Knapsack problem with DP

Knapsack problem

- Part1: Typical
- Part2: Unbounded
- Part3: With the limited number of each element

Part1

You have N items that you want to put them into a knapsack. Item i has value v_i and weight w_i.

You want to find a subset of items to put such that:

- The total value of the items is as large as possible.
- The items have combined weight at most W, that is capacity of the knapsack.

Find the maximum total value of items in the knapsack.

Source: AOJ DPL1B(http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=DPL_1_B)

Constraints (1) 1/3

In many problems, the solution depends on constraints.

1. Typical

- $1 \leq N \leq 10^2$
- $1 \leq v_i \leq 10^3$
- $1 \leq w_i \leq 10^3$
- $1 \leq W \leq 10^4$

Constraints (1) 2/3

- Def.

$$\begin{aligned} dp[i][j] := &\text{ the maximum value with } 0, 1, \ldots, i-1 \text{ th items,} \\ &\text{subject to } j \text{ capacity of knapsack} \end{aligned}$$

- Init.
  - $dp[i][j] = 0$

Constraints (1) 3/3

- Trans.

  We know the maximum value with 0 to $i-1$ items subject to j capacity.

  If ith item is not used, value and weight are not be changed.

  If ith item is used, value increases $v[i]$, and weight increases $w[i]$.

  - $dp[i+1][j] = \max(src, dp[i][j])$
  - $dp[i+1][j+w[i]] = \max(src, dp[i][j] + v[i])$

- Ans.

  $dp[N][W]$

Constraints (2) 1/4

2. w_i is huge but v_i is small

- $1 \leq N \leq 10^2$
- $1 \leq v_i \leq 10^2$
- $1 \leq w_i \leq 10^7$
- $1 \leq W \leq 10^9$

Constraints (2) 2/4

Bad solution

Let definition is this:

$$dp[i][j] := \text{the maximum value with } 0, 1, \ldots, i-1 \text{ th items,} \\ \text{subject to } j \text{ capacity of knapsack}$$

But we cannnot make that array because of memory (the size of $dp[110][1100000000]$ is too large.)

So instead of having weight as index, let dp table have value as index.

Constraints (2) 3/4

Good solution

- Def.

$$\begin{aligned} dp[i][j] := &\text{ the minimum weight with } 0, 1, \ldots, i-1 \text{ th items,} \\ &\text{subject to } j \text{ value} \end{aligned}$$

- Init.
  - $dp[i][j] = 0$
  - $\text{others} = \text{INF}$

Constraints (2) 4/4

- Trans.

  We know the minimum weight with 0 to $i-1$ items subject to j value.

  If ith item is not used, value and weight are not be changed.

  If ith item is used, value increases $v[i]$, and weight increases $w[i]$.

  - $dp[i+1][j] = \min(src, dp[i][j])$
  - $dp[i+1][j+v[i]] = \min(src, dp[i][j] + w[i])$

- Ans.

  maximum of j such that $dp[N][j] \leq W$

Constraints (3)

Called huge knapsack problem.

3. Both w_i and v_i are huge but N is small

- $1 \leq N \leq 40$
- $1 \leq v_i \leq 10^{15}$
- $1 \leq w_i \leq 10^{15}$
- $1 \leq W \leq 10^{15}$

This problems is solved with "Split and List(半分全列挙)", which is not DP and out of this slide.

Part2 1/4

You have N kinds of items that you want to put them into a knapsack. Item i has value v_i and weight w_i.

You want to find a subset of items to put such that:

The total value of the items is as large as possible.
The items have combined weight at most W, that is capacity of the knapsack.
You can select **as many items as possible** into a knapsack for each kind.
Find the maximum total value of items in the knapsack.

