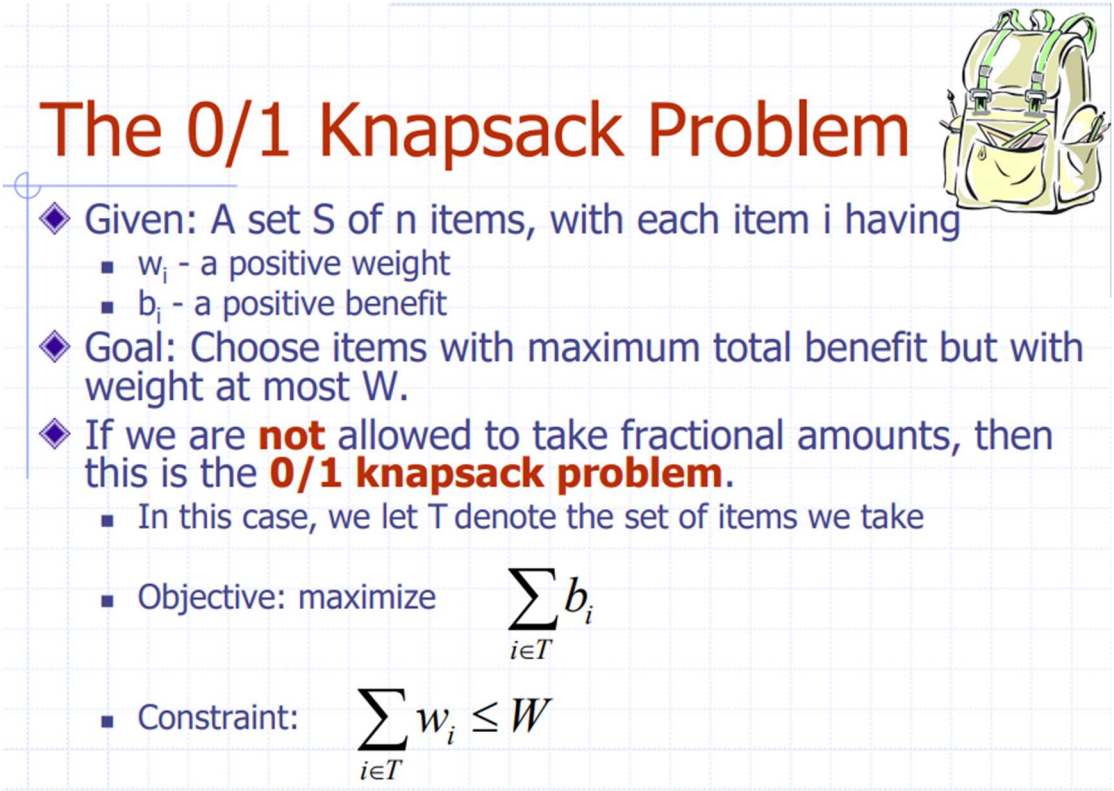
Dynamic Programming --- 0/1 Knapsack

Telescope Scheduling

Coin in a line

0/1 Knapsack

Explain the problem



And we are not allowed to take fractional amounts. Each item has a fixed weight and we cannot split it.

Give an example or given an instance of the problem, solve it

See iPad

Explain the algorithm and how are choices made and why, any important properties involved in making such a choice

0-1 Knapsack Algorithm from Lecture

Algo: 0-1 Knapsack (S, W)
Input: Set S of n ≥ 0 items, each with benefit b(i) ≥ 0 and weight w(i) ≥ 0, and knapsack capacity W. Assume w(i), n and W are integers and 0≤ i≤ n with i representing the i-*th* item
Output: Maximum benefit that can be obtained from S with a knapsack of capacity W

```
M[n, W] ← new n x W matrix
for i ← to n do
  for w ←0 to W // all possible residual capacities
    M[i, w] ← 0  // base case and initialization

for i ← 1 to n do
  for w ← 1 to W //assume weights are integers
    if w(i) ≤ w then
      M[i, w] ← b(i) + M[i-1, w - w(i)]
    M[i, w] ← max{ M[i, w], M[i-1, w] }
return M[n, W]
```

Is it greedy or dynamic or some other type of algorithm, explain why?

Dynamic programming is better than Greedy Algorithm for the "0/1 Knapsack" problem.

Reason:

DP can construct simple subproblems, create subproblem optimality, and solve the trouble of subproblem overlap. In this context, DP can globally explore all possible choices and consequences, ensuring a globally optimal solution.

While Greedy Algorithm seems faster, it may not guarantee global optimality, because it always does locally optimal choices without considering the global impact.

通用回答，仅需替换红色部分问题名字.

Compare/contrast the problems and algorithms

0/1: **We are not allowed to take fractional amounts from items. Each item has a fixed weight and we cannot split it.**

Given a scenario would you use 0-1 Knapsack or Fractional Knapsack, or

You give a scenario of when using 0-1 knapsack is appropriate and when Fractional is appropriate

见例题吧

Given Fractional Knapsack algorithm prove it is correct, that it returns an optimal choice of weights of item to be included in the knapsack

见 Fraction Knapsack