo que obtenga el recorrido desde $(1,1)$ hasta (n,n) que pasa por una máxima cantidad de celdas ocupadas moviéndose de la forma indicada más arriba.

(b) En esta parte, el recorrido debe ir desde $(1,1)$ hasta (n,n) como en la parte anterior, pero después se debe volver a $(1,1)$. Al volver, solo se puede mover hacia la izquierda y hacia arriba. El objetivo sigue siendo pasar por la mayor cantidad de celdas ocupadas. Es posible pasar por una misma celda tanto en la ida como en la vuelta, pero en el caso que sea una celda ocupada solo se la cuenta una vez.

Como forma de comenzar a pensar el problema, considere el siguiente algoritmo que **no** lo resuelve correctamente:
Se encuentra un recorrido óptimo de ida desde $(1,1)$ hasta (n,n) como en la parte anterior; luego, se marcan como libres las celdas de ese recorrido. Se encuentra otro recorrido óptimo desde $(1,1)$ hasta (n,n) en el tablero modificado; se lo invierte para que quede como un recorrido de vuelta (desde (n,n) hasta $(1,1)$) y se lo concatena con el primer recorrido.
En la figura se ve mediante un contrajemplo que este algoritmo no siempre encuentra el recorrido óptimo. Las celdas prohibidas están en negro y las ocupadas tienen un punto rojo. El recorrido obtenido por el algoritmo se ve en la línea azul discontinua. El camino de ida, $((1,1)–(3,1)–(3,5)–(5,5))$, pasa por cinco celdas ocupadas, y el de vuelta por otras dos, llegando a un total de siete (se vuelve a pasar por las celdas (2,1) y (3,1) que inicialmente estaban ocupadas, pero no se las vuelve a contar porque ya se pasó por ellas en el primer recorrido). Sin embargo, hay dos recorridos que pasan por ocho celdas ocupadas, uno de los cuales se muestra con línea verde continua (el otro pasa por las mismas celdas pero en el sentido opuesto).



Para seguir pensando en el problema, consideremos la fila y columna genérica f y c, respectivamente. Notemos que dada cualquier diagonal $f + c = d$, $2 \leq d \leq 2n$, un recorrido debe pasar dos veces por ella (puede ser que las dos veces pase por la misma celda).
En la figura se ve un recorrido que pasa por la diagonal $f + c = 5$ en las celdas $(3,2)$ y $(1,4)$.



I. Ahora, consideremos la siguiente generalización del problema en la que las celdas de inicio y final no son la misma, sino que están en diagonales que van de izquierda a derecha y de abajo hacia arriba. Como la suma $f + c$ es constante en las celdas de una diagonal, se trata de las celdas (f, c) y $(f - k, c + k)$, con $0 \leq k \leq \text{mín}\{n - c, f - 1\}$. Note que k puede ser 0, lo que produce el caso particular en que las celdas de inicio y de fin son las mismas.
Entonces, en esta versión general del problema, el recorrido de ida es de (f, c) hasta (n, n), y el de vuelta desde (n, n) hasta $(f - k, c + k)$.
Avanzando en la resolución del problema inicial, piense y explique cuáles son los subproblemas que se deben considerar para resolver el problema genérico con inicio en (f, c) y fin en $(f - k, c + k)$.
Considere que OPT(f, c, k) representa la máxima cantidad de celdas ocupadas por las que se puede pasar con un recorrido de ida desde (f, c) hasta (n, n) y uno de vuelta desde (n, n) hasta $(f - k, c + k)$.
Formule una relación de recurrencia para OPT(f, c, k) y justifique su corrección.

II. Finalmente, resuelva el problema planteado inicialmente, es decir, el recorrido de ida es desde $(1, 1)$ hasta (n, n) y el de vuelta es desde (n, n) hasta $(1, 1)$.
El orden del tiempo de ejecución debe ser polinomial; analícelo y muestre por qué lo es.

