Algoritmos Programación Dinámica

José de Jesús Lavalle Martínez

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Maestría en Ciencias de la Computación
Análisis y Diseño de Algoritmos
MCOM 20300

Otoño 2020

Contenido

Introducción I

- En la sesión anterior vimos que a menudo es posible dividir una instancia en subinstancias, resolver las subinstancias (quizás dividiéndolas más) y luego combinar las soluciones de las subinstancias para resolver la instancia original.

Introducción I

- A veces sucede que la forma natural de dividir una instancia sugerida por la estructura del problema nos lleva a considerar varias subinstancias superpuestas.

Introducción I

- Si resolvemos cada uno de estos de forma independiente, a su vez crearán una gran cantidad de subinstancias idénticas.

Introducción II

- Si no prestamos atención a esta duplicación, es probable que terminemos con un algoritmo ineficaz; si, por otro lado, aprovechamos la duplicación y nos disponemos a resolver cada subinstancia sólo una vez, guardando la solución para su uso posterior, resultará un algoritmo más eficiente.

Introducción II

- La idea subyacente de la programación dinámica es, por lo tanto, bastante simple: evite calcular lo mismo dos veces, generalmente manteniendo una tabla de resultados conocidos que se llena a medida que se resuelven las subinstancias.

Introducción II

- Divide y vencerás es un método de arriba hacia abajo. Cuando un problema se resuelve mediante el método divide y vencerás, atacamos inmediatamente la instancia completa, que luego dividimos en subinstancias cada vez más pequeñas a medida que avanza el algoritmo.

Introducción III

- La programación dinámica, por otro lado, es una técnica de abajo hacia arriba.

Introducción III

- Por lo general, comenzamos con las subinstancias más pequeñas y, por lo tanto, las más simples.

Introducción III

- Combinando sus soluciones, obtenemos las respuestas a subinstancias de tamaño creciente, hasta que finalmente llegamos a la solución de la instancia original.

Introducción III

- Veremos dos ejemplos simples de programación dinámica que ilustran la técnica general en un entorno sencillo.

Cálculo del coeficiente binomial I

- Considere el problema de calcular el coeficiente binomial

$$\binom{n}{k} = \begin{cases} 1 & \text{si } k = 0 \text{ o } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{si } 0 < k < n \\ 0 & \text{en otro caso.} \end{cases}$$

Cálculo del coeficiente binomial I

- Considere el problema de calcular el coeficiente binomial

$$\binom{n}{k} = \begin{cases} 1 & \text{si } k = 0 \text{ o } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{si } 0 < k < n \\ 0 & \text{en otro caso.} \end{cases}$$

- Suponga que $0 \leq k \leq n$. Si calculamos $\binom{n}{k}$ directamente con

function $\mathrm{C}(n, k)$
1 **if** $k = 0$ **or** $k = n$
2 **then return** 1
3 **else return** $\mathrm{C}(n-1, k-1) + \mathrm{C}(n-1, k)$

Cálculo del coeficiente binomial I

- Considere el problema de calcular el coeficiente binomial

$$\binom{n}{k} = \begin{cases} 1 & \text{si } k = 0 \text{ o } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{si } 0 < k < n \\ 0 & \text{en otro caso.} \end{cases}$$

- Suponga que $0 \leq k \leq n$. Si calculamos $\binom{n}{k}$ directamente con

```
function C(n, k)
1   if k = 0 or k = n
2      then return 1
3      else  return C(n − 1, k − 1) + C(n − 1, k)
```

- muchos de los valores $\mathrm{C}(i, j), i < n, j < k$, se calculan una y otra vez. Por ejemplo, el algoritmo calcula $\mathrm{C}(5, 3)$ como la suma de $\mathrm{C}(4, 2)$ y $\mathrm{C}(4, 3)$. Ambos resultados intermedios requieren que calculemos $\mathrm{C}(3, 2)$. De manera similar, el valor de $\mathrm{C}(2, 2)$ se usa varias veces.

