IIC2133 Estructuras de datos y algoritmos
1° semestre 2019

Solución Ayudantía-Taller Programación Dinámica

Introducción

Para que un problema se pueda resolver usando programación dinámica, es necesario que se cumplan dos condiciones:

- La solución al problema debe poder resolverse usando soluciones de sub problemas del mismo tipo. Esto le da un carácter recursivo, lo que hace que se pueda expresar como una función de recurrencia.
- Las llamadas recursivas a la función de recurrencia deben repetirse para que tenga sentido usar PD ya que estas repeticiones son las que hacen que la recurrencia se haga ineficiente.

Dicho esto, los pasos a seguir para resolver estos problemas son:

1. Encontrar la función de recurrencia que resuelve el problema. Normalmente este es el paso más difícil.
2. Programar esta función de recurrencia literalmente. Este paso es el más fácil pero no logra una solución eficiente.
3. Optimizar la solución anterior con el uso de una tabla en el que se almacenen los valores ya calculados para no repetir los cálculos. Este paso es fácil ya que solo debes identificar las variables que definen un subproblema y agregar una tabla que se acceda usando esas variables.

Ya que el paso 1 es normalmente el más difícil, es lo que daremos de solución en esta guía.

Problemas

1. El cálculo del coeficiente binomial se obtiene de la siguiente fórmula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Sin embargo, esta fórmula es inviable en la práctica ya que los factoriales intermedios pueden ser muy grandes. ¿Cómo podemos calcularlo sino?

Solución: El coeficiente binomial está muy relacionado con el triángulo de Pascal:

$$\binom{0}{0}$$

$$\binom{1}{0} \quad \binom{1}{1}$$

$$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

$$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

$$\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4}$$

$$\binom{5}{0} \quad \binom{5}{1} \quad \binom{5}{2} \quad \binom{5}{3} \quad \binom{5}{4} \quad \binom{5}{5}$$

$$\binom{6}{0} \quad \binom{6}{1} \quad \binom{6}{2} \quad \binom{6}{3} \quad \binom{6}{4} \quad \binom{6}{5} \quad \binom{6}{6}$$

Analizando este triángulo, se obtiene la siguiente relación:

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

Si lo expresamos como una ecuación de recurrencia sería:

$$B(n,k) = \begin{cases} 1 & \text{si } n = k \text{ ó } k = 0 \\ B(n-1,\ k-1) + B(n-1,\ k) & \text{e.o.c.} \end{cases}$$

2. Considere la siguiente definición del problema del vendedor viajero:

   Dado un grafo completo con pesos en las aristas, se define como Camino Hamiltoniano (CH) aquel que recorre todos los nodos una única vez y vuelve al nodo inicial. El costo de un CH es la suma de los pesos de todas sus aristas. Dado un grafo $G(V, E)$, ¿cuál es el costo del CH de menor costo en G?

   Solución:

   Escogemos un nodo arbitrario $f \in V$ como el nodo donde inicia y termina el recorrido.

   El resto del CH debe pasar por los demás nodos, que están determinados por $V' = V - \{f\}$

   Si consideramos $|V'| = n$, sea $L = [i_1, i_2, \cdots, i_n]$ la lista de los vertices en el orden en que se recorren en un CH. Es decir, el CH consiste en

   $$f \to i_1 \to i_2 \to \cdots \to i_n \to f$$

   Para construir un CH de manera recursiva, vamos eligiendo un nodo no visitado en cada paso, y lo vamos agregando a L, la cual inicialmente está vacía. S es el conjunto de nodos no visitados:

   $$CH(S, L) = \begin{cases} \text{agregar el único elemento de S al final de L} & \text{si } |S| = 1 \\ \text{elegir un } i \in S \text{ , agregarlo al final de L, y llamar } CH(S - \{i\}, L) \end{cases}$$

   Pero en este caso no nos interesa el camino que se construye, sino su costo. Definimos una función análoga para el costo, donde i es el nodo que se agregó en el paso anterior, y f es el nodo que fijamos al principio.

   $$C(S, i, f) = \begin{cases} w(i, i') + w(i', f) & \text{si } |S| = 1 \text{ con } S = \{i'\} \\ w(i, i') + C(S - \{i'\}, i', f) & \text{con } i' \in S \end{cases}$$

   Si llamamos $C(V', f, f)$ obtenemos el costo de un CH que parte y termina en f.

   Ahora nuestro objetivo es encontrar el mínimo posible costo de un CH en el grafo G.

