Programación Dinámica

Certamen Nacional OIA 2016

La sucesión de Fibonacci

La sucesión de Fibonacci se define como:

- Los primeros dos términos son 1.
- Desde ahí, cada término es la suma de los dos anteriores.

Los primeros términos son: $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \ldots$

Problema

Calcular el término número 100 de la sucesión.

Programación Dinámica

La técnica de programación dinámica consiste en:

- Construir la solución de mi problema a partir de la solución de problemas más sencillos.
- Hay que identificar cuales son los problemas más sencillos.

Importante

¡No calcular el mismo problema dos veces!

Un problema posible

Los tesoros del pirata

En la guarida del tesoro hay una hilera de n cofres con monedas. El i-ésimo cofre tiene m_i monedas. Los cofres están protegidos con un sistema de alarmas. Sin embargo, como se produjeron muchas falsas alarmas últimamente, se decidió que no se activen las alarmas cuando se abre un sólo cofre sino cuando se abren al menos dos cofres consecutivos de la hilera. Los dueños sospechan que un ladrón abrirá en su apuro varios cofres seguidos y será fácil detectarlos. Sin embargo, se dieron cuenta que el sistema no es infalible y un ladrón con la información de como funciona el sistema se podría robar muchas monedas. Nos piden que calculemos cual es la máxima cantidad de monedas que se pueden llevar sin activar las alarmas.

Algunos ejemplos

Si nuestros cofres tienen:

- 50, 20, 40, y 30 monedas.

Algunos ejemplos

Si nuestros cofres tienen:

- 50, 20, 40, y 30 monedas. Agarrar 50, 20, 40 y 30 se lleva 90 monedas.

Algunos ejemplos

Si nuestros cofres tienen:

- 50, 20, 40, y 30 monedas. Agarrar 50, 20, 40 y 30 se lleva 90 monedas.
- 10, 20, 10 y 5 monedas.

Algunos ejemplos

Si nuestros cofres tienen:

- 50, 20, 40, y 30 monedas. Agarrar 50, 20, 40 y 30 se lleva 90 monedas.
- 10, 20, 10 y 5 monedas. Agarrar 10, 20, 10 y 5 se lleva 25 monedas.

Algunos ejemplos

Si nuestros cofres tienen:

- 50, 20, 40, y 30 monedas. Agarrar 50, 20, 40 y 30 se lleva 90 monedas.
- 10, 20, 10 y 5 monedas. Agarrar 10, 20, 10 y 5 se lleva 25 monedas.
- 20, 15, 15 y 20 monedas.

Algunos ejemplos

Si nuestros cofres tienen:

- 50, 20, 40, y 30 monedas. Agarrar 50, 20, 40 y 30 se lleva 90 monedas.
- 10, 20, 10 y 5 monedas. Agarrar 10, 20, 10 y 5 se lleva 25 monedas.
- 20, 15, 15 y 20 monedas. Agarrar 20, 15, 15 y 20 se lleva 40 monedas.

Solución

Un problema más facil es pensar qué pasaría si sacamos el último cofre.

Solución

Un problema más facil es pensar qué pasaría si sacamos el último cofre.
Creamos un arreglo primeros, donde primeros[n] nos dice lo máximo que se puede llevar un ladrón si roba sólo de los primeros *n* cofres.

Solución

Un problema más facil es pensar qué pasaría si sacamos el último cofre.
Creamos un arreglo primeros, donde primeros[n] nos dice lo máximo que se puede llevar un ladrón si roba sólo de los primeros *n* cofres.
primeros[0] = 0, primeros[1] = m[1].

Solución

Un problema más facil es pensar qué pasaría si sacamos el último cofre.
Creamos un arreglo primeros, donde primeros[n] nos dice lo máximo que se puede llevar un ladrón si roba sólo de los primeros *n* cofres.
primeros[0] = 0, primeros[1] = m[1].
primeros[n] = primeros[n-1] + primeros[n-2].

Seguimos

Caminando en un tablero

Tenemos un tablero de $N \times M$. En cada casilla hay un número que representa la cantidad de puntos que vale esa casilla. La casilla de la fila *i* y la columna *j* da puntos[i][j]. Comenzamos con una ficha en la casilla de arriba a la izquierda. En cada turno nos movemos una casilla a la derecha ó una casilla hacia abajo sin salirnos del tablero. Cuando llegamos a la casilla de abajo a la derecha terminamos. Nuestro puntaje total es la suma de los puntos que dan todas las casillas por las que pasó la ficha. ¿Cuál es la máxima cantidad de puntos que podemos ganar?

