07

December 7, 2020

1 Day 7: Handy Haversacks

Zach Bogart

12/07/2020

- https://adventofcode.com/2020/day/7

1.1 Setup

[1]:
```
import pandas as pd
import re
```

[2]:
```
with open("inputs/07-input.txt") as f:
    raw = f.readlines()
```

[3]:
```
raw = [re.sub(" bag[s]?", "", bag) for bag in raw]
```

[4]:
```
raw[:10]
```

[4]:
```
['posh blue contain 5 plaid chartreuse, 3 plaid lime.\n',
 'clear teal contain 2 dotted salmon, 2 wavy red.\n',
 'faded blue contain 1 dotted chartreuse, 3 dim bronze.\n',
 'plaid black contain 5 muted beige, 2 pale gold, 3 wavy lavender, 5 dull
yellow.\n',
 'bright cyan contain 2 vibrant teal.\n',
 'clear magenta contain 2 dim chartreuse.\n',
 'muted crimson contain 1 clear violet, 5 dark coral, 1 pale salmon, 3 light
red.\n',
 'dotted green contain 3 muted plum.\n',
 'pale crimson contain 3 pale maroon, 2 mirrored tan.\n',
 'shiny black contain 1 wavy tomato.\n']
```

[5]:
```
bags_nested = [line[:-2].split(" contain ") for line in raw]
```

[6]:
```
bags_nested = [[re.sub("[\d]+ ", "", item) for item in bag] for bag in␣
↪bags_nested]
```

[7]:
```
bags = {bag[0]: bag[1].split(", ") for bag in bags_nested}
```

[8]:
```
{key: bags[key] for key in list(bags.keys())[:10]}
```

[8]:
```
{'posh blue': ['plaid chartreuse', 'plaid lime'],
 'clear teal': ['dotted salmon', 'wavy red'],
 'faded blue': ['dotted chartreuse', 'dim bronze'],
 'plaid black': ['muted beige', 'pale gold', 'wavy lavender', 'dull yellow'],
 'bright cyan': ['vibrant teal'],
 'clear magenta': ['dim chartreuse'],
 'muted crimson': ['clear violet', 'dark coral', 'pale salmon', 'light red'],
 'dotted green': ['muted plum'],
 'pale crimson': ['pale maroon', 'mirrored tan'],
 'shiny black': ['wavy tomato']}
```

1.2 Part 1

[9]:
```python
# for pretty printing
# via https://stackoverflow.com/questions/287871/
 ↪how-to-print-colored-text-in-python
class bcolors:
    WARNING = '\033[93m'
    ENDC = '\033[0m'
```

[10]:
```python
def look_in_the_bag(bag, bag_dict, show_me=False):
    if show_me: print(f"{bag} --> ", end="")
    if 'no other' in bag_dict[bag]:
        if show_me: print("no other")
        return False
    if 'shiny gold' in bag_dict[bag]:
        if show_me: print(f"{bcolors.WARNING}shiny gold{bcolors.ENDC}")
        return True
    else:
        for bag in bag_dict[bag]:
            if look_in_the_bag(bag, bag_dict, show_me) == 1:
                return True
        return False
```

[11]:
```python
look_in_the_bag("dim tan", bags, show_me=True)
```

```
dim tan --> plaid bronze --> posh lime --> shiny tomato --> no other
dark brown --> muted turquoise --> no other
plaid white --> no other
bright tomato --> dark brown --> muted turquoise --> no other
plaid white --> no other
pale salmon --> shiny orange --> no other
dark brown --> muted turquoise --> no other
plaid white --> no other
dark coral --> shiny bronze --> no other
shiny tomato --> no other
bright tomato --> dark brown --> muted turquoise --> no other
```

```
plaid white --> no other
plaid white --> no other
faded red --> no other
pale magenta --> shiny gold
```

[11]:
```
True
```

[12]:
```
shiny_gold_inside = {}
for bag in bags:
    shiny_gold_inside[bag] = look_in_the_bag(bag, bags)
```

[13]:
```
sum(shiny_gold_inside.values())
```

[13]:
```
142
```

1.3 Part 2

[14]:
```
bags_nested_2 = [line[:-2].split(" contain ") for line in raw]
```

[15]:
```
bags_2 = {bag[0]: bag[1].split(", ") for bag in bags_nested_2}
```

[16]:
```
bags_2["posh blue"]
```

[16]:
```
['5 plaid chartreuse', '3 plaid lime']
```

[17]:
```
def repeat_and_flatten(bag):
    nested =[]
    for item in bag:
        if item == "no other":
            return bag
        else:
            contents = re.search("([\d]+ )(.*)", item)
            repeat_count = int(contents.group(1))
            color = contents.group(2)

            repeated = [color] * repeat_count

            nested.append(repeated)

            flattened = [val for sub_list in nested for val in sub_list]

    return flattened
```

[18]:
```
bags_repeated = {key:repeat_and_flatten(val) for key, val in bags_2.items()}
```

[21]:
```
# count up every nested bag
def all_bags_unpacked(bag, bag_dict, show_me=False):
    if show_me: print(f"{bag} --> ", end="")
    if 'no other' in bag_dict[bag]:
        if show_me: print("no other")
        return 1
```

```
    else:
        total = 1
        for bag in bag_dict[bag]:
            total += all_bags_unpacked(bag, bag_dict, show_me)
        return total
```

[22]:
```
# subtract the first bag, just the ones it contains
all_bags_unpacked("shiny gold", bags_repeated, show_me=False) - 1
```

[22]: 10219

1.4 Testing

[23]:
```
test = {"shiny gold": ["dark red", "dark red"],
 "dark red": ["dark orange", "dark orange"],
 "dark orange": ["dark yellow", "dark yellow"],
 "dark yellow": ["dark green", "dark green"],
 "dark green": ["dark blue", "dark blue"],
 "dark blue": ["dark violet", "dark violet"],
 "dark violet": ["no other"],
}

test2 = {
    "faded blue": ["no other"],
    "dotted black": ["no other"],
    "vibrant plum": ["faded blue", "faded blue", "faded blue", "faded blue",␣
 ↪"faded blue",
                     "dotted black", "dotted black", "dotted black", "dotted␣
 ↪black" ,"dotted black", "dotted black"],
    "dark olive": ["dotted black", "dotted black", "dotted black", "dotted␣
 ↪black",
                   "faded blue", "faded blue", "faded blue"],
    "shiny gold": ["dark olive", "vibrant plum", "vibrant plum"]
}
```

[24]:
```
all_bags_unpacked("shiny gold", test, show_me=False)
```

[24]: 127

[25]:
```
all_bags_unpacked("shiny gold", test2, show_me=False)
```

[25]: 33