GSourceCode

GSOURCECODE

INTRODUCCIÓN A LA PROGRAMACIÓN DINÁMICA - CORTAR RODS

12 de abril 2012 · *por* MGhareeb · *en algoritmos , programación dinámica , Problemas , Utilidades · 2 Comments*

2 0 Valora este

Este problema se presenta en la **introducción a los algoritmos** como una introducción a la *programación dinámica* .
Dada una varilla de longitud `n` pulgadas y una tabla de precios `p i` para longitudes de varilla: `i = 1, 2, ... n` , determinar el ingreso máximo `r n` obtenible por cortar la varilla de piezas y venderlas.
La mesa se parece a esto una solución ingenua podría ser

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

```
CUT-ROD (p, n)
q = 0
para i = 1 hasta n
        q = max (q, p [i] + CUT-ROD (p, n - i))
retorno q
```

Esta solución tiene tiempo asintótica exponencial. Así, nos presentan a la programación dinámica.
El método funciona de la siguiente manera:

- Nos reorganizamos para cada subproblema a resolver sólo una vez.
- Si tenemos que hacer referencia a la solución de este subproblema de nuevo más tarde, sól arriba en una tabla hash o una matri:

EXPLORAR UNA CATEGORÍA

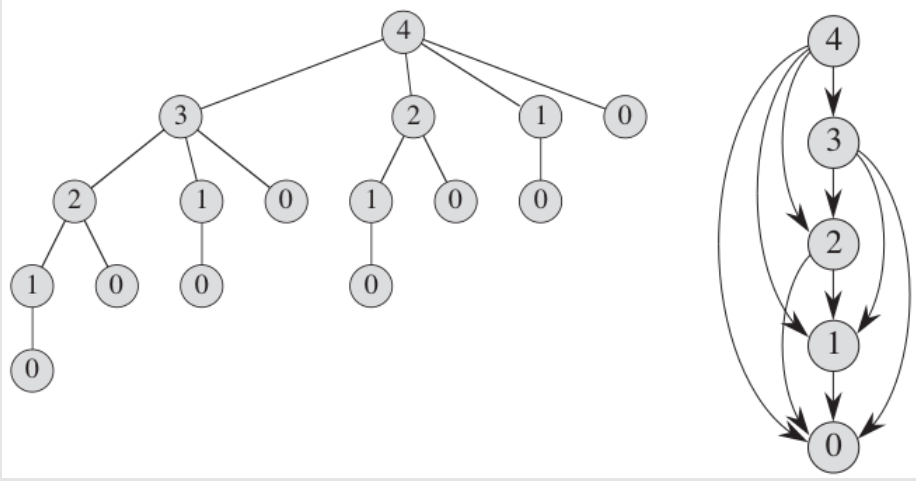
La versión modificada del algoritmo anteri


```
CUT-ROD (p, n)
si CUT-ROD (p, n) se resuelve antes
soluciones anteriores
        retorno s [n]
q = 0
para i = 1 hasta n
        q = max (q, p [i] + CUT-ROD (p, n - i))
s [n] = q // almacenar la nueva solución
retorno q
```

Hay una *de arriba hacia abajo* y de un enfoque *de abajo hacia arriba* enfoque:

- El **top-down** enfoque recursivamente de problemas más grandes a subproblemas más pequeños hasta llegar a un subproblema calculada previamente. Después de eso, devuelve luego combina las soluciones de los subproblemas para resolver los más grandes. El algoritmo anterior es de arriba hacia abajo.
- El **ascendente** método, como se puede deducir de su nombre, resuelve todos los subproblemas requeridos primeros después los más grandes. Ambos métodos son $O(n^2)$ sino la forma de abajo hacia arriba tiene mejores constantes.

Para tener una idea clara de cuál es la diferencia, consulte el siguiente árbol subproblema:



El gráfico subproblema para el problema de barra de corte con n = 4.
Las etiquetas de los vértices dan los tamaños de los subproblemas correspondientes.
Una arista dirigida (x, y) indica que necesitamos una solución al subproblema y en la resolución de subproblema x.
A la izquierda es la solución ingenua.
A la derecha es el DP.

