1 de Abril de 2022

Universidad de los Andes

Diseño y Análisis de Algoritmos

Miguel Sandoval 201923157 y Tales Losada

Proyecto final entrega 1

Punto 1: Algoritmo de solución.

El algoritmo escogido para la solución, menosEsfuerzo(torre) es uno basado en la programación dinámica. La entrada de este algoritmo es la información de la torre: número de pisos n, número de cuartos por piso m, la lista de esfuerzos por piso y la lista de portales. Inicialmente se define una matriz con número de filas igual a n y número de columnas igual a m; cada entrada (i,j) de esta matriz representa el esfuerzo que se debe realizar para ir del cuarto (1,1) al cuarto (i,j) donde i es el piso y j el número de cuarto, si no se puede ir hasta un cuarto la entrada en la matriz será infinito. Para el cálculo del mínimo esfuerzo definimos una recursión:

- Caso base: ir de (1, 1) a (1, 1) y esto toma esfuerzo 0;
- Caso recursivo: para el caso recursivo debemos comparar 3 valores diferentes, el esfuerzo para llegar al cuarto anterior (si lo hay) sumado al esfuerzo de moverse en el piso actual, el esfuerzo para llegar al cuarto siguiente (si lo hay) sumado al esfuerzo de moverse en el piso actual, el mínimo esfuerzo de llegar a un cuarto de donde hay un portal cuyo destino es el cuarto actual.

Y estos casos son precisamente los que trata el algoritmo: inicialmente (líneas 21-22) se define el caso base matriz(0, 0) = 0. Después de esto se realiza la recursión (se hace con dos ciclos uno exterior que determina el piso y uno interior que determina el cuarto): inicialmente (línea 24) se define la entrada en la matriz como infinito; posteriormente (líneas 25-37) se busca sobre la lista de portales uno que llegue al cuarto actual y, si existe, se escoge el mínimo entre el valor actual y el valor de llegar al cuarto del que nace el portal, se repite este proceso hasta haber revisado todos

los portales. A continuación (líneas 38-40), se escoge el mínimo entre el valor actual y el valor de llegar al cuarto anterior (si existe); finalmente (líneas 41-44), cuando ya se ha llenado todo el piso con estos valores, se escoge el mínimo entre el valor actual de cada cuarto con el del cuarto siguiente (si existe), esto se hace ya que los portales no pueden ir hacia abajo, y entonces no se podrá volver a un piso por el que ya se pasó. Por último, se retorna la última entrada de la matriz que se desea, es decir matriz(n-1, m-1).

El algoritmo interaccion( ) representa la interacción entre usuario y máquina. Inicialmente se leen todas las líneas del caso que se quiere evaluar. A continuación, se crean todas las diferentes torres siguiendo la entrada de texto con la función crearTorres(texto). Finalmente, con estas torres creadas se evalúa el mínimo esfuerzo posible para llegar al cuarto (n, m) con la función menosEsfuerzo(Torre) y se imprime este valor (si el valor es infinito se imprime “NO EXISTE”).

El algoritmo crearTorres(texto) tiene la función de crear las diferentes torres del caso que se quiere evaluar, su entrada es el texto que ingresa el usuario. Inicialmente (líneas 68-72) transforma cada línea del texto en una lista de enteros; la primera línea es el número de torres que hay en el caso evaluado, y por lo tanto se define una lista donde cada entrada representa un caso (líneas 75-76). A continuación, se empiezan a crear las torres: la primera línea es la cantidad de pisos, la cantidad de cuartos por piso y el número de portales, estos valores se guardan (líneas 84-87); lo siguiente que se debe hacer, en la segunda línea del caso, es guardar la lista de esfuerzos por piso, para hacer esto se define una lista y mientras esta lista no contenga tantos elementos como pisos haya se guarda el valor leído en la lista (líneas 88-90). Se sigue agregando todos los portales que debe haber (pues ya se ha definido el número de portales) a una lista de portales (líneas 91-95). Por último, cuando ya se ha leído toda la información de la torre actual, se guarda la información de la torre en la lista de casos (líneas 97-99).