Guía

Algo que puede ayudar a entender el problema es compararlo con otro problema conocido, por ejemplo el primero del libro, el de los intervalos. Una instancia de ese problema es un conjunto de n intervalos. Nos ayuda tratar ese conjunto como una secuencia ordenada según los tiempos de finalización. Para resolver el problema consideramos un conjunto, una estructura, de subproblemas, cada uno de los cuales está definido por los primeros i intervalos de la secuencia, para i desde 0 (o 1) hasta n. El problema inicial es un caso particular de este conjunto. Además, nos hacemos una pregunta acerca de la solución tal que cada posible respuesta determine cuál subproblema se debe tener resuelto. Si caracterizamos la solución en orden decreciente de sus índices, como $(s_h, s_{h-1}, \ldots, s_1)$ la pregunta es acerca de s_h, y es si s_h es el intervalo i. Si la respuesta fuera que no, entonces se debe resolver el subproblema de los primeros $i - 1$ intervalos. Si la respuesta fuera que sí, entonces se debe resolver el subproblema de los primeros p_i intervalos (p_i es el último intervalo que no se solapa con i). Los dos subproblemas son los que luego se incluyen en la relación de recurrencia.

El problema de la parte (a) es encontrar un recorrido óptimo desde la celda $(1, 1)$ hasta la celda (n, n). Igual que en el problema de los intervalos, se considera un conjunto de subproblemas, cada uno de los cuales está definido por una celda (f, c) genérica, y consiste en encontrar un recorrido óptimo desde (f, c) hasta la celda (n, n). Aquí también el problema inicial es un caso particular de este conjunto de subproblemas. Esta estructura de problemas es adecuada porque el recorrido óptimo desde $(1, 1)$ hasta (n, n) incluye, para cada una de las celdas (f, c) que lo componen, el recorrido óptimo desde (f, c) hasta (n, n). O sea, incluye la solución del subproblema definido por (f, c). Para hacer la pregunta que determine los subproblemas que se deben tener resueltos, podemos caracterizar la solución como un recorrido que empieza en (f, c) y sigue con

$$(f', c') \to (f'', c'') \to \ldots \to (n, n).$$

Aquí debemos pensar cuál es la pregunta correspondiente a la que en el problema de los intervalos era saber algo acerca de s_h. Cada posible respuesta determina un subproblema que se incluye en la relación de recurrencia.

Después que hayamos resuelto el problema de la parte (a) pasamos al de la parte (b). Ahora el recorrido va desde $(1,1)$ hasta (n,n) y vuelve hasta $(1,1)$. Podríamos, igual que en el problema anterior, considerar que cada subproblema esté definido por una celda (f,c), y ese subproblema sería encontrar un recorrido óptimo que vaya desde (f,c) hasta (n,n) y vuelva a (f,c). Pero esto no es adecuado, porque si R es el recorrido óptimo para el problema $(1,1)$ y (f,c) es una de sus celdas, en general R no contiene el recorrido óptimo del subproblema (f,c) (porque en general R no pasa dos veces por la misma celda). Por eso es que se amplía la estructura de subproblemas. En esta nueva estructura un subproblema está definido por una celda de inicio y una celda de fin. Esta estructura es adecuada porque si R es un recorrido óptimo para el subproblema definido por las celdas (f,c) y $(f-k,c+k)$ e incluye las celdas (f',c') y $(f'-k',c'+k')$, entonces R contiene el reorrrido óptimo entre estas últimas celdas, o sea, la solución del problema definido por las celdas $(f'c')$ y $(f'-k',c'+k')$. El problema inicial, recorrido de ida y vuelta desde $(1,1)$, es un caso particular de este conjunto de subproblemas en el que las celdas de inicio y de fin son la misma, $(1,1)$.

Al problema definido por las celdas (f,c) y $(f-k,c+k)$ que están en una misma diagonal lo denotamos mediante la fila f y la columna c de la celda de inicio, y la distancia k entre las celdas de inicio y fin, (f,c,k). De manera similar a lo que vimos para la parte (a), podemos caracterizar la solución para el problema (f,c,k) como un recorrido que empieza en (f,c), sigue con un camino de ida hasta (n,n) y uno de vuelta,

$$\begin{array}{l} (f_i',c_i') \rightarrow (f_i'',c_i') \rightarrow \ldots (f_i^{(h)},c_i^{(h)}) \searrow \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (n,n) \\ (f_v',c_v') \leftarrow (f_v'',c_v') \leftarrow \ldots (f_v^{(h)},c_v^{(h)}) \swarrow \end{array}$$

y termina en $(f-k,c+k)$.