Implementado por debajo en `C ++` .

```
# include <iostream>
# include <cstring>
utilizando  el espacio de nombres  std;

const  int  N = 1000;
int  p [11];
int  r [N], s [N];

// Inicializador para prieces y soluciones óptima
anular  init () {
    memset (r, -1, N);
    r [0] = 0;
    p [0] = 0;
    p [1] = 1;
    p [2] = 5;
    p [3] = 8;
    p [4] = 9;
    p [5] = 10;
    p [6] = 17;
    p [7] = 17;
    p [8] = 20;
    p [9] = 24;
    p [10] = 30;
}

// Solución exponencial naieve
int  cutRod ( int  n) {
    int  q = 0;
    para  ( int  i = 1; i <= n; ++ i)
        q = max (q, p [i] + cutRod (n - i));
    volver  q;
}

// Solución de arriba hacia abajo
int  topDownCutRod ( int  n) {
    si  (r [n]! = -1)
        volver  r [n];
    int  q = 0;
    para  ( int  i = 1; i <= n; ++ i)
        q = max (q, p [i] + topDownCutRod (n - i)
    volver  r [n] = q;
}

// Solución de abajo hacia arriba
int  buttomUpCutRod ( int  n) {
    si  (r [n]! = -1)
        volver  r [n];
    para  ( int  j = 1; j <= n; ++ j) {
        int  q = 0;
        para  ( int  i = 1; i <= j; ++ i)
            q = max (q, p [i] + r [j - i]);
        r [j] = q;
    }
    volver  r [n];
}

// Solución de abajo hacia arriba que mantiene no
int  extendedButtomUpCutRod ( int  n) {
    si  (r [n]! = -1)
        volver  r [n];
    para  ( int  j = 1; j <= n; ++ j) {
        int  q = 0;
        para  ( int  i = 1; i <= j; ++ i)
            si  (q <p [i] + r [j - i]) {
                q = p [i] + r [j - i];
                s [j] = i;
            }
        r [j] = q;
    }
    volver  r [n];
}
```

```
72
73  // Prins salida del método extendido
74  anular  printCutRodSoln ( int  n) {
75      hacer
76          cout << s [n] << "" ;
77      mientras que  ((n - = s [n])> 0);
78  }
79
80  int  main () {
81      init ();
82      int  n;
83      cin >> n;
84      cout << extendedButtomUpCutRod (n) << endl;
85      printCutRodSoln (n);
86      devolver  0;
87  }
```


Compartir

Relacionados

10943 - ¿Cómo se puede añadir
en "Algoritmos"

Subarreglo máxima - Kadane
En "Algoritmos"

1213 - Suma de distintos Primes
En "Algoritmos"

Etiquetas: algoritmos , análisis , parte inferior , la fuerza bruta , corte , corte , abajo , dp , dinámico , introducción , programación , barra , barras , subproblema , arriba , arriba

2 comentarios

638 - Déjame contar las maneras | ·28 de abril 2012 - 20:12 · *Responder →*

[...] - Déjame contar las maneras Publicado el 28 Abril 2012 por MGhareeb32 Un problema de programación dinámica de cambio de la moneda. Un enfoque buttom-up C [...]

Kaidul ·15 de mayo 2013 - 13:59 · *Responder →*

Niza explicación. Yo estaba buscando ayuda para UVA - 10003 y la encontré. Continua!

std :: cout <<

Enter your comment here...

← cálculo relleno de inundación y zona

11703 Sqrt Iniciar Sin →

TOP 5

- 11703 Sqrt Iniciar Sin
- Introducción a la Programación Dinámica - Cortar Rods
- 272 - TEX Cotizaciones
- 10684 - El premio mayor
- gsourcecode

BLOG ESTADÍSTICAS

- 31.079 visitas

AUTORES

MGhareeb

Ad-hoc Algoritmos Android búsqueda binaria completado la estructura de datos de aplicaciones de escritorio Programación Dinámica Geometría Gráfica Búsqueda Heurística J2ME Matemáticas Operaciones de Investigación Problemas Utilidades

META

- Registro
- Iniciar sesión
- Entradas RSS
- Comentarios RSS
- Blog de WordPress.com .