Assignment #1: Brute Force Set Cover

Notes and Explanation

CS 224

Jason Hibbeler

University of Vermont

Fall 2022

The `permute()` Function

The function `permute(int n)` will return an `ArrayList` consisting of 2^n arrays

- each array will be of length n and will have a sequence of `true` and `false` values
- the `ArrayList` will contain all possible unique such arrays

people who understand Java better than I do say that we should use `List<>` instead of `ArrayList<>`

Example

`permute(1)` will return a list consisting of these two arrays:

```
[false]
[true]
```

Note: the base case `permute(0)` will return a list consisting of the array `[]`

Example

To form `permute(2)`, use the return value from `permute(1)`

- for each element in `permute(1)`, form two new arrays
- one of the arrays has the element along with `true`
- and the other array has the element along with `false`

Like this:

$$\begin{tikzcd}
\text{return value from } \texttt{permute(1)} & \text{return value from } \texttt{permute(2)} \\
\texttt{[false]} \arrow[r] \arrow[dr] & \texttt{[false, false]} \\
 & \texttt{[false, true]} \\
\texttt{[true]} \arrow[r] \arrow[dr] & \texttt{[true, false]} \\
 & \texttt{[true, true]}
\end{tikzcd}$$

Example

To form `permute(3)`, use the return value from `permute(2)`

$$\begin{tikzcd}[column sep=large]
\text{return value from } \texttt{permute(2)} & \text{return value from } \texttt{permute(3)} & \text{etc.} \\
\texttt{[false, false]} \arrow[r] \arrow[dr] & \texttt{[false, false, false]} & \\
 & \texttt{[false, false, true]} & \\
\texttt{[false, true]} \arrow[r] \arrow[dr] & \texttt{[true, true, false]} & \\
 & \texttt{[true, true, true]} & \\
\texttt{[true, false]} \arrow[r] \arrow[dr] & \texttt{[true, false, false]} & \\
 & \texttt{[true, false, true]} & \\
\texttt{[true, true]} \arrow[r] \arrow[dr] & \texttt{[true, true, false]} & \\
 & \texttt{[true, true, true]} &
\end{tikzcd}$$

Using the Permutations

I will create a single `ArrayList` of all 2^n possible `true`/`false` combinations

Then, I'll treat each element of this array as a "what-if" scenario

- for example: `[true, true, false, true, false, true, true, true, false, false]` says "include the instructors #1, #2, #4, #6, #7, #8"
- with this subset of teachers, see how many of the courses are covered
- if all of the courses are covered, then this subset represents a set cover

For each subset that does represent a set cover, count how many elements it has

- in other words, how many `true` values the subset contains

And then keep track of the covering subset having the fewest number of elements!