   Para esto en lugar de elegir un $i' \in S$ aleatorio en cada paso, elegimos el i' que nos permite obtener el menor costo final.

   $$C(S, i, f) = \begin{cases} w(i, i') + w(i', f) & \text{si } |S| = 1 \text{ con } S = \{i'\} \\ \min\limits_{i' \in S} [w(i, i') + C(S - \{i'\}, i', f)] \end{cases}$$

   Es decir, si ya agregué todos los nodos al CH menos uno, entonces agrego el costo de conectar el último nodo al anterior y al final. En todo otro caso, pruebo todos los nodos de S, agregándolos al final de mi CH que estoy construyendo y me quedo el que me genere el menor costo total. Para obtener el minimo llamamos $C(V', f, f)$

3. Tengo un frasco con n pastillas y tengo que tomarme media al día. Cada noche, antes de ir a dormir meto la mano al frasco: Si saco media pastilla, me la tomo y si saco una entera, la parto por la mitad, me tomo una mitad y devuelvo la otra mitad al frasco. ¿Cuál es la probabilidad exacta de sacar media pastilla al meter la mano en el día x, donde x es un número entero entre 1 y $2n$?

   Solución:

   El evento que nos interesa es $M_{D,E,M,X}$, que vamos a definir como el hecho de sacar **media** pastilla en X días más a partir del día D en que tengo E pastillas enteras y M mitades.

   En este caso, el complemento de este evento, $\overline{M}_{D,E,M,X}$ podemos definirlo como $E_{D,E,M,X}$, es decir, el hecho de sacar una pastilla **entera** en X días más a partir del día D en que tengo E pastillas enteras y M mitades.

   Cuando tenemos un caso como este podemos usar el teorema de **probabilidad total**, que dice que

$$P(A) = \begin{array}{c} P(A \mid B) \cdot P(B) \\ + \\ P(A \mid \overline{B}) \cdot P(\overline{B}) \end{array}$$

   Donde la probabilidad condicional $P(A \mid B)$ es la probabilidad de que ocurra A **dado que** ocurrió B

   Por definición, la probabilidad de sacar media pastilla el día D, es decir, $X = 0$ dias despues de D es

$$P(M_{D,E,M,0}) = \frac{M}{E+M}$$

   Si $X \neq 0$ sabemos que lo que pase en X días más dependerá de lo que pase hoy, y hoy pueden pasar dos cosas: puedo sacar media pastilla, o una pastilla entera. Aplicando el teorema de probabilidad total:

$$P(M_{D,E,M,X}) = \begin{array}{c} P(M_{D+1,E,M,X-1} \mid M_{D,E,M,0}) \cdot P(M_{D,E,M,0}) \\ + \\ P(M_{D+1,E,M,X-1} \mid E_{D,E,M,0}) \cdot P(E_{D,E,M,0}) \end{array}$$

   La primera parte podemos leerla como la probabilidad de sacar media pastilla en $X-1$ días más a partir del día $D+1$, dado que saqué media pastilla en el día D. La segunda parte es análoga, pero dado que saqué una pastilla entera. Cada una multiplicada por su probabilidad de ocurrencia.

Por definición, si el día D tenía E y M, y saqué saqué media pastilla, entonces en el día $D+1$ tendré E y $M-1$. Por otro lado, si en el día D una pastilla entera, entonces en el día $D+1$ tendré $E-1$ y $M+1$.

Podemos usar esto para eliminar la probabilidad condicional:

$$P(M_{D+1,E,M,X-1} \mid M_{D,E,M,0}) = P(M_{D+1,E,M-1,X-1})$$

$$P(M_{D+1,E,M,X-1} \mid E_{D,E,M,0}) = P(M_{D+1,E-1,M+1,X-1})$$

Por lo que queda como

$$P(M_{D,E,M,X}) = \begin{array}{c} P(M_{D+1,E,M-1,X-1}) \cdot P(M_{D,E,M,0}) \\ + \\ P(M_{D+1,E-1,M+1,X-1}) \cdot P(E_{D,E,M,0}) \end{array}$$

Reemplazando los casos base nos queda

$$P(M_{D,E,M,X}) = \begin{array}{c} P(M_{D+1,E,M-1,X-1}) \cdot \dfrac{M}{E+M} \\ + \\ P(M_{D+1,E-1,M+1,X-1}) \cdot \dfrac{E}{E+M} \end{array}$$

