Intermediate Math Circles

October 21, 2020

Sunken Treasure - Solutions

The Centre for Education in Mathematics and Computing
Faculty of Mathematics, University of Waterloo

www.cemc.uwaterloo.ca

Problem Set #1

Question:

What is the maximum value of items you can achieve?

Answer:

For the first dive, the maximum value is 2900 gold.

For the second dive, the maximum value is $11\,550$ gold.

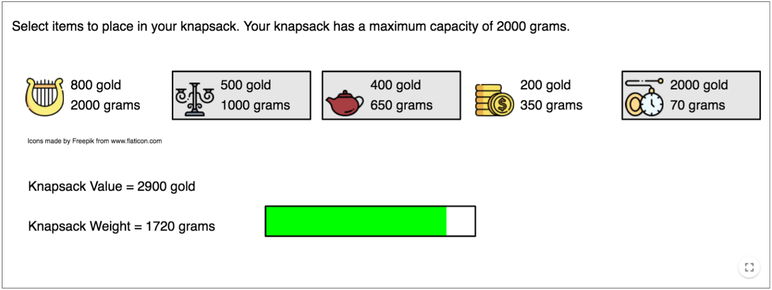
Problem Set #2

Question:

Which subset of items achieves this maximum value?

Answer:

For the first dive, the maximum value of 2900 gold is achieved by selecting the candlestick, teapot, and pocket watch.



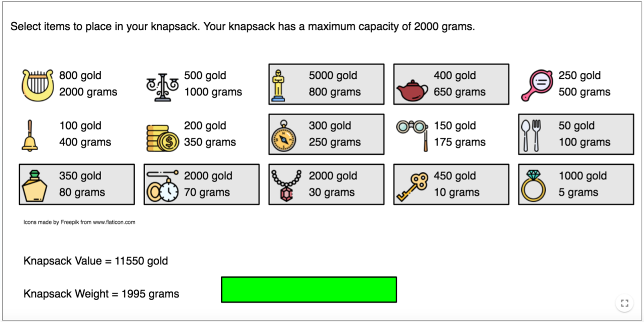
Problem Set #2

Question:

Which subset of items achieves this maximum value?

Answer:

For the second dive, the maximum value of $11\,550$ gold is achieved by selecting the perfume, pocket watch, statue, compass, necklace, teapot, key, cutlery, and ring.



Problem Set #3

Question:

What was your process?

Answer:

For the first dive, where the number of items was small, you may have used some trial and error or logical reasoning to come up with the optimal solution.

The second dive required a more formal plan of attack. Did you choose the most valuable items first? Maybe you chose the lightest items first?

A very common approach is to calculate each item's rate of *gold per gram*, sort them from largest to smallest rate, and then select items in this same order as long as they fit in the knapsack. This algorithm, where you choose the best item at each step in the hope of getting the best outcome overall, is known as a *greedy algorithm*.

Greedy algorithms are easy to implement, and produce good solutions, but they don't always yield the best solutions. If you used this algorithm you likely achieved a maximum value of $11\,500$ gold which is 50 gold shy of optimal.

Problem Set #4

Question:

Can you guarantee or prove that your subset is optimal? If so, how?

Answer:

The best known way to guarantee that a subset is optimal is to check the value of every possible subset. This is known as a *brute force algorithm*.

When forming a subset, each item is either in the subset or not. So there are 2 possibilities for each of the 15 available items: in or out. This means there are $2^{15} = 32\,768$ subsets.

Checking the values of all these subsets by hand is not feasible, but suppose a computer can check one million subsets every second. In this case, it would take a computer less than one second to check all the subsets and find the optimal solution.

Easy peasy? Not really. This algorithm does not scale well. If the number of items is increased from 15 to 100, there would be 2^{100} subsets to check which would take a computer approximately 40 quadrillion years!