Part2 2/4

Constraints

- $1 \leq N \leq 10^2$
- $1 \leq v_i \leq 10^3$
- $1 \leq w_i \leq 10^3$
- $1 \leq W \leq 10^4$

Source: AOJ DPL1C (http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=DPL_1_C)

Part2 3/4

- Def.

$$dp[i][j] := \text{the maximum value with } 0, 1, \ldots, i-1 \text{ th items,} \\ \text{subject to } j \text{ capacity of knapsack}$$

- Init.
  - $dp[i][j] = 0$

Part2 4/4

- Trans.
  We know the maximum value with 0 to $i-1$ items subject to j capacity.
  If ith item is not used, value and weight are not be changed.
  If ith item is used one time or more than two times, value increases $v[i]$, and weight increases $w[i]$.
    - $dp[i+1][j] = \max(src, dp[i][j])$
    - $dp[i+1][j+w[i]] = \max(src, dp[i][j] + v[i])$
    - $dp[i+1][j+w[i]] = \max(src, dp[i+1][j+w[i]] + v[i])$
- Ans.
  $dp[N][W]$

Part3 1/9

You have N items that you want to put them into a knapsack. Item i has value v_i, weight w_i and limitation m_i.

You want to find a subset of items to put such that:

- The total value of the items is as large as possible.
- The items have combined weight at most W, that is capacity of the knapsack.
- You can select at most m_i items for ith item.

Find the maximum total value of items in the knapsack.

Part3 2/9

Constraints

- $1 \leq N \leq 10^2$
- $1 \leq v_i \leq 10^3$
- $1 \leq w_i \leq 10^3$
- $1 \leq m_i \leq 10^4$
- $1 \leq W \leq 10^4$

Source: AOJ DPL1G (http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=DPL_1_G)

Part3 3/9

Bad solution

- Def.

$$
\begin{aligned}
dp[i][j] := &\text{ the maximum value with } 0, 1, \ldots, i-1 \text{ th items,} \\
&\text{ subject to } j \text{ capacity of knapsack}
\end{aligned}
$$

- Init.
    - $dp[i][j] = 0$

Part3 4/9

- Trans.

  We know the maximum value with 0 to $i-1$ items subject to j capacity.

  For each $k = 0, 1, \ldots, m[i]$, you can choose ith item k times:
  - $dp[i+1][j+k \cdot w[i]] = \max(src, dp[i][j] + k \cdot v[i])$
- Ans.

  $dp[N][W]$

But the time complexity is $O(NW \max_i \{m_i\})$

Part3 5/9

Before solution...

Note that fact:

$$
\begin{aligned}
1 + 2 + 2^2 + \cdots + 2^k &= \frac{2^{k+1} - 1}{2 - 1} \\
&= 2^{k+1} - 1
\end{aligned}
$$

Part3 6/9

If the maximum value k such that

$$
\begin{aligned}
X &= 1 + 2 + 2^2 + \cdots + 2^k + a \\
&= b + a
\end{aligned}
$$

then a satisfies $0 < a < 2^{k+1}$, and $b = 2^{k+1} - 1$.

- A number y such that $0 < y < 2^{k+1}$ is expressed with $\{1, 2, 2^2, \ldots 2^k\}$ (understand base-2 number).
- $0 < a < 2^{k+1}$

⇒ Any number x such that $0 < x < X$ is expressed with $\{1, 2, 2^2, \ldots, 2^k, a\}$

Part3 7/9

Program to print $1, 2, 2^2, \ldots, 2^k, a$

```
int x;
cin >> x;
for (int i = 0; x > 0; i++) {
    int t = min(1LL<<i, x);
    cout << t << ' ';
    x -= t;
}
cout << endl;
```

Part3 8/9

```
100 (input)

1 2 4 8 16 32 37
```

Part3 9/9

In concludion, you can consider N items as $\Sigma_{i=1}^{N} \log m_i$ items:

$i\text{th item } (v_i, w_i) \rightarrow \{(v_i, w_i), (2v_i, 2w_i), (2^2 v_i, 2^2 w_i), \ldots, (2^k v_i, 2^k w_i), (av_i, aw_i)\}$

We can only solve the knapsack problem of $\Sigma_{i=1}^{N} \log m_i$ items.

The time complexity is $O(N \max_i \{\log m_i\} W)$