Cálculo del coeficiente binomial II

- Dado que el resultado final se obtiene sumando un número de 1s, el tiempo de ejecución de este algoritmo seguramente estará en $\Omega(\binom{n}{k})$.

Cálculo del coeficiente binomial II

- Encontramos un fenómeno similar antes en el algoritmo FIBREC para calcular la secuencia de Fibonacci; consulte la Sección 2.7.5.

Cálculo del coeficiente binomial II

- Si, por el contrario, utilizamos una tabla de resultados intermedios, este es, por supuesto, el triángulo de Pascal, obtenemos un algoritmo más eficiente; vea la Tabla 1.

Cálculo del coeficiente binomial III

	0	1	2	$\cdots$	$k-1$	k
0	1					
1	1	1				
2	1	2	1			
$\vdots$						
$n-1$	1				$\mathrm{C}(n-1,k-1)$	$\mathrm{C}(n-1,k)$
n	1					$\mathrm{C}(n,k)$

Tabla 1: El triángulo de Pascal

Cálculo del coeficiente binomial IV

- La tabla debe llenarse línea por línea.

Cálculo del coeficiente binomial IV

- De hecho, ni siquiera es necesario almacenar la tabla completa: basta con mantener un vector de longitud k, que represente la línea actual, y actualizar este vector de izquierda a derecha.

Cálculo del coeficiente binomial IV

- Así, para calcular $\binom{n}{k}$ el algoritmo toma un tiempo en $\Theta(nk)$, si asumimos que la suma es una operación elemental, y un espacio en $\Theta(k)$.

La serie mundial I

- Imagine una competencia en la que dos equipos A y B juegan no más de $2n-1$ juegos, siendo el ganador el primer equipo en lograr n victorias.

La serie mundial I

- Suponemos que no hay juegos empatados, que los resultados de cada partido son independientes y que para cualquier partido dado hay una probabilidad constante p de que el equipo A sea el ganador y, por lo tanto, una probabilidad constante $q = 1 - p$ de que el equipo B ganará.

La serie mundial I

- Sea $P(i, j)$ la probabilidad de que el equipo A gane la serie dado que todavía necesita i victorias más para lograrlo, mientras que el equipo B aún necesita j victorias más si quiere ganar.

La serie mundial II

- Por ejemplo, antes del primer juego de la serie, la probabilidad de que el equipo A sea el ganador de la serie es $P(n, n)$: ambos equipos aún necesitan n victorias para ganar la serie.

La serie mundial II

- Si el equipo A requiere 0 victorias más, entonces de hecho ya ganó la serie, por lo que $P(0, i) = 1, 1 \leq i \leq n$.

La serie mundial II

- De manera similar, si el equipo B requiere 0 victorias más, entonces ya ha ganado la serie y la probabilidad de que el equipo A sea el ganador de la serie es cero: entonces $P(i, 0) = 0, 1 \leq i \leq n$.

La serie mundial III

- Dado que no puede haber una situación en la que ambos equipos hayan ganado todos los partidos que necesitan, $P(0,0)$ no tiene sentido.

La serie mundial III

- Finalmente, dado que el equipo A gana un partido dado con probabilidad p y lo pierde con probabilidad q,

$$P(i,j) = pP(i-1,j) + qP(i,j-1), i \geq 1, j \geq i.$$

La serie mundial III

- Finalmente, dado que el equipo A gana un partido dado con probabilidad p y lo pierde con probabilidad q,

$$P(i,j) = pP(i-1,j) + qP(i,j-1), i \geq 1, j \geq i.$$

- Así podemos calcular $P(i,j)$ como sigue:

function $\mathrm{P}(i,j)$
1 **if** $i = 0$
2 **then return** 1
3 **else if** $j = 0$
4 **then return** 0
5 **else return** $p\mathrm{P}(i-1,j) + q\mathrm{P}(i,j-1)$

La serie mundial IV

- Sea $T(k)$ el tiempo necesario en el peor de los casos para calcular $P(i, j)$, donde $k = i + j$.

