Rewriting the bag graph as a matrix

Let's start with some example input:

- A red bag contains 2 blue bags and 1 green bag.
- A blue bag contains 3 yellow bags and 5 purple bag.
- A green bag contains 7 purple bags.
- A yellow bag contains no other bags.
- A purple bag contains no other bags.

We first need to map (in my code this is a dictionary) colours to numbers, so we pick red: 1, blue: 2, green: 3, yellow: 4, and purple: 5.

Rewriting the bag graph as a matrix

We can then rewrite the above as a matrix B, where the elements B_{ij} are given by the number of bag is contained in bag j. With this example, we then see $B_{21} = 2$, $B_{31} = 1$, etc. Continuing in this vein we construct the matrix B:

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 5 & 7 & 0 & 0 \end{pmatrix}$$

...Why?

Obvious question is why bother?

Needlessly mathematical sentence: “Matrices can be thought of as representations of linear functions on abstract vector spaces.“

...let’s unpack that.

With our graph written as a matrix B, we can think of B as an ‘opening’ operation on whatever bags we happen to have. We open all our bags, and get the new bags that were inside the old bags...and throw the now opened old bags away.

...Why?

Going back to our example, let's say we have 1 red bag and 3 green bags. With our map from colours to numbers above, we can write this as a column vector $(1, 0, 3, 0, 0)^T$. (That T means 'transpose' and it just saves space on the slide.) We can then apply B to this vector with the standard rules of matrix multiplication:

$$B\begin{pmatrix}1\\0\\3\\0\\0\end{pmatrix} = \begin{pmatrix}0 & 0 & 0 & 0 & 0\\2 & 0 & 0 & 0 & 0\\1 & 0 & 0 & 0 & 0\\0 & 3 & 0 & 0 & 0\\0 & 5 & 7 & 0 & 0\end{pmatrix}\begin{pmatrix}1\\0\\3\\0\\0\end{pmatrix} = \begin{pmatrix}0\\2\\0\\0\\21\end{pmatrix}$$

..so we now have 2 bag 2s and 21 bag 5s, i.e. 2 blue bags, 21 purple bags. We've discarded the red and green bags we had.

Yeah, but... why?

We can repeat this as many times as we like, opening our bags k times is given by the matrix B^k. These matrices B^k contain all the information we need.

- The number of non-zero elements in row i of B^k tells us how many bags contain bag i after k openings. Looking at our B, we see row 5 has two non-zero elements; purple bags are contained by 2 other bags, blue and green.
- The sum of column i of B^k tells us the number of bags that bag i contains. Again, column 2 of our B tells us that blue bags contain 8 bags - 3 yellow and 5 purple. We knew that anyway.

It turns out this is just what we need!

Actually solving the puzzle

Now we've set up all this machinery, let's restate the problem parts 1 and 2:

- Part 1: How many bags contain at least one bag n?
- Part 2: How many bags does bag n contain?

Bag n here is a shiny gold bag in our puzzle - but it could be anything. We have a lot of counting to do.

It's like Christmas morning

To solve parts 1 and 2, we need to open some bags! But how many times should we open? As much as we can! For part 1, we need *all* the bags containing one bag n, so we need to open bags (compute B), add the non-zero elements of row n, open again (compute B^2), add the non-zero elements of row n, and so on, and so on. Basically we want the number of non-zero elements of row n of the matrix

$$B + B^2 + B^3 + \ldots$$

Part 2 comes for free - it's just the sum of column n of this matrix.

Wait, isn't that infinite?

I haven't told you when to stop when computing $B + B^2 + B^3 + \ldots$
Fair point.
Luckily, if B represents a 'physical' system of bags, we know we have to stop eventually - we don't have bags that contain themselves. Eventually $B^k = 0$ (a matrix of zeroes) for some positive integer k. In maths-speak, B is a *nilpotent* matrix.
Of course, that doesn't tell us when to stop, it just tells us that we will :D

Maths is just applied laziness

Another problem is that we have like, 600 bags, and matrix multiplication is *hard work*. Computing that sum directly, especially when we don't know when to stop, is a right pain.

We can, however, use the fact that we know that $B^k = 0$ for some k to make our lives easier. If we let

$$\begin{aligned} M = B+ \quad & B^2 + B^3 + ... + B^{k-1} \\ \implies BM = \quad & B^2 + B^3 + ... + B^{k-1} + B^k \end{aligned}$$

...then

$$(I - B)M = B \implies M = (I - B)^{-1}B = -I + (I - B)^{-1}$$

where I is the identity matrix and $(I - B)^{-1}$ is the matrix *inverse* of $I - B$.

Finally done.

...and we're finished! The number of non-zero elements of row n of $M = -I + (I - B)^{-1}$ tells us how many bags contain at least one bag n (part 1). The sum of column n of M tells us how many bags bag n contains (part 2). All for the price of one matrix inverse (which is expensive, but easier than repeated multiplication an arbitrary number of times.)

Full confession - this is much slower than Tom's solution; about 500 times slower. I think this is because we're effectively solving all the problems at once, not just for the shiny gold bag. All you have to do is read the right row/column of M, once you've computed it.

There's no kill like overkill.