Hay que notar que en cada paso en el recorrido de ida se incrementa uno, o bien la fila o bien la columna, por lo que se pasa a la diagonal siguiente. Y en el recorrido de vuelta se decrementa uno, o bien la fila o bien la columna, por lo que se pasa a la diagonal anterior. Teniendo esto en cuenta, se puede ver que el par de celdas $(f_i',c_i'),(f_v',c_v')$ pertenecen a la diagonal siguiente a la de $(f,c),(f-k,c+k)$, y que $(f_i'',c_i''),(f_v'',c_v'')$ pertenecen a la diagonal siguiente, y lo mismo se cumple con los siguientes pares.

En el problema de los intervalos nos preguntamos acerca de los posibles valores de s_h y nos interesaban dos. En el problema de la parte (a), ¿nos preguntamos acerca de los posibles valores de (f',c'), y también nos interesan dos? En el problema de la parte (b), teniendo en cuenta que los subproblemas están definidos por dos celdas de la misma diagonal, ¿qué podemos preguntar acerca de la solución cuya respuesta determine un subproblema? Hay varias respuestas, tal vez más de dos, cada una de las cuales determina un subproblema. Es conveniente preguntarse que condición debe cumplirse entre (f,c) y (f_i',c_i'), y entre $(f-k,c+k)$ y (f_v',c_v').

Otras sugerencias

- En la relación de recurrencia es posible que convenga tratar de manera diferente el caso $k=0$, o sea, cuando las celdas de inicio y fi son la misma.
- Los estados de las celdas de la cuadrícula C, la instancia del problema, cuyos valores son libre, ocupado o prohibido, puede convenir modelarlos mediante números, por ejemplo 0, 1 e $-\infty$, respectivamente. Si no es posible realizar un recorrido desde (f,c) hasta $(f-k,c+k)$, de manera consistente con lo anterior se le puede asignar $-\infty$ a $\text{OPT}(f,c,k)$.
- Si se usa la sugerencia de agregar filas y columnas fuera de los bordes y asignarles estado prohibido, entonces no hay recorrido si una de las celdas está fuera de los bordes. Para esos casos, al asignar casos base de OPT alcanza con decir que si (f,c) o $(f-k,c+k)$ están fuera de los bordes entonces $\text{OPT}(f,c,k)$ es $-\infty$ (si se sigue la sugerencia del punto anterior).

Solución:

(a) I. El conjunto de subproblemas (subinstancias) a tratar consiste en encontrar el camino óptimo desde cada celda de la cuadrícula hasta la celda (n, n).

Si existe algún recorrido desde (f, c) hasta (n, n), y (f, c) no es (n, n), ese recorrido tiene un primer paso para el cual hay hasta dos opciones, que son ir hacia la celda $(f + 1, c)$ o hacia la celda $(f, c + 1)$, dependiendo en ambos casos que la celda no sea prohibida. Supongamos que el primer paso de un recorrido óptimo R sea ir hacia la celda $(f + 1, c)$. El resto de ese recorrido, $R_{\downarrow}$, es un recorrido desde $(f + 1, c)$ hasta (n, n) y es, por lo tanto, una solución válida para la subinstancia correspondiente a la celda $(f + 1, c)$, que además debe ser óptimo. Supongamos que no es óptimo, que hay otro recorrido $R'_{\downarrow}$ solución válida de la instancia $(f + 1, c)$ que pasa por más celdas ocupadas que $R_{\downarrow}$. En ese caso, agregando (f, c) al inicio de $R'_{\downarrow}$ se obtiene una solución válida para la instancia (f, c) que pasaría por más celdas ocupadas que R, en contradicción con la optimalidad de R. Conclusiones similares valen para el caso en que el primer paso de R sea hacia la celda $(f, c + 1)$.

Por lo tanto, el valor óptimo para la instancia (f, c) se obtiene de calcular los valores óptimos para las instancias $(f + 1, c)$ y $(f, c + 1)$ y sumarle al mayor de ellos 1 o 0 según que la celda (f, c) esté ocupada o no:

$$\text{OPT}(f, c) = \mathbb{1}(C[n, n] = \text{ocupado}) + \text{máx}\{\text{OPT}(f + 1, c), \text{OPT}(f, c + 1)\},$$

Para (n, n) no están definidos ninguno de los subproblemas, y es el caso base. Para las otras celdas de la fila n y de la columna n no está definido uno de los subproblemas, por lo que vamos a usar la fila $n + 1$ y la columna $n + 1$, a cuyas celdas le asignaremos el valor óptimo que corresponde a las celdas prohibidas.

