给你一个 只包含正整数 的 非空 数组 nums 。请你判断是否可以将这个数组分割成两个子集，使得两个子集的元素和相等。（416）

分析：

1. 分成两个子集，和相等，那么数组的和一定能够被2整除；记target = sum / 2
2. 目标变为从一个数组中选中一组数使其值为target, 换个角度为：给定一组重量，一个背包可以放置target的重量，能否选择一组重量使得和为target,类似于0-1背包问题。
3. 和0-1背包的区别为：此处只能选择数据放入或不放入，无需计算价值，且最后重量一定等于target。

```
class Solution {
public:
  bool canPartition(std::vector<int>& nums) {
    int sum = getSum(nums);
    if (0 != sum % 2) {
      return false;
    }

    int                              target = sum / 2;
    std::vector<std::vector<bool>> dp =
        std::vector<std::vector<bool>>(nums.size() + 1,
                                       std::vector<bool>(target + 1,
false));
    // base case
    for (int i = 0; i <= nums.size(); i++) {
      dp[i][0] = true;
    }

    // dp 数组计算
    for (int i = 1; i <= nums.size(); i++) {
      for (int j = 1; j <= target; j++) {
        if (j - nums[i - 1] < 0) {  // 不能放下nums[i-1]
          dp[i][j] = dp[i - 1][j];
        } else {
          dp[i][j] = dp[i - 1][j] | dp[i - 1][j - nums[i - 1]];
        }
      }
    }

    return dp[nums.size()][target];
  }

private:
  int getSum(std::vector<int>& nums) {
    int sum = 0;
    for (auto item : nums) {
      sum += item;
    }
```

```
    return sum;
  }
};
```

完全背包问题

有n种重量和价值分别为wi和vi的物品。从这些物品中挑选总重量不超过W的物品，求出挑选物品价值总和的最大值，每种物品可以挑选多样。

分析：

1. 设dp[i][j]表示从前i个物品中选中最大重量为j时的最大价值。
2. 此时的dp计算公式为： dp[i][j] = std::max(dp[i][j], dp[i][j-k*w[i-1]]+k*v[i-1])

```
class Solution {
public:
  int CompletePackage(std::vector<int> &weight,
                      std::vector<int> &val,
                      int               w,
                      int               n) {
    std::vector<std::vector<int>> dp =
        std::vector<std::vector<int>>(n + 1, std::vector<int>(w + 1, 0));
    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= w; j++) {
        for (int k = 0; k * weight[i - 1] <= j; k++) {
          dp[i][j] =
              std::max(dp[i][j],
                       dp[i - 1][j - k * weight[i - 1]] + k * val[i - 1]);
        }
      }
    }

    return dp[n][w];
  }
};
```

多重部分和问题

有n种不同大小的数字ai，每个数字各有mi个。判断是否能从这些数字中挑选出若干使之和为K。

分析： 完全背包变形

```
class Solution {
public:
  bool MultiPartSum(std::vector<int> &nums, std::vector<int> &m, int n,
int K) {
    std::vector<std::vector<bool>> dp =
        std::vector<std::vector<bool>>(n + 1, std::vector<bool>(K + 1,
false));
    dp[0][0] = true;

    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= K; j++) {
        for (int k = 0; k <= m[i - 1] && k * nums[i - 1] <= j; k++) {
          dp[i][j] = dp[i][j] | dp[i - 1][j - nums[i - 1] * k];
        }
      }
    }

    return dp[n][K];
  }
};
```