Day 19: Linen Layout

Teil 1 lässt sich wunderbar (und relativ einfach) mit **Rekursion** lösen. Für Teil 2 benötigt man zwingend eine kleine Erweiterung um die **dynamische Programmierung** zu nutzen - es müssen nämlich etwa einige Billionen (!) Kombinationen gefunden bzw. berechnet werden. Bei der dynamischen Programmierung werden Zwischenergebnisse gespeichert und später mehrfach wiederverwendet, um nicht wieder und wieder dasselbe zu berechnen.

Teil 1

Vorgehensweise:

- Lies zunächst alle towels der ersten Zeile in eine `ArrayList<String>` ein.
- Gehe anschließend Zeile für Zeile über die nachfolgenden Designs. Rufe pro Zeile eine rekursive Methode auf, die sich Buchstaben für Buchstaben durch das "Design" arbeiten soll, um ein jeweils passendes Towel zu finden. Die Rekursion benötigt die folgenden Parameter:
  - Sie muss das Design/pattern kennen (Also den String der jeweiligen Input-Zeile).
  - Sie muss wissen, für den wievielten Buchstaben nun das passende towel gefunden werden muss (eine Positionsangabe)
  - Sie muss die ArrayList aller towels kennen.
- Mit diesen Parametern gewappnet muss die Rekursion nun Folgendes tun:
  - Der Basisfall: Die vorherigen Rekursionen waren alle so "erfolgreich", dass die Positionsangabe bereits über die Länge des Strings hinausgewachsen ist (Position = String-Länge (Indizes beginnen bei 0!)). Wenn das der Fall ist, dann wurden erfolgreich Towel-Kombinationen für das aktuelle Design gefunden → Es kann "1" zurückgegeben werden, um zu signalisieren, dass ein Treffer gefunden wurde.
  - Ansonsten sind wir noch nicht am Ende angelangt, müssen also für den Buchstaben an der gegebenen Position ein passendes Towel finden! Daher müssen wir in einer Schleife über alle Towels iterieren.
    - Wenn die Länge eines Towels länger ist als die verbleibende Länge von der Positionsangabe bis zum Ende des Strings, dann kann man direkt mit dem nächsten Towel weitermachen (`continue;` → Der Rest der Towel-Schleife wird übersprungen und der nächste Schleifendurchlauf wird begonnen).
    - Ansonsten muss man das aktuelle Towel Buchstaben für Buchstaben mit dem Design ab der übergebenen Positionsangabe vergleichen. Passt es nicht, so muss mit dem nächsten Towel weitergemacht werden. Wenn es gepasst hat, dann kann die Positionsangabe um die Länge des Towels verschoben werden und es beginnt ein neuer Rekursionaufruf.
    - Je nachdem, was dieser innere Rekursionsaufruf schließlich für ein Ergebnis zurückliefert (1 = es wurde ein Ergebnis gefunden / 0 = kein Ergebnis), so wird entweder direkt die 1 als Rückgabewert der Rekursion zurückgeliefert, oder man muss das nächste Towel ausprobieren.
  - Sind schließlich alle Towels ausprobiert worden, und die Schleife wird verlassen, so bedeutet das, dass der `return`-Aufruf nie stattgefunden hat, dass also keine Lösung gefunden wurde. So muss nun 0 zurückgegeben werden.

Lösungsvorschlag

```
public void partOne() {
    // Towels
    String[] input = inputLines.get(0).split(",");
    ArrayList<String> towels = new ArrayList();
    for (String s: input) {
        towels.add(s.trim());
    }

    // Iteriere über die Designs/patterns
    int possible = 0;
    for (int i = 2; i < inputLines.size(); i++) {
        String pattern = inputLines.get(i);

        possible += matchPattern(pattern, 0, towels);
    }
    System.out.println(possible);
}

private int matchPattern(String pattern, int pos, ArrayList<String> towels)
{
    if (pos == pattern.length()) {
        return 1;
    }

    //probiere jedes towel aus
    for (String towel: towels) {
        // wenn towel zu lang -> nächstes
        if (towel.length() > pattern.length()-pos) {
            continue;
        }

        // vergleiche Buchstaben für Buchstaben
        boolean passt = true;
        for (int i = 0; i < towel.length(); i++) {
            // passt nicht -> nächstes
            if (pattern.charAt(pos+i) != towel.charAt(i)) {
                passt = false;
                break;
            }
        }
        if (!passt) {
            continue;
        }

        // hat gepasst -> rekursion
        int res = matchPattern(pattern, pos + towel.length(), towels);
        if (res == 1) {
            return 1;
        }
```

```
    }

    // zuvor wurde 'return' nicht aufgerufen -> wurde KEIN Treffer gefunden.
    return 0;
}
```

Teil 2

Es müssen nur kleine Änderungen vorgenommen werden.

- Es muss eine **HashMap<Integer, Long>** erstellt werden, welche für eine Positionsangabe speichert, wie viele Lösungen dafür gefunden wurden.
- Wichtig: Verwende nun Long statt Integer an vielen Stellen, da die Zahlen sehr (!) groß werden können.
- In der Haupt-Methode muss die HashMap pro Design neu Initialisiert/"zurückgesetzt" werden, da die Längenangaben natürlich pro Design unterschiedlich sind.
- Prüfe pro Rekursionaufruf zu Beginn einen zweiten Basisfall: Falls für die aktuelle Positionsangabe bereits ein Ergebnis berechnet und gespeichert wurde, dann gib dieses direkt zurück.
- Innerhalb der Towel-Schleife in der Rekursionsmethode muss nun das Ergebnis der rekursiven Methode anders behandelt werden. Wenn die Rekursion erfolgreich war und einen oder mehrere Treffer hatte, dann wird dieses Ergebnis **nicht** direkt zurückgegeben. Vielmehr müssen alle Ergebnisse der Schleife aufsummiert werden und erst dieses Ergebnis wird nach der Schleife in der HashMap gespeichert und dann zurückgegeben. Hintergrund: Pro Position innerhalb eines Designs können ja verschiedene Towels angehängt werden und erfolgreich zu mehreren Abschlüssen führen.

Lösungsvorschlag

```
public void partTwo() {
    String[] input = inputLines.get(0).split(",");
    ArrayList<String> towels = new ArrayList();
    for (String s: input) {
        towels.add(s.trim());
    }

    long possible = 0;
    for (int i = 2; i < inputLines.size(); i++) {
        String pattern = inputLines.get(i);

        amountOfResults = new HashMap();
        possible += findAllMatches(pattern, 0, towels);
    }
    System.out.println(possible);
}

private long findAllMatches(String pattern, int pos, ArrayList<String>
towels) {
    if (pos == pattern.length()) {
```

```
            return 1;
        }

        Long amount = amountOfResults.get(pos);
        if (amount != null) {
            return amount;
        }

        long sum = 0;

        //probiere jedes towel aus
        for (String towel: towels) {
            // wenn towel zu lang -> nächstes
            if (towel.length() > pattern.length()-pos) {
                continue;
            }

            // vergleiche Buchstaben für Buchstaben
            boolean passt = true;
            for (int i = 0; i < towel.length(); i++) {
                // passt nicht -> nächstes
                if (pattern.charAt(pos+i) != towel.charAt(i)) {
                    passt = false;
                    break;
                }
            }
            if (!passt) {
                continue;
            }

            // hat gepasst -> rekursion
            long res = findAllMatches(pattern, pos + towel.length(), towels);

            sum += res;
        }

        amountOfResults.put(pos, sum);
        return sum;
    }
```