Si desde (f, c) no hay recorrido es porque la celda es prohibida, o porque no hay recorrido ni desde $(f + 1, c)$ ni desde $(f, c + 1)$. Al valor óptimo para estas celdas le asignamos un número negativo, $-\infty$. Para integrar esto en la relación anterior modificamos el término que corresponde al estado de la celda para incluir el caso en que sea prohibida. Creamos el arreglo bidimensional de valores, V, en el que a cada celda le asignamos 1, 0 o $-\infty$ según que el estado de C sea ocupado, libre o prohibido respectivamente. De este modo, en los dos casos por los que puede no haber recorrido al menos uno de los dos términos de la relación tiene valor $-\infty$, por lo que la suma también es $-\infty$, ya que el valor del otro término es finito.

Entonces, con

$$V[f, c] = \begin{cases} 1 & \text{si } (f, c) \text{ es ocupada,} \\ 0 & \text{si } (f, c) \text{ es libre,} \\ -\infty & \text{si } (f, c) \text{ es prohibida} \end{cases} \tag{1}$$

tenemos los siguientes casos base

$$\text{OPT}(f, c) = \begin{cases} V[f, c] & (f, c) = (n, n), \\ -\infty & f = n + 1 \text{ o } c = n + 1 \end{cases} \tag{2}$$

y el caso general

$$\text{OPT}(f, c) = V[f, c] + \text{máx}\{\text{OPT}(f + 1, c), \text{OPT}(f, c + 1)\}. \tag{3}$$

II.

Si la celda de inicio o de fin son prohibidas entonces no hay un recorrido. En ese caso el algortimo termina asignando $-\infty$ a $\text{OPT}(1, 1)$ para que sea usado por el algoritmo de la figura 2.

```
1  Algoritmo Cantidad máxima
       Entrada: Cuadricula C
2      si (1,1) o (n,n) son celdas prohibidas entonces
3          Asignar −∞ a OPT(1,1)
4          devolver OPT(1,1) y TERMINAR
5      Crear OPT[n+1, n+1]
6      Crear V[n+1, n+1], según (1)
7      Asignar OPT(f,c) mediante (2)
8      para d desde 2n − 1 decrementando hasta 2
9          para cada celda (f,c) tal que f + c = d, 1 ≤ f ≤ n, 1 ≤ c ≤ n
10             Asignar OPT(f,c) mediante (3)
11     devolver OPT(1,1)
```

Figura 1: Cantidad máxima de celdas ocupadas.

Al estructurar el bucle por diagonales se evita tener que excluir la celda (n,n) en cada paso. Además, es más sencilla la modificación que se hará en la parte (b)

III.

```
1  Algoritmo Recorrido
       Entrada: arreglo OPT calculado en el algoritmo de la figura 1
2      si OPT(1,1) es −∞ entonces TERMINAR indicando que no hay recorrido
3      Inicializar el recorrido R vacío
4      Asignar (1,1) a (f,c)
5      mientras (f,c) sea distinto de (n,n)
6          Agregar (f,c) al final de R
7          si OPT(f+1,c) es mayor o igual a OPT(f,c+1) entonces
8              Incrementar f
9          en otro caso
10             Incrementar c
11     Agregar (n,n) al final de recorrido
12     devolver R
```

Figura 2: Recorrido con cantidad máxima de celdas ocupadas.

Si no hay un recorrido desde $(1,1)$ hasta (n,n) el algoritmo termina con tiempo de ejecución $\Theta(1)$.

Veamos que si hay recorrido entonces termina con tiempo $\Theta(n)$. En cada iteración del bucle se incrementa f o c. Como hay un recorrido, al menos uno de $\text{OPT}(f+1,c)$ y $\text{OPT}(f,c+1)$ no es $-\infty$. Por lo tanto ni f ni c pueden llegar a $n+1$, porque en las celdas correspondientes el valor óptimo es $-\infty$ y no pueden ser las celdas en que está el máximo de los dos valores óptimos. Entonces en $2n-2$ iteraciones tanto f como c son iguales a n, con lo que se cumple la condición de terminación del bucle.

El algoritmo de la figura 1 termina con tiempo de ejecución $\Theta(n^2)$ porque, excepto cuando las celdas de inicio o fin sean prohibidas en cuyo caso el orden es $\Theta(1)$, recorre una vez cada celda y las operaciones por cada una de ellas tienen orden $\Theta(1)$.