La serie mundial IV

- Con este método, vemos que

$$T(1) = c$$
$$T(k) \leq 2T(k-1) + d, k > 1$$

donde c y d son constantes.

La serie mundial IV

- Reescribiendo $T(k-1)$ en términos de $T(k-2)$, y así sucesivamente, encontramos

$$
\begin{aligned}
T(k) &\leq 4T(k-2) + 2d + d, k > 2 \\
&\vdots \\
&\leq 2^{k-1}T(1) + (2^{k-2} + 2^{k-3} + \cdots + 2 + 1)d \\
&= 2^{k-1}c + (2^{k-1} - 1)d \\
&= 2^k(c/2 + d/2) - d.
\end{aligned}
$$

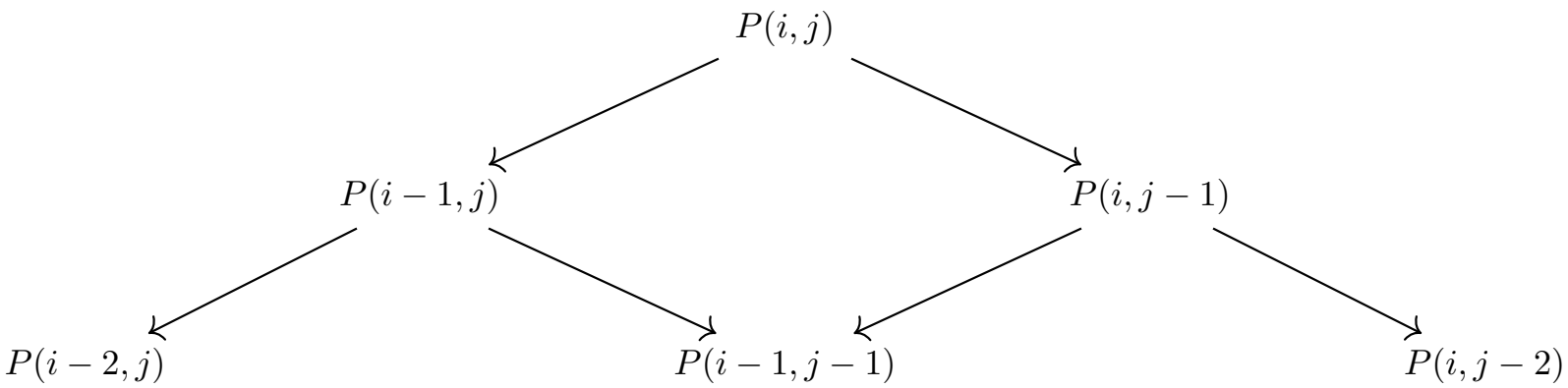
La serie mundial V

- Por lo tanto $T(k)$ está en $O(2^k)$, que está en $O(4^n)$ si $i = j = n$.

La serie mundial V

- De hecho, si observamos la forma en que se generan las llamadas recursivas, encontramos el patrón que se muestra en la Figura 1, que es idéntico al obtenido en el cálculo ingenuo del coeficiente binomial.

La serie mundial VI



etc.

Figura 1: Llamados recursivos que se hacen por un llamado a $P(i,j)$.

La serie mundial VII

- Para ver esto, imagine que cualquier llamada $P(m, n)$ en la figura se reemplaza por $C(m + n, n)$.

La serie mundial VII

- Para ver esto, imagine que cualquier llamada $P(m, n)$ en la figura se reemplaza por $C(m + n, n)$.
- Así, $P(i, j)$ se reemplaza por $C(i + j, j)$, $P(i - 1, j)$ por $C(i + j - 1, j)$ y $P(i, j - 1)$ por $C(i + j - 1, j - 1)$.

