Soluciones entrenos OIE

Raisins

Solución

Las ideas principales son:

- Utilizar 2D prefix sums para poder calcular el valor de todo subrectángulo en tiempo constante.
- Utilizar programación dinámica para determinar el mínimo valor a pagar. La formulación recursiva simplemente examina todos los posibles cortes, verticales y horizontales, y lleva la cuenta del mínimo.

Los detalles de cómo adaptarlos al problema en cuestión están comentados en el código.

Código

C++

```cpp
#include <climits>
#include <iostream>
#include <vector>

using namespace std;

const int N = 51;  // el máximo valor, más 1 para operar en el rango 1..50.
const int M = 51;
vector<vector<int>> ps_v;  // 2D prefix sums de los valores (núm. de raisins).

// Retorna el mínimo coste de partir una tableta rectangular de dimensiones
// horizontales x..X y verticales y..Y
int f(int x, int X, int y, int Y, vector<vector<vector<vector<int>>>> & dp) {
  if (x + 1 == X && y + 1 == Y) return 0;  // caso base
  if (dp[x][X][y][Y]) return dp[x][X][y][Y];
  int minimo = INT_MAX; // Asignamos el mínimo a "infinito", que consideramos que
  ↪  es el máximo valor que entra en un int
  // Calculamos, de entre todos los cortes verticales y horizontales posibles,
  // aquel de coste mínimo.
  // Cortes horizontales
  for (int i = x + 1; i < X; i++) {
    minimo = min(minimo, f(x, i, y, Y, dp) /* cacho superior */ +
                         f(i, X, y, Y, dp) /* cacho inferior */);
  }
```

```cpp
  // Cortes verticales
  for (int i = y + 1; i < Y; i++) {
    minimo = min(minimo, f(x, X, y, i, dp) /* cacho izquierdo*/ +
                         f(x, X, i, Y, dp) /* cacho derecho */);
  }
  // Combinamos el resultado mínimo obtenido de los subproblemas con el coste
  // del problema actual, utilizando 2D prefix sums.
  return dp[x][X][y][Y] =
             minimo + ps_v[X][Y] - ps_v[x][Y] - ps_v[X][y] + ps_v[x][y];
}

int main() {
  int n, m;
  cin >> n >> m;
  ps_v = vector<vector<int>>(n + 1, vector<int>(m + 1));
  vector<vector<vector<vector<int>>>> dp(N, vector<vector<vector<int>>>(M,
  ↪  vector<vector<int>>(N, vector<int>(M, 0))));
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
      cin >> ps_v[i][j];
    }
  }
  // Cálculo de las 2D prefix sums
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
      ps_v[i][j] += ps_v[i - 1][j] + ps_v[i][j - 1] - ps_v[i - 1][j - 1];
    }
  }
  cout << f(0, n, 0, m, dp) << '\n';
}
```