一种方法是０１背包，参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | //货币面值 -01背包  #include <iostream>  #include <cstdlib>  using namespace std;    int N,MAX;  int value[101],dp[10001];    int Compare(const void \* p, const void \* q)  {  return \*(int \*)p - \*(int \*)q;  }    int Max(int a, int b)  {  return (a > b) ? a : b;  }    int ZeroOnePack()  {  int i, j;  memset(dp, 0, sizeof(dp));  for (i = 1; i <= N; ++i)  for (j = MAX; j >= value[i]; --j)  dp[j] = Max(dp[j], dp[j-value[i]] + value[i]);  for (i = 1; i <= MAX; ++i)  if (dp[i] != i)  return i;  }    int main()  {  freopen("Currency.in","r",stdin);  freopen("Currency.out","w",stdout);  int i;  while (scanf("%d", &N) != EOF)  {  MAX = 0;  for (i = 1; i <= N; ++i)  {  scanf("%d", &value[i]);  MAX += value[i];  }  qsort(value, N, sizeof(int), Compare);  cout<<ZeroOnePack()<<endl;  }  return 0;  } |

使用动态规划的思想, 对于从第1个到第i个数的和total，如果第i+1个数大于total+1则不会组成total+1。

具体来讲，就是计算已排好序的数据，计算到第i个的时候，total记录的是前面i – 1个数的和。那么第i个数如果大于total + 1，而第i个数后面的数已知都比第i个数大，都不可能构成total + 1，那么对于读到的这个i，显然total + 1是最小的无法构成的数。

再来看更普遍的：其实对于一个范围内的数，如果要用最少的数构成所有这个范围内的数，只要满足得到的数都是关于2的幂次,比如要构成16以内所有的数，只用1，2，4，8便足够了，这是很容易证明的，因为所有的数都可以表示成若干个2次幂数的和，其中构成数的最大2次幂数一定小于这个数，所以得证。

知道了以上这个规律，再观察2次幂数的特点， 对于1，2，4，8，16 这样的数列，会发现前i – 1项的和都恰好比第i项小1，那么，对于一个有序数列，如果满足上述这个规律，或者更普遍点，不用正好小1，小更多，只要是前提已经是有序，那么都会有这个数列的特性，即构成value[i]范围内所有的数，最小不能构成的数就是total + 1。

参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | //货币面值－动规２  #include <stdio.h>  #include <stdlib.h>  int Compare(const void \* p, const void \* q)  {  return \*(int \*)p - \*(int \*)q;  }  int main()  {  freopen("Currency.in","r",stdin);  freopen("Currency.out","w",stdout);  int N,i,ans,value[100];  while (scanf("%d", &N) != EOF)  {  for (i = 0; i < N; ++i)  scanf("%d", &value[i]);  qsort(value, N, sizeof(int), Compare);  ans = 0;  for (i = 0; i < N; ++i)  if (value[i] > ans + 1)  break;  else  ans += value[i];  printf("%d\n", ans + 1);  }  return 0;  } |