

William Randall

40567986

HW4

Problem. On the planet Sigma, robots excavate chunks of a very precious cristaline. These chunks can be divided into smaller part only on the Earth. Once a month, a spaceship comes to Sigma to bring cristaline to the Earth. The spaceship cannot take more than k pounds of the load. Naturally, it is desirable to take as much cristaline as the spaceship can.
(100 pts) Write an efficient algorithm that will allow bringing the maximal amount of cristaline to the Earth. Prove that it is correct and estimate its time complexity.

```
initialize 2d array T [amount of chunks + 1][k + 1] // True if subsequence w given sum
let chunks[] be an ordered array of available chunk sizes
possible_ship:
  for i = 1 to k:
    | T[i][0] = False // no items in list & sum != 0
  end for
  for i = 0 to amount of chunks:
    | T[i][0] = True // sum is 0 then True
  end for
  for i = 1 to amount of chunks:
    | for j = 1 to k:
      | if chunks[i-1] > j:
      | | T[i][j] = T[i-1][j] // do not include if chunk size > weight ind
      | else
      | | T[i][j] = T[i-1][j] or T[i-1][j - chunks[i-1]] // include or exclude
      | endif
    | end for
  end for

  let weight_ind = sum
  while weight_ind != 0: // find weight index that is True
    | if T[amount of chunks][weight_ind] is True then break from while
    | weight_ind --
    | if weight_ind is 0 then return null set
  end while
```

Continue
↓



initialize list of best chunks, BC

Let chunk_ind be amount of chunks

While weight_ind != 0 // make the list of available chunks

If $T[\text{chunk_ind} - 1][\text{weight_ind}]$ is True then chunk_ind--

else

Add chunks[chunk_ind] to BC

chunk_ind--

weight_ind = weight_ind - chunks[chunk_ind]

endif

end while

Return BC

Time complexity for this will be $O(\text{amount of chunks} \cdot k)$ because filling out the array T will take the longest amount of time for this algorithm.

This algorithm will return a set of the chunks which will result in the maximum weight which is below k.

First we make T which is a 2d array of bools which is true if there exists a subsequence of chunks with given sum.

For the column with sum=0 it is all true.

For no items in the list it is all false.

Then it loops through the T array and if the chunk size is greater than the current k then it will not include said chunk.

Otherwise it will try to include + exclude the chunk + take the or of those two values. Once the T array is complete it finds the largest k value with a true value in the T array

↓ Cont

at the index of the amount of Chunks. Then from that position it will look at the value above it in array T and if said value is true then that chunk is not included in the final set, but if it is not true then you include that chunk and you move lower in the k values (aka decrease your weight index by the weight amount of the chunk you just added to the list). It does this until it reaches a sum of 0 then returns the list.