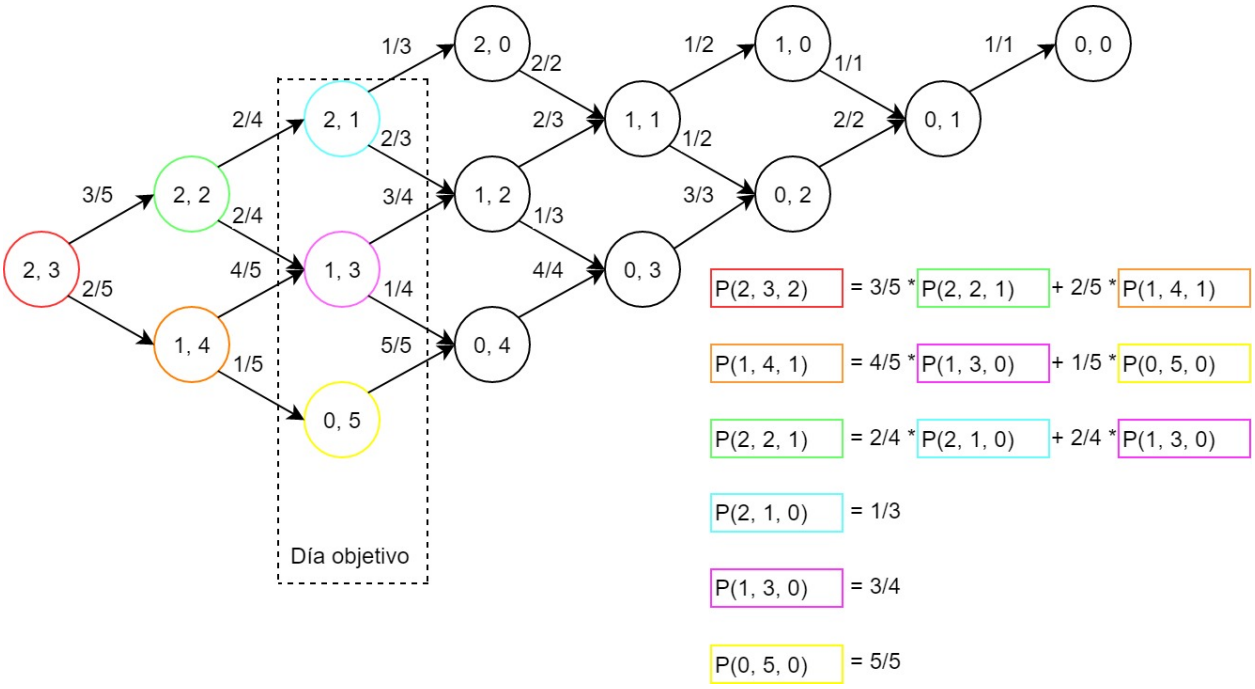
La recurrencia completa quedaría entonces como

$$P(E,M,X) = \begin{cases} \dfrac{M}{E+M} & \text{si } X = 0 \\ \dfrac{M}{E+M} \cdot P(M_{D+1,E,M-1,X-1}) + \dfrac{E}{E+M} \cdot P(M_{D+1,E-1,M+1,X-1}) & \text{e.o.c.} \end{cases}$$

Podemos ver que el parámetro D no afecta realmente en la función, por lo que es posible eliminarlo. Renombrando la función $P(M_{D,E,M,X})$ como $P(E,M,X)$ tenemos entonces que

$$P(E,M,X) = \begin{cases} \dfrac{M}{E+M} & \text{si } X = 0 \\ \dfrac{M}{E+M} \cdot P(E,M-1,X-1) + \dfrac{E}{E+M} \cdot P(E-1,M+1,X-1) & \text{e.o.c.} \end{cases}$$

A continuación se incluye una imagen que ilustra el problema:



Para el problema que se propone entonces se obtiene la probabilidad al llamar la función con los parámetros $P(n, 0, x - 1)$.

Ejercicios Propuestos

4. Dado un string $s = a_1 \cdots a_n$, ¿Cuál es el mínimo k tal que $s = p_1 \cdots p_k$, donde p_i es un palíndromo?

   Solución: Lo primero que podemos notar de este problema es que si el string s es un palíndromo, entonces la solución es $k = 1$, ya que $s = p_1$.

   Si s no es un palíndromo tenemos que ver cómo subdividir el string para obtener la solución óptima en el string completo. Por ejemplo:

   s = BANANA puede ser dividido de manera óptima en s_1 = B y s_2 = ANANA ya que tanto B como ANANA son palíndromos. En este caso la cantidad mínima de palíndromos con que escribimos BANANA = 2.

   Por otro lado si dividimos BANANA en s_1 = BA y s_2 = NANA vamos a estar haciendo una división que no nos lleva al óptimo.

   Dicho esto podemos definir nuestra función de recursión usando la división de la palabra. Para trabajar con índices en vez de strings la función se llamará como $K(s, i, j)$ y su significado es: el mínimo de palíndromos con que se puede escribir el string $s[i : j]$. Por ejemplo:

   $K(\text{BANANA}, 0, 2)$ = mínimo de palíndromos con que se puede escribir BAN, en este caso es igual a 3.

