**🧠 Concepto: Meet in the Middle**

**¿Qué es?**

Es una técnica de **dividir y conquistar**, usada cuando:

* El espacio de búsqueda es demasiado grande (por ejemplo 2n2^n2n),
* Pero puedes **dividirlo en dos mitades de tamaño n/2n/2n/2** y combinarlas.

En vez de probar todas las combinaciones posibles de nnn elementos (2n2^n2n), se generan todas las posibles combinaciones de las dos mitades (2n/22^{n/2}2n/2 cada una), y luego se usan **búsquedas binarias o hashing** para combinarlas.

**Usos típicos**

* Subconjuntos con suma objetivo (subset sum).
* Problemas de combinaciones, optimización, o conteo.
* Algunos problemas de grafos o rutas (con estado medio).

**⚙️ Esquema general**

// Ejemplo clásico: subset sum con Meet in the Middle

int n;

vector<int> a;

// Divides el arreglo en dos mitades

vector<int> left(a.begin(), a.begin() + n/2);

vector<int> right(a.begin() + n/2, a.end());

// Generas todas las sumas posibles de cada mitad

vector<long long> Lsums, Rsums;

for (int mask = 0; mask < (1 << left.size()); mask++) {

long long sum = 0;

for (int i = 0; i < left.size(); i++)

if (mask & (1 << i)) sum += left[i];

Lsums.push\_back(sum);

}

for (int mask = 0; mask < (1 << right.size()); mask++) {

long long sum = 0;

for (int i = 0; i < right.size(); i++)

if (mask & (1 << i)) sum += right[i];

Rsums.push\_back(sum);

}

// Luego combinas usando búsqueda binaria

sort(Rsums.begin(), Rsums.end());

long long target = k;

bool found = false;

for (auto x : Lsums) {

long long y = target - x;

if (binary\_search(Rsums.begin(), Rsums.end(), y))

found = true;

}

Complejidad: O(2n/2log⁡2n/2)O(2^{n/2} \log 2^{n/2})O(2n/2log2n/2) en lugar de O(2n)O(2^n)O(2n).  
Ideal para n≤40n \le 40n≤40.