Entonces, como descomponemos todas las operaciones en solo dos partes, el tiempo de ejecución es el mayor de las dos, $\Theta(n^2)$.

(b)

Un primer intento de generalizar el problema para encontrar la estructura de subinstancias (subproblemas) es considerar que las celdas de inicio y de fin son la misma pero asumir que puede ser cualquiera de la cuadrícula. Veremos que con esto no es suficiente.

Otro intento es una generalización aún mayor que la propuesta, en la que las celdas de inicio puede ser cualquiera de las n^2 celdas, y la de fin también puede ser cualquiera. De este modo se tendrían n^4 subinstancias. Veamos que alcanza con las propuestas.

En el problema inicial en que las celdas de inicio y fin son la misma, la celda $(1,1)$, un recorrido óptimo R puede verse como un paso inicial desde $(1,1)$ hacia una celda c_i, seguido de un recorrido R', y seguido de un paso final desde una celda c_f hacia $(1,1)$. Las celdas c_i y c_f deben ser $(2,1)$ o $(1,2)$, las vecinas de $(1,1)$. El recorrido R' es un recorrido desde c_i hasta c_f. Con razonamientos similares a los de la sección (a)I se llega a que R' tiene que ser óptimo para la subinstancia (c_i, c_f), y que para encontrar R hay que evaluar todos los posibles casos de pares (c_i, c_f). Estos son $((2,1),(2,1))$, $((1,2),(1,2))$, $((2,1),(1,2))$ y $((1,2),(2,1))$. En los dos primeros, las celdas de inicio y fin coinciden, como en el primer intento de generalización. Pero la últimas dos no son cubiertas por esa estructura de instancias. De estas dos alcanza con considerar una, ya que ambas recorren las mismas celdas en sentido contrario.

En todos los casos las celdas de inicio y fin, sean la misma o no, pertencen a la misma diagonal, $f + c = 3$, mientras que la de la instancia inicial pertenece a la diagonal $f + c = 2$. Esto nos lleva a plantear una estructura de subproblemas en la que cada subproblema está definido por un par de celdas de la misma diagonal. Para ello, alcanza con especificar las coordenadas (f, c) de una de las celdas, por ejemplo la que tiene menor valor de columna, y la distancia, k, entre ambas celdas.

I. Como en la parte (a) usamos el arreglo V para modelar la cuadrícula C.

Para cada (f, c, k) genérica, que no corresponda a casos base, hay cuatro opciones para construir un recorrido. La decisión tiene dos componentes, hacia qué celda es el primer paso del recorrido, y desde qué celda es el último paso. Para cada una de esas componentes hay dos opciones, hacia la derecha o hacia abajo, y desde la derecha o desde abajo. Todas las celdas involucradas están en la misma diagonal, por lo que definen un subproblema (con una excepación cuando k es 0, que trataremos más adelante). Además, esa diagonal es más cercana a la celda (n, n).

Para $k > 0$ hay cuatro subinstancias:

$\downarrow\leftarrow$ El recorrido empieza hacia $(f+1, c)$ y termina desde $(f-k, c+k+1)$.
La subinstancia es $I_1 = (f+1, c, k+1)$.

$\downarrow\uparrow$ El recorrido empieza hacia $(f+1, c)$ y termina desde $(f-k+1, c+k)$.
La subinstancia es $I_2 = (f+1, c, k)$.

$\rightarrow\leftarrow$ El recorrido empieza hacia $(f, c+1)$ y termina desde $(f-k, c+k+1)$.
La subinstancia es $I_3 = (f, c+1, k)$.

$\rightarrow\uparrow$ El recorrido empieza hacia $(f, c+1)$ y termina desde $(f-k+1, c+k)$.
La subinstancia es $I_4 = (f, c+1, k-1)$.

