8'5

Problema 43

RAVE ROBAYO, JOSÉ DANIEL (MARP44)

ID envio	Usuario/a	Hora envío	Veredicto
44517	MARP44	2021-11-18 10:35	AC
44508	MARP44	2021-11-18 10:20	AC

Fichero main.cpp

```
  *
  * Nombre y Apellidos:
  *

Sobre un vector de tamaño Max + 1 , siendo Max la potencia permitida, se va rellenando cada
componente del vector usando la función de recurrencia.
FUNCION:
costesDePotencia(i,j):
Consiste en elegir el coste mínimo para conseguir llegar exactamente a la potencia j-ésima
    del vector,
considerando las bombillas de j hasta n, siendo n el número de bombillas.

Una vez rellenado el vector, se recorre desde la posicion (Potencia Mínima) del vector hasta
    el final de éste,
guardando la potencia cuyo coste sea el menor conseguido entre este rango. Se devuelve una
    tupla con el indice
de dicho coste minimo y el coste.

El algoritmo tiene complejidad de O(N*max) siendo N el número de bombillas y max la potencia
    máxima permitida.

struct Sol {
  int power;
  EntInf costMin;
};

Sol minCostLight(vector<int> const& potencia, vector<int> const& coste, int min, int max) {
  int N = potencia.size();
  vector<EntInf> costesDePotencia(max + 1, Infinito);
  costesDePotencia[0] = 0;
  for (int i = 1; i <= N; i++) {
    for (int j = potencia[i - 1]; j <= max; j++) {
      costesDePotencia[j] = std::min(costesDePotencia[j], costesDePotencia[j - potencia[i -
    1]] + coste[i - 1]);
    }
  }

  int k = min;
  //
  int index = -1;
  EntInf menor = Infinito;
```

```
  while (k <= max) {
    if (costesDePotencia[k] < menor) {
      menor = costesDePotencia[k];
      index = k;
    }
    k++;
  }

  return { index, menor };
}

bool resuelveCaso() {
  // leemos la entrada
  int N, PMax, PMin;
  cin >> N >> PMax >> PMin;
  if (!cin)
    return false;

  // leemos las características de los tipos de bombillas
  vector<int> potencia(N); // 0-based
  for (int& x : potencia) cin >> x;
  vector<int> coste(N); // 0-based
  for (int& x : coste) cin >> x;

  // resolver el caso
  Sol sol = minCostLight(potencia, coste, PMin, PMax);
  //Escribir la sol
  if (sol.power != -1) std::cout << sol.costMin << "␣" << sol.power << "\n";
  else  std::cout << "IMPOSIBLE\n";

  return true;
}
```