**İki Silo Çözüm**

2 boyutlu Dynamic Programming ile soru çözülebilir.

dp[i][j] = 1 değeri

birinci silo j kilo dolu olduğunda i tane kamyon boşalmıştır demek olsun.

Eğer ilk i kamyonun toplam ağırlığı sum[i] ise dp[i][j]'de birinci silo j kilo dolu olduğundan ikinci silo sum[i]-j kadar dolu demektir.

dp[i][j] = 0 böyle bir durumun yani birinci silo j kilo dolu iken i tane kamyonun boşaltılmış olmasının mümkün olmadığını göstermektedir.

N bir silonun kapasitesi olsun ve K[i] de i. kamyonun ağırlığı olsun. 0 <= j <= N için Dynamic Programming reccurrence'i şu şekilde olur:

dp[i][j] = (j>=K[i] && dp[i-1][j-K[i]]) || (sum[i]-j<=N && dp[i-1][j])

OR'un sol kısmı birinci siloya koymayı, sağ kısmı da ikinci siloya koymayı gösterir.

Çözüm kodu aşağıda verilmiştir:

```
#include <iostream>
#include <vector>
#include <stack>

using namespace std;

const int MaxSilo = 101 * 1000;
const int MaxKamyon = 202;

vector<int> kamyonlar;
bool birOncekiBirinciSiloya[MaxKamyon][MaxSilo];

bool agirlikUlasilabilir[2][MaxSilo];

int main()
{
	kamyonlar.clear();
	int sonBosaltilan = 0;
	int N;
	int M;
	cin >> N;
```

```
cin >> M;
N*= 1000;

int toplamBosalan = 0;
int kamyonSayisi = 0;
int birOnceki;
int Ki;
bool mumkun = true;

agirlikUlasilabilir[sonBosaltilan][0] = true;
for (int i = 1; i <= N; ++i)
   agirlikUlasilabilir[sonBosaltilan][i] = false;
for (int k = 1; k <= M; k++)
{
   cin >> Ki;
   if (mumkun)
   {
      kamyonlar.push_back(Ki);
      const int birSonrakiBosalacak = 1 - sonBosaltilan;
      for (int i = 0; i <= N; ++i)
         agirlikUlasilabilir[birSonrakiBosalacak][i] = false;

      toplamBosalan += Ki;
      bool bosalttik = false;
      for (int i = 0; i <= N; ++i)
         if (agirlikUlasilabilir[sonBosaltilan][i])
         {
            if (i + Ki <= N)
            {
               bosalttik = true;
               birOnceki = i + Ki;
               birOncekiBirinciSiloya[kamyonSayisi + 1][i + Ki] = true;
               agirlikUlasilabilir[birSonrakiBosalacak][i + Ki] = true;
            }
            if (toplamBosalan - i <= N && kamyonSayisi != 0)
            {
               bosalttik = true;
               birOnceki = i;
               birOncekiBirinciSiloya[kamyonSayisi + 1][i] = false;
               agirlikUlasilabilir[birSonrakiBosalacak][i] = true;
            }
         }
      mumkun = bosalttik;
      if (mumkun)
```

```
            ++kamyonSayisi;
          sonBosaltilan = birSonrakiBosalacak;
        }
      }

      cout << kamyonSayisi << '\n';
      stack<string> out;

      int siloBir = 0, siloIki = 0;
      for (; kamyonSayisi > 0; --kamyonSayisi)
      {
        if (birOncekiBirinciSiloya[kamyonSayisi][birOnceki])
        {
          out.push("1");
          birOnceki -= kamyonlar[kamyonSayisi - 1];
          siloBir += kamyonlar[kamyonSayisi - 1];
        }
        else
        {
          out.push("2");
          siloIki += kamyonlar[kamyonSayisi - 1];
        }
      }

      while (!out.empty())
      {
        cout << out.top() << '\n';
        out.pop();
      }
}
```