Al máximo de los óptimos de estas cuatro subinstancias se le suma los valores $V[f, k]$ y $V[f-k, c+k]$.

$$\text{OPT}(f, c, k) = V[f, c] + V[f-k, c+k] + \max_{I \in \{I_1, \ldots, I_4\}} \{\text{OPT}(I)\}$$

Si $k = 0$ se deben hacer dos modificaciones ya que las dos celdas son la misma. Una es que la cuarta subistancia no puede definirse en términos de los parámetros (f, c, k) que hemos elegido, porque implicaría un valor negativo de k; el valor de esta instancia es igual al de la primera recorrida en sentido opuesto. La otra modificación es que no se debe sumar $V[f-k, c+k]$ porque si fuera una celda ocupada se la estaría contando dos veces.

$$\text{OPT}(f, c, 0) = V[f, c] + \max_{I \in \{I_1, \ldots, I_3\}} \{\text{OPT}(I)\}$$

Reuniendo ambas recurrencias tenemos

$$\text{OPT}(f,c,k) = \begin{cases} V[f,c]+ & \text{máx}_{I\in\{I_1,\ldots,I_3\}}\{\text{OPT}(I)\} \text{ si } k=0, \\ V[f,c]+V[f-k,c+k]+ & \text{máx}_{I\in\{I_1,\ldots,I_4\}}\{\text{OPT}(I)\} \text{ en otro caso.} \end{cases} \quad (4)$$

También definimos la subinstancia con la que se obtiene el óptimo

$$S = \begin{cases} \text{argmax}_{I\in\{I_1,\ldots,I_3\}}\{\text{OPT}(I)\} \text{ si } k=0, \\ \text{argmax}_{I\in\{I_1,\ldots,I_4\}}\{\text{OPT}(I)\} \text{ en otro caso.} \end{cases} \quad (5)$$

Un caso base es $(n,n,0)$, y como en la parte (a), $\text{OPT}(n,n,0)$ es igual a $V[n,n]$.

También son casos base los subproblemas en los que alguna de las coordenadas de (f,c) o $(f-k,c+k)$ está fuera de los bordes, en cuyo caso la celda es prohibida. Para esos casos, entonces, $\text{OPT}(f,c,k)$ es $-\infty$. Estos casos se necesitan cuando

- la celda de inicio es (n,c) y se considera que el primer paso es hacia $(n+1,c)$,
- la celda de inicio es (f,n) y se considera que el primer paso es hacia $(f,n+1)$,
- la celda de fin, $(f-k,c+k)$ es $(f-k,n)$ y se considera que el último paso es desde $(f-k,n+1)$.

Por lo tanto, son casos base los subproblemas en que f, c, o $c+k$ son $n+1$. No hace falta incluir el caso en que la celda de fin sea $(n,c+k)$ y se considera que el último paso es desde $(n+1,c+k)$, porque la celda de fin coinicide con la de inicio, por lo que k es 0, de donde $f-k$ es f, y por el primero de los puntos ya se trata a $f=n+1$ como un caso base.

Entonces los casos base son

$$\text{OPT}(f,c,k) = \begin{cases} V[f,c] & (f,c,k)=(n,n,0), \\ -\infty & f=n+1 \text{ o } c=n+1 \text{ o } c+k=n+1. \end{cases} \quad (6)$$

II.

Cálculo de los máximos

```
1  Algoritmo Cantidad máxima
       Entrada: Cuadricula C
2      si (1,1) o (n,n) son celdas prohibidas entonces
3          Asignar −∞ a OPT(1,1,0)
4          devolver OPT(1,1,0) y TERMINAR
5      Crear OPT[n+1, n+1, n−1]
6      Crear V[n+1, n+1], según (1)
7      Asignar OPT(f,c,k) mediante (6)
8      para d desde 2n−1 decrementando hasta 2
9          para cada (f,c,k) tal que f+c=d, 0≤k≤mín{n−c, f−1}, f≤n, c≥1
10             Asignar OPT(f,c,k) mediante (4)
11     devolver OPT(1,1,0)
```

Figura 3: Cantidad máxima de celdas ocupadas.

El bucle de la línea 9 se puede estructurar con dos bucles anidados

```
1  para f desde mín{n, d−1} decrementando hasta máx{d−n, 1}
2      Sea c = d − f
3      para k desde 0 hasta mín{n−c, f−1}
4          Asignar OPT(f,c,k) mediante (4)
```

El conjunto de subproblemas se recorre por diagonales, como en la parte (a), ya que es la forma natural de definirlos porque para tratar cada subproblema de la diagonal $f+c$ se necesita tener resueltos tres o cuatro subproblemas de la diagonal $f+c+1$.

Un forma incorrecta de recorrer los subproblemas es separarlo en dos etapas, en la primera se trata el caso k igual a 0 y en la segunda los casos k mayor que 0. Esta forma es incorrecta porque cada subproblema $(f,c,0)$ se necesita tener resuelto el subproblema $(f,c,1)$, que todavía no se resolvió.

