# 背包问题

# 1、与价值无关的背包

- 从9件物品中选出3件，使其重量和500g之差的绝对值最小

- 用什么算法?

# 2、物品可分割

- 背包容量为**10kg**，现有**8**种不同的米粉可放入背包，每种米粉价值不同，求最多可放多少价值？

- 还记得**Fat Mouse**吗？

# 3、适配背包

- 背包容量S，现有n件物品，重量分别为w1,w2…wn。能否选择若干件放入背包，重量之和刚好为S

- 递归+穷举
- f(s,n)，返回值：0或者1
  - s表示背包剩余容量，n表示未考虑的物品件数（即物品编号为1，2，…，n），选取物品从n到1（方便递归）
- 递归关系
  - 不选择wn，则f(s,n)=f(s,n-1)
  - 选择wn，则f(s,n)=f(s-wn,n-1)
- 递归出口
  - s=0
  - s<0
  - s>0 && n<1

# 4、0-1背包

- 背包容量S，n个物品，第i个物品价值vi，求最大价值

- 递归
  - 选取第i件：f(s,i)=f(s-wi,i-1)+vi
  - 不选取：f(s,i)=f(s,i-1)
  - 取较大值为当前解

  - 算法复杂度O(2^n)
- 回溯法

- 动态规划
  - 当前背包容量<wi，不能选取i
  - 当前背包容量>=wi，可能选取i
- f[n][m]表示1～n件物品，m表示当前背包容量
- 从n开始
  - f[n][j]=0　　　j<wn
  - f[n][j]=vn　　　j>=wn

- 考虑n-1
  - 若j<w[n-1]，则不能选取n-1：f[n-1][j]=f[n][j]
  - 否则：
    - 选取n-1，则：f[n-1][j]=f[n][j-$w_{n-1}$]+$v_{n-1}$
    - 不选n-1，则：f[n-1][j]=f[n][j]
- 最终：f[1][S]
- 复杂度O(n*m)

# 举例：动规解决0-1背包

- S=116，n=3（背包总容量为116，3个物品）
  - w：100，14，10（每个物品的容量）
  - v：20，18，15（每个物品的价值）
- 初始：
  - f[n,j]=0　　　j<wn
  - f[n,j]=vn　　　j>=wn
- 递推：　f[n-1,j]=
  - f[n,j]　　$j<w_{n-1}$或者不选择n-1
  - $f[n,j-w_{n-1}] + v_{n-1}$　　　选择n-1

- 最终要求f[1,116] =38
  - =f[2,116] 不选择1 =33
  - =f[2,16] + 20 选择1 =38
- f[2,116]= =33
  - f[3,116] =15
  - f[3,102] + 18 =33
- f[2,16]= =18
  - f[3,16] =15
  - f[3,2] + 18 =18
- f[3,j]=
  - 0 j<10
  - 15 j>=10