Punto 2: complejidad de tiempo y almacenamiento

$$P = numero\ de\ portales$$

$$N = numero\ de\ pisos$$

$$M = numero\ de\ cuartos$$

$$C = Cantidad\ de\ casos$$

$$C * (P_1 + P_2 + \cdots P_c) + 2C + 1 = Total$$

$$c_1 * total + c_2 * total + f_1 + c_3(3 + total + c) + c(c_4 + f_2 + c_5)$$

$$f_1 = d_1 * 4 + \text{total} * (2 + \text{n}) + c * (4 + total) + d_2 * c * \big(1 + total * (5 + n + p)\big)$$

$$f_2 = k_1 + N * M * P * k_2$$

$$O(N, M, C, P) = C * C * N * M * P$$

Explicación: tablas en documento adjunto “Punto2”

Una suma simple de los procedimientos llevados a cabo en el algoritmo solución lleva a que el tiempo promedio dependa de la cantidad de casos al cuadrado multiplicado por la cantidad de pisos y cuartos de cada caso, dado que se utiliza un ciclo que revisa el total de líneas ingresadas con un ciclo anidado que procesa cada caso del problema propuesto (líneas 58-63) por lo tanto se lleva a tiempo polinomial.

Punto 3: Respuestas a los escenarios de comprensión de problemas algorítmicos.

ESCENARIO 1: Suponga ahora que los portales son bidireccionales.

(i) ¿Qué nuevos retos presupone este nuevo escenario?

Tocaría replantear en la teoría de hallar el recorrido más corto, ósea el mínimo, pues él podría cambiar en ciertas situaciones del recorrido si un portal se puede acceder desde ambas direcciones.

(ii) ¿Qué cambios le tendría que realizar a su solución para que se adapte a este nuevo escenario?

Una vez creadas las torres, se debería hacer una modificación al algoritmo de menosEsfuerzo(Torre) para que considere los casos en los que la bidireccionalidad de los portales recortaran el camino más corto posible para llegar al cuarto (n, m). Pues en varios casos, pero no todos, al cambiar de un cuarto a otro, su mínimo con bidireccionalidad seria 0, pero con portales unidireccionales seria 1.

ESCENARIO 2: Se le pide ahora calcular el número de rutas que existen.

(i) ¿Qué nuevos retos presupone este nuevo escenario?

Establecer un contador que lleve el numero de rutas posibles de cada piso, cada ves que se mueva por la torre.

(ii) ¿Qué cambios le tendría que realizar a su solución para que se adapte a este nuevo escenario?

En este caso, en el que no hay nodos que se repiten, tocaría añadir un contador que sumara uno en cada comparación de la búsqueda de el recorrido mínimo de un cuarto a otro, en el que sume la ruta más favorable y también todas las demás (en las que no se halla estado), y que continue recorriendo por el camino más largo. Probablemente se requiere hacer un método independiente para más claridad en la búsqueda.

ESCENARIO 3: Se le pide ahora calcular la ruta optima en la cual el estudiante puede visitar todos los cuartos y finaliza en (n,m).

(i) ¿Qué nuevos retos presupone este nuevo escenario?

Visitar todos los cuartos de la torre y terminar en el deseado. Posiblemente un nuevo algoritmo para poder usar los portales a conveniencia de la ubicación de (n,m).

(ii) ¿Qué cambios le tendría que realizar a su solución para que se adapte a este nuevo escenario?

Un algoritmo original que, para empezar, recorra todos los cuartos de la torre y luego calcule la ruta de menor esfuerzo hasta (n,m). Dependiendo de la ubicación de (n,m), se puede hallar una manera mas eficiente mediante el uso de los portales, por ejemplo: si resulta que (n,m) es una salida de un portal, se visitaran todos los demás y luego se devuelve a la entrada del portal, que no toma energía, para salir a (n,m) y finalizar el recorrido.