Ejemplo

26	73	26	77	62	13
37	85	28	48	62	61
33	30	73	20	38	10
18	44	73	54	23	16
84	35	27	83	95	26
34	45	56	23	46	17

Este sería un posible tablero del juego.

Ejemplo

26	73	26	77	62	13
37	85	28	48	62	61
33	30	73	20	38	10
18	44	73	54	23	16
84	35	27	83	95	26
34	45	56	23	46	17

Este camino gana
$26 + 37 + 85 + 28 + 73 + 20 + 54 + 83 + 23 + 46 + 17 = 492$ puntos.

Solución

Un problema más facil es pensar qué pasaría si el objetivo es llegar a una casilla más cerca del inicio.

Solución

Un problema más facil es pensar qué pasaría si el objetivo es llegar a una casilla más cerca del inicio.
Creamos una matriz puntajeMaximo, donde puntajeMaximo[i][j] nos dice el máximo puntaje posible terminando en la casilla de la fila *i* y la columna *j*.

Solución

Un problema más facil es pensar qué pasaría si el objetivo es llegar a una casilla más cerca del inicio.
Creamos una matriz puntajeMaximo, donde puntajeMaximo[i][j] nos dice el máximo puntaje posible terminando en la casilla de la fila *i* y la columna *j*.
puntajeMaximo[0][0] = puntos[0][0].

Solución

Un problema más facil es pensar qué pasaría si el objetivo es llegar a una casilla más cerca del inicio.
Creamos una matriz puntajeMaximo, donde puntajeMaximo[i][j] nos dice el máximo puntaje posible terminando en la casilla de la fila *i* y la columna *j*.
puntajeMaximo[0][0] = puntos[0][0].
puntajeMaximo[i][j] = max(puntajeMaximo[i-1][j], puntajeMaximo[i][j-1]).

Importante

¡Cuidado con los bordes!

Problema 3

Máximo subarreglo

Dado un arreglo *numeros* de *n* números enteros, se quiere elegir algunos elementos del arreglo tales que sean consecutivos en el arreglo y sumen lo máximo posible ¿Cuál es la suma máxima que se puede conseguir?

Problema 3

Máximo subarreglo

Dado un arreglo *numeros* de *n* números enteros, se quiere elegir algunos elementos del arreglo tales que sean consecutivos en el arreglo y sumen lo máximo posible ¿Cuál es la suma máxima que se puede conseguir?

Importante

Pueden haber números negativos.

Ejemplos

- $[10, 30, 20, 40, 50]$.

Ejemplos

- $[10, 30, 20, 40, 50]$. Conviene agarrar todos.

Ejemplos

- $[10, 30, 20, 40, 50]$. Conviene agarrar todos.
- $[20, 30, -50, 10, 15]$.

Ejemplos

- $[10, 30, 20, 40, 50]$. Conviene agarrar todos.
- $[20, 30, -50, 10, 15]$. Conviene agarrar los primeros dos.

Ejemplos

- $[10, 30, 20, 40, 50]$. Conviene agarrar todos.
- $[20, 30, -50, 10, 15]$. Conviene agarrar los primeros dos.
- $[-10, 5, -1, 5, -4]$.

Ejemplos

- $[10, 30, 20, 40, 50]$. Conviene agarrar todos.
- $[20, 30, -50, 10, 15]$. Conviene agarrar los primeros dos.
- $[-10, 5, -1, 5, -4]$. Conviene agarrar los tres del medio.

Solucion

Un problema más fácil es el subarreglo máximo que termina en cierta posición.

Solucion

Un problema más fácil es el subarreglo máximo que termina en cierta posición.
Creamos un arreglo sumaMaxima, donde sumaMaxima[i] es la suma máxima elegiendo números tal que *i* sea el último.

Solucion

Un problema más fácil es el subarreglo máximo que termina en cierta posición.
Creamos un arreglo sumaMaxima, donde sumaMaxima[i] es la suma máxima elegiendo números tal que *i* sea el último.
sumaMaxima[0] = numeros[0].