   Esta notación nos permite iterar por las subdivisiones de una palabra para encontrar la división que minimiza el número de palíndromos:

$$K(s, i, j) = \min_{x\,=\,i}^{j}[K(s, i, x) + K(s, x + 1, j)]$$

   Usando esto podemos escribir la recurrencia completa:

$$K(s, i, j) = \begin{cases} 1 & \text{si } a_i a_{i+1} \cdots a_{j-1} a_j \text{ es un palíndromo} \\ \min\limits_{x=i}^{j}(K(s, i, x) + K(s, x + 1, j)) & \text{en otro caso} \end{cases}$$

   En palabras esto es: Si la palabra es un palíndromo, entonces $k = 1$, y si no es un palíndromo entonces k es el mínimo número de palíndromos que puedo obtener subdividiendo la palabra en todos los modos posibles.

5. Dado 2 strings A y B, y 3 operaciones:

   - Eliminar una letra en cualquier posición del string.

     Ejemplo:
     $A = abcd$
     $>$ Eliminar $A[1]$
     $A = acd$
   - Agregar una letra arbitraria en cualquier posición del string.
     Ejemplo
     $A = abcd$
     $>$ Insertar x en $A[2]$
     $A = abxcd$
     Notar que $A[2] = x$
   - Reemplazar una letra en cualquier posición del string por una letra arbitraria.
     Ejemplo:
     $A = abcd$
     $>$ Reemplazar $A[3]$ por x
     $A = abcx$

   ¿Cuál es la mínima cantidad de operaciones que deben aplicarse sobre A para que $A = B$?

   Solución: Tenemos que tratar de calcular la edit distance entre 2 palabras usando la edit distance entre 2 palabras más pequeñas. Para esto se puede pensar a nivel de letra en cada palabra:

   Digamos que $A = a_1, a_2, ..., a_n$ y $B = b_1, b_2, ..., b_m$. Si $a_1 = b_1$ entonces la edit distance de A con B es la misma que la edit distance entre $A[2 : n]$ y $B[2 : m]$ (A y B sin sus primeras letras).

   Si en cambio $a_1 \neq b_1$ entonces tenemos 3 opciones: eliminar a_1, eliminar b_1 o cambiar a_1 para que sea igual a b_1. Esto se ve mejor en un ejemplo:

   $A =$ BANANA, $B =$ ANANA. En este caso $a_1 \neq b_1$ y nos conviene eliminar a_1 y a que así la edit distance de lo que queda es 0.

   En el caso contrario en que $A =$ ANANA, $B =$ BANANA nos conviene eliminar b_1. Notar que esto es equivalente a agregar una letra B al string A pero esto no achica las palabras.

   En el caso en que $A =$ BANANA y $B =$ CANANA la operación que nos conviene usar es el cambio de letra de manera de que $a_1 = b_1$ y así $A = B$.

   En este ejemplo es fácil ver la operación correcta pero muchas veces no es tan fácil. Por ejemplo: $A =$ NGRFAJHFMY y $B =$ FGYRBAJHFJHF.

Para solucionar esto simpleente probamos las 3 operaciones y nos quedamos con el mejor resultado:

$$ED(A,i,B,j) = \min[ED(A,i,B,j+1)+1, ED(A,i+1,B,j)+1, ED(A,i+1,B,j+1)]$$

La notación $ED(A,i,B,j)$ quiere decir edit distance entre el string $A[i:]$ y el string $B[j:]$

Estos 3 sub problemas dentro de min[] reprecentan cada una de las operaciones que pude haber hecho a A y a B, y se les suma 1 ya que este es el costo de la operación.

Ahora solo nos faltan los casos base:

Si $A[i:] = \emptyset$, entonces $ED(A,i,B,j) = |B[j:]|$. Esto es así ya que la edit distance entre un string vacío y otro string es el largo del string.

En el caso simétrico si $B[j:] = \emptyset$, entonces $ED(A,i,B,j) = |A[i:]|$

Finalmente nuestra función de recurrencia nos queda como:

$$ED(A,i,B,j) = \begin{cases} |B[j:]| & \text{si } A[i:] = \emptyset \\ |A[i:]| & \text{si } B[j:] = \emptyset \\ ED(A,i+1,B,j+1) & \text{si } A[i] = B[j] \\ \text{mín}[ED(A,i,B,j+1)+1, \\ ED(A,i+1,B,j)+1, \\ ED(A,i+1,B,j+1)+1] & \text{else} \end{cases}$$