Dynamic Planning

Knapsack

例題 | 参考

n 個の品物があり，i 番目の品物のそれぞれ重さと価値が w[i]、v[i] となっている. これらの品物から重さの総和が W を超えないように選んだときの、価値の総和の最大値を求める．

方針

```
dp[i+1][j]  >>>   i 番目までの品物の中で重さの総和が j 以下となるように選んだときの
最大価値
```

漸化式

```
if j >= w[i] then
  dp[i+1][j] = max{ dp[i][j - w[i]] + v[i], dp[i][j] }
else
  dp[i+1][j] = dp[i][j]
```

初期条件

```
REP(j, 0, W)
  dp[0][j] = 0
```

Number Partitioning 1

例題 | 参考

n 個の正の整数 a[0], a[1], …, a[n−1] の部分和が整数 A になるものはあるか．

方針

```
dp[i+1][j]  >>>   i番目までの整数の中からいくつか選んで総和をjとすることが可能かど
うか
```

漸化式

```
if j >= a[i] then
  dp[i+1][j] = dp[i][j-a[i]] | dp[i][j]
else
  dp[i+1][j] = dp[i][j]
```

初期条件

```
REP(j, 0, A)
  dp[0][j] = (j==0) ? True : False
```

Number Partitioning 2　(部分和数え上げ)

例題 | 参考

n 個の正の整数 a[0], a[1], …, a[n−1] から何個か選んで，総和を整数 A にする方法は何通りあるか．(答えは mod 1e9+7)

方針

```
dp[i+1][j]  >>>   i番目までの整数からいくつか選んで総和をjとする場合の数
```

漸化式

```
if j >= a[i] then
  dp[i+1][j] += dp[i][j - a[i]]

dp[i+1][j] += dp[i][j]
dp[i+1][j] %= MOD;
```

初期条件

```
REP(j, 0, A)
  dp[0][j] = (j==0) ? 1 : 0;
```

Number Partitioning 3（最小個数部分和）

例題 | 参考

n 個の整数 a[0], a[1], …, a[n−1] から何個か選んで総和を A にする方法を考えたとき，選ぶ整数の個数の最小値はどうなるか．（できない場合も有り）

方針

```
  dp[i+1][j]  >>>   i番目までの整数からいくつか選んで総和をjとするときの選んだ整数の最小個数
```

漸化式

```
  if j >= a[i] then
    dp[i+1][j] = min{ dp[i][j - a[i]], dp[i][j] }
  else
    dp[i+1][j] = dp[i][j]
```

初期条件

```
  REP(j, 0, A)
    dp[0][j] = (j==0) ? 0 : infinity
```

Number Paritioning 4（K個以内部分和）

例題 | 参考

n 個の整数 a[0], a[1], …, a[n−1] からK個以内の個数を選んで総和を A にすることは可能かどうか．

方針

```
  dp[i+1][j][k] >>>   i番目までの整数からk個の整数を選んで総和をjにすることは可能かどうか
```

漸化式

```
  if j >= a[i] then
    if k + a[i] <= k then
      dp[i+1][j][k] = dp[i][j - a[i]][k-1] | dp[i][j][k]
```

```
dp[i+1][j][k] |= dp[i][j][k]
```

初期条件

```
REP(j, 0, A)
  dp[0][j][0] = (j==0) ? True : False
```

しかし，これだと計算量が O(nKA) となるため微妙．そこで，Nubmer Partitioning 1 の結果において

```
(dp[n][A] <= k) ? yes : no < pre>
```

と表すことができる．これなら計算量は O(nA) で済む．

Number Partitioning 5（個数制限付き部分和）

例題 | 参考

方針

```

```

漸化式

```

```

初期条件

```

```