Cálculo del recorrido óptimo

El algortimo es similar al de la parte (a).

```
1 Algoritmo Recorrido
    Entrada: arreglo OPT calculado en el algoritmo de la figura 3
2   si OPT(1,1,0) es −∞ entonces TERMINAR indicando que no hay recorrido
3   Inicializar recorridos de ida R_i y de vuelta R_v vacíos
4   Asignar (1,1,0) a (f,c,k)
5   mientras (f,c) sea distinto de (n,n)
6     Agregar (f,c) al final de R_i, y (f−k,c+k) al inicio de R_v
7     Asignar (f,c,k) mediante (5)
8   Agregar (n,n) al final de R_i
9   devolver la concatenación de R_i y R_v
```

Figura 4: Recorrido con cantidad máxima de celdas ocupadas.

Análisis de complejidad

El análisis es similar al de la parte (a), pero ahora la estructura OPT tiene $O(n^3)$ celdas. El algoritmo de la figura 3 recorre a lo sumo una vez cada una de esas celdas con operaciones $\Theta(1)$ por cada una, por lo que es $O(n^3)$.

El algoritmo de la figura 4 empieza en la diagonal dada por $f+c=2$ y en cada paso pasa a la diagonal siguiente, cumpliendo con (5). Si hay un recorrido, la fila f y la columna c calculadas en cada paso deben estar en el rango $\{1,\ldots,n\}$, por lo que al llegar a la diagonal definida por $f+c=2n$, se tiene que cumplir $(f,c)=(n,n)$, que es la condición de terminación del bucle. Entonces el algoritmo termina en $2n-1$ pasos, en cada uno de los cuales las operaciones son $\Theta(1)$, por lo que el algoritmo es $\Theta(n)$.

Como tenemos dos algoritmos, el tiempo de ejecución total queda determinado por el mayor de los tiempos de cada uno de ellos, por lo que es $O(n^3)$, y por lo tanto, polinomial.

Se puede ver que la cota $O(n^3)$ es ajustada. Si ℓ es el largo de la diagonal, la cantidad de subinstancias que corresponden a la diagonal es $\binom{\ell}{1}+\binom{\ell}{2}$ (el primer sumando corresponde a cuando inicio y fin son iguales), o sea, $\ell(\ell+1)/2$. Hay una diagonal de largo n y dos diagonales para cada largo desde 2 hasta $2n$, por lo que la cantidad de subproblemas es

$$\begin{aligned}
&\frac{n(n+1)}{2}+2\sum_{l=2}^{n-1}\frac{l(l+1)}{2}\\
=&\frac{n(n+1)}{2}+\sum_{l=2}^{n-1}l+\sum_{l=2}^{n-1}l^2\\
=&\frac{n(n+1)}{2}+\frac{n(n-1)}{2}-1+\frac{(n-1)(2(n-1)-1)(n)}{6}-1
\end{aligned}$$

Y como hay un término $n^3/3$ llegamos a que la complejidad del algoritmo es $\Theta(n^3)$.

Al aplicar la técnica de Programación Dinámica en problemas de optimización se diseñan dos algorimos. En el primero se calculan los valores óptimos para cada subinstancia, recorriendo el espacio de esas subinstancias desde el caso base hasta el caso principal. En el segundo, una vez conocido el valor óptimo de cada subinstancia se contruye la solución, recorriendo en sentido inverso del algoritmo anterior, desde el caso principal hacia el caso base.
En general no es posible construir la solución solo con el primer algoritmo (sería posible si el problema admitiera un algoritmo greedy).

En el actual problema un intento que no obtiene la solución correcta es determinar para cada diagonal cuál es el par de celdas con las que se obtiene el recorrido óptimo, y asumir que ese par de celdas es parte de la solución. En la diagonal $f + c = 9$ de la figura, ese par está formado por las celdas $(5,4)$ y $(4,5)$ $((f,c,k) = (5,4,1))$ con un recorrido de valor 2, uno de los cuales está trazado con línea azul discontinua. Sin embargo el recorrido óptimo tiene valor 4, que se ve en línea verde continua. Las celdas de la diagonal $f + c = 9$ que pertenecen a la solución son $(6,3)$ y $(3,6)$ $((f,c,k) = (6,3,3))$ a pesar de que el recorrido óptimo entre ellas tiene valor 0.