

Programación dinámica



29 de MARZO de 2018

Gema rico pozas

UO238096

TRABAJO PEDIDO 1

**a) Diseñar el algoritmo en cuestión**

public void robar() {

for (int i = 0; i < botinMax; i++) {

for (int j = 0; j <= cargaMax; j++) {

if (i == 0) {

if (j < robo.get(0).getPeso()) {

matriz[i][j] = 0;

} else {

matriz[i][j] = robo.get(0).getValor();

}

}

if (i >= 1) {

if (j < robo.get(i).getPeso()) {

matriz[i][j] = matriz[i - 1][j];

} else {

matriz[i][j] = getMax(matriz[i - 1][j],matriz[i - 1][j - robo.get(i).getPeso()] + robo.get(i).getValor());

}

}

}

}

}

**b) ¿Cree que ese algoritmo hace el robo óptimo para cualquier surtido de joyas (n, p, v y k)?**

Si, este algoritmo de programación dinámica hace el robo óptimo para cualquier surtido de joyas.

**c) Implementar el algoritmo seleccionado.**

*Ver código del proyecto adjunto.*

**d) Calcule la complejidad teórica de la implementación anterior**.

O(n2) 🡪 2 bucles ***for*** anidados con complejidad O(n).

**e) Haga una medición de tiempos de ejecución, debe ir creciendo el tamaño n así: 10, 20, 40, 80…, hasta que se desborde el HEAP. Para cada tamaño n, la generación tanto de cada elemento pi, como de cada vi, serán valores enteros aleatorios en el rango [10..99] y el**

**valor de k será en cada caso k=25\*n**

|  |  |
| --- | --- |
| **Carga de trabajo (n)** | **tiempo en micros** |
| 10 | 0,201 |
| 20 | 0,599 |
| 40 | 2,064 |
| 80 | 12,274 |
| 160 | 48,348 |
| 320 | 425,77 |
| 640 | 2533,3 |
| 1280 | 11371,3 |
| 2560 | 43727,3 |

**f) Compruebe si los tiempos obtenidos en el apartado anterior concuerdan o no con la**

**complejidad teórica, en cada uno de los tres casos o implementaciones.**

Como podemos apreciar en la gráfica la complejidad del algoritmo coincide con lo establecido en el punto d).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 69 | 69 | 69 | 69 | 69 | 69 | 69 | 69 | 69 | 69 | 69 | 69 | 69 |
| 0 | 0 | 22 | 22 | 22 | 22 | 22 | 69 | 69 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 |
| 0 | 0 | 22 | 22 | 22 | 22 | 22 | 69 | 69 | 91 | 91 | 112 | 112 | 112 | 112 | 112 | 159 | 159 | 181 | 181 |
| 0 | 0 | 22 | 22 | 41 | 41 | 63 | 69 | 69 | 91 | 91 | 112 | 112 | 132 | 132 | 153 | 159 | 159 | 181 | 181 |
| 0 | 0 | 22 | 29 | 41 | 51 | 63 | 70 | 70 | 92 | 98 | 112 | 120 | 132 | 141 | 153 | 161 | 161 | 182 | 188 |

TABLA BOTIN 04

TABLA BOTIN 03

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 4 |
| 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 8 |
| 0 | 0 | 0 | 0 | 4 | 6 | 6 | 6 | 8 |