La serie mundial VII

- Para ver esto, imagine que cualquier llamada $P(m, n)$ en la figura se reemplaza por $C(m + n, n)$.
- Así, $P(i, j)$ se reemplaza por $C(i + j, j)$, $P(i - 1, j)$ por $C(i + j - 1, j)$ y $P(i, j - 1)$ por $C(i + j - 1, j - 1)$.
- Ahora el patrón de llamadas que muestran las flechas corresponde al cálculo

$$C(i + j, j) = C(i + j - 1, j) + C(i + j - 1, j - 1)$$

de un coeficiente binomial.

La serie mundial VIII

- Por tanto, el número total de llamadas recursivas es exactamente $2\binom{i+j}{j} - 2$; vea el problema 8.1 del libro de texto.

La serie mundial VIII

- Para calcular la probabilidad $P(n, n)$ de que gane el equipo A dado que la serie aún no ha comenzado, el tiempo requerido está entonces en $\Omega(\binom{2n}{n})$.

La serie mundial VIII

- El problema 1.42 del libro de texto pide al lector que muestre que $\binom{2n}{n} \geq 4^n/(2n+1)$.

La serie mundial VIII

- Combinando estos resultados, vemos que el tiempo requerido para calcular $P(n, n)$ está en $O(4^n)$ y en $\Omega(4^n/n)$.

La serie mundial VIII

- Por lo tanto, el método no es práctico para valores grandes de n (aunque las competiciones deportivas con $n > 4$ son la excepción, ¡este problema tiene otras aplicaciones!).

La serie mundial IX

- Para acelerar el algoritmo, procedemos más o menos como con el triángulo de Pascal: declaramos una matriz del tamaño apropiado y luego completamos las entradas.

La serie mundial IX

- Esta vez, sin embargo, en lugar de llenar la matriz línea por línea, trabajamos diagonal por diagonal.

La serie mundial X

Aquí está el algoritmo para calcular $P(n, n)$.

function SERIES(n, p)

```
 1  array P[0...n, 0...n]
 2  q ← 1 − p
 3  for s ← 1 to n
 4  do P[0, s] ← 1;  P[s, 0] ← 0
 5      for k ← 1 to s − 1
 6      do P[k, s − k] ← pP[k − 1, s − k] + qP[k, s − k − 1]
 7  for s ← 1 to n
 8  do for k ← 0 to n − s
 9      do P[s + k, n − k] ← pP[s + k − 1, n − k] + qP[s + k, n − k − 1]
10  return P[n, n]
```

La serie mundial XI

- Dado que el algoritmo tiene que llenar una matriz $n \times n$ y dado que se requiere un tiempo constante para calcular cada entrada, su tiempo de ejecución está en $\Theta(n^2)$.

La serie mundial XI

- Al igual que con el triángulo de Pascal, es fácil implementar este algoritmo para que un espacio de almacenamiento en $\Theta(n)$ sea suficiente.

Ejercicios I

Para el viernes 6 de noviembre:

- Equipo 1: Secciones 8.2 y 8.3 del libro de texto.
- Equipo 2: Sección 8.4 del libro de texto.
- Equipo 3: Sección 8.5 del libro de texto.
- Equipo 4: Sección 8.6 del libro de texto.
- Equipo 5: Sección 8.7 del libro de texto.
- Equipo 6: Sección 8.8 del libro de texto.

Ejercicios II

Para el viernes 13 de noviembre:
Para todos estudiar las secciones 10.1 a 10.4 del libro de texto.

- Equipo 1: Secciones 9.1 y 9.2 del libro de texto.
- Equipo 2: Secciones 9.3 y 9.4 del libro de texto.
- Equipo 3: Secciones 9.5 y 9.6 del libro de texto.
- Equipo 4: Secciones 9.7 y 9.8 del libro de texto.
- Equipo 5: Sección 10.6 del libro de texto.
- Equipo 6: Sección 10.7 del libro de texto.

Ejercicios III

Para el viernes 4 de diciembre:

- Equipo 1: Swarm intelligence.
- Equipo 2: Tabu search.
- Equipo 3: Simulated Annealing.
- Equipo 4: Genetic algorithms.
- Equipo 5: Artificial Neural Networks.
- Equipo 6: Support Vector Machines.