Solucion

Un problema más fácil es el subarreglo máximo que termina en cierta posición.
Creamos un arreglo sumaMaxima, donde sumaMaxima[i] es la suma máxima elegiendo números tal que *i* sea el último.
sumaMaxima[0] = numeros[0].
sumaMaxima[i] = max(numeros[i], sumaMaxima[i-1] + numeros[i])

Problema 4

Suma de subconjuntos

Se tienen *n* varillas. La *i*-ésima varilla mide longitud[i]. Se pueden elegir algunas varillas y pegarlas para armar una varilla de longitud más grande ¿Cuántas longitudes menores ó iguales a *m* distintas de varillas se pueden armar? Se pueden reusar las varillas.

Ejemplos

- $[1, 2, 2]$ y $m = 10$.

Ejemplos

- $[1, 2, 2]$ y $m = 10$. Se puede hacer cualquier varilla de longitud 1 a 5.

Ejemplos

- $[1, 2, 2]$ y $m = 10$. Se puede hacer cualquier varilla de longitud 1 a 5.
- $[1, 2, 8]$ y $m = 10$.

Ejemplos

- $[1, 2, 2]$ y $m = 10$. Se puede hacer cualquier varilla de longitud 1 a 5.
- $[1, 2, 8]$ y $m = 10$. Se pueden hacer 7 longitudes distintas, sólo 6 menores ó iguales a 10.

Ejemplos

- $[1, 2, 2]$ y $m = 10$. Se puede hacer cualquier varilla de longitud 1 a 5.
- $[1, 2, 8]$ y $m = 10$. Se pueden hacer 7 longitudes distintas, sólo 6 menores ó iguales a 10.
- $[3, 3, 3, 3]$ y $m = 10$.

Ejemplos

- $[1, 2, 2]$ y $m = 10$. Se puede hacer cualquier varilla de longitud 1 a 5.
- $[1, 2, 8]$ y $m = 10$. Se pueden hacer 7 longitudes distintas, sólo 6 menores ó iguales a 10.
- $[3, 3, 3, 3]$ y $m = 10$. Puedo armar sólo de longitudes 3, 6 y 9.

Solución

Un problema más fácil es si con las primeras varillas puedo armar una de longitud k.

Solución

Un problema más fácil es si con las primeras varillas puedo armar una de longitud *k*. Creo una matriz *puedo*, donde puedo[i][k] me dice si puedo armar una varilla de longitud *k* usando sólo las primeras *i*.

Solución

Un problema más fácil es si con las primeras varillas puedo armar una de longitud k. Creo una matriz *puedo*, donde puedo[i][k] me dice si puedo armar una varilla de longitud k usando sólo las primeras i. puedo[0][0] = true, puedo[0][k] = false si $k > 0$.

Solución

Un problema más fácil es si con las primeras varillas puedo armar una de longitud k. Creo una matriz *puedo*, donde puedo[i][k] me dice si puedo armar una varilla de longitud k usando sólo las primeras i. puedo[0][0] = true, puedo[0][k] = false si $k > 0$. puedo[i][k] = puedo[i-1][k] or puedo[i-1][k-longitud[i-1]].

Recursión

Podemos pensar un problema de programación dinámica como una función recursiva:

```
fibonacci(n) :
    si n < 2 :
        return 1;
    else :
        return fibonacci(n-1) + fibonacci(n-2);
```

Dinamizando la recursión

Estamos calculando muchas veces lo mismo. Guardemos lo que ya hayamos calculado:

```
fibonacci(n) :
    si fibo[n] != 0 :
        return fibo[n];
    si n < 2 :
        fibo[n] = 1;
        return 1;
    else :
        fibo[n] = fibonacci(n-1) + fibonacci(n-2);
        return fibo[n];
```

Problemas para pensar

- En el problema del tablero, modificar la solución para que le permita a la ficha retroceder hasta 3 veces en todo el juego. (Pista $O(3 \cdot n \cdot m)$)
- Dada una palabra de longitud n, calcular la mínima cantidad de letras que hay que borrar para que quede un palíndromo. (Pista: $O(n^3)$)
- Dadas dos palabras de longitud n y m. Se quiere transformar la primera en la segunda. Las operaciones permitidas son: borrar una letra, agregar una letra y cambiar una letra por otra ¿Cuál es la mínima cantidad de operaciones necesarias? (Pista: $O(n \cdot m)$)