Day 4: Ceres Search

Der heutige Tag ist gar nicht mal so schwer, sobald man 1-2 mögliche Kniffe erkannt hat, wie man die Lösung angehen kann.

Klar ist, dass man den Input in einem **zweidimensionalen Array** speichern muss. Als Datentyp ist `char` allerdings nicht wirklich geeignet, da wir ja bestimmte **Abfolgen** in dem Text erkennen müssen. `int` bietet sich daher viel besser an!

Übertrage also als Erstes den Input in ein zweidimensionales int-Array, wobei X=1, M=2, A=3, S=4 ist.

Anschließend müssen wir nun alle Vorkommnisse der Zahlenfolge 1234 in dem Array finden (auch rückwärts, diagonal, etc). Es wäre absolut ungünstig, für jede der **8 Richtungen**, in die man suchen muss, eine eigene Methode/Strategie zu entwickeln. Klar ist also, wir brauchen eine Lösung, die alle 8 Richtungen universell abdecken kann.

Folgende Vorgehensweise bietet sich an: Suche den Startpunkt jeder Folge, also eine 1. Starte von dort ausgehend dann die Suche in alle 8 Richtungen. Diese 8 Richtungen weisen sich dadurch aus, dass man sich pro Schritt sowohl in x- als auch in y-Richtung jeweils -1, 0 oder +1 bewegt. Man kann also vom Startpunkt aus eine verschachtelte Schleife starten, die alle Kombinationen der delta-x und delta-y Schrittweiten abdeckt. Diese verschachtelte Schleife soll uns also z. B. die Schrittkombination `(dx=-1, dx=-1)` → nach links oben, `(dx=0, dy=-1)` → nach mittig oben, etc geben.

Innerhalb dieser verschachtelten Schleife bietet sich nun eine rekursive Methode an, die sich jeweils um die dx- und dy-Werte weiterbewegt und dabei jeweils prüft, ob an der nächsten Position der nächsthöhere int-Werte gefunden wird. Schafft diese Methode es, alle Zahlen bis inkl. der 4 zu finden, so kann 1 zurückgegeben werden (ein weiteres XMAS wurde gefunden). Andernfalls wird zurückgegeben. Mit einer 0 abbrechen muss die Methode sowohl, wenn die Arraygrenzen überschritten werden, als auch, wenn die nächstes gefundene Zahl falsch ist.

Lösungsvorschlag

```
public void partOne() {

    // Instanz-Variablen, damit die rekursive Methode darauf zugreifen kann!
    width = inputLines.get(0).length();
    height = inputLines.size();
    puzzle = new int[width][height];

    // übertrage den Input in ein int-Array
    for (int y = 0; y < height; y++) {
        String line = inputLines.get(y);
        for (int x = 0; x < width; x++) {
            char c = line.charAt(x);
            if (c == 'X') {
                puzzle[x][y] = 1;
            } else if (c == 'M') {
                puzzle[x][y] = 2;
```

```
            } else if (c == 'A') {
                puzzle[x][y] = 3;
            } else if (c == 'S') {
                puzzle[x][y] = 4;
            } else {
                // dürfte niemals eintreten...
                puzzle[x][y] = 0;
            }
        }
    }

    // zählt die Vorkommnisse des gesuchten Strings
    int xmas = 0;

    // Gehe über jede array-Koordinate
    for (int y = 0; y < height; y++) {
        for (int x = 0; x < width; x++) {
            // Wenn die Startposition ('X') gefunden wird...
            if (puzzle[x][y] == 1) {
                // Iteriere in einer doppelten Schleife über alle 8
Bewegungen.
                // Auch die "0-0 Bewegung" (Verbleib an derselben Position)
wird gefunden,
                // das wird aber korrekt in der rekursiven Methode
abgefangen
                for (int dx = -1; dx <= 1; dx++) {
                    for (int dy = -1; dy <= 1; dy++) {
                        xmas += searchXmas(x+dx, y+dy, dx, dy, 2);
                    }
                }
            }
        }
    }

    System.out.println(xmas);
}

/**
 * @param x aktuelle x-pos
 * @param y aktuelle y-pos
 * @param dx nächste x-richtung
 * @param dy nächste y-richtung
 * @param c aktuell zu überprüfende nummer/char
 */
private int searchXmas(int x, int y, int dx, int dy, int c) {
    // falls die Array-Grenzen überschritten werden...
    if (x < 0 || x >= width || y < 0 || y >= height) {
        return 0;
    }
    // falls der nächste Buchstabe die XMAS-Folge nicht korrekt fortsetzt...
```

```
    if (puzzle[x][y] != c) {
        return 0;
    }
    // Falls der letzte Buchstabe ('S') gefunden wurde, dann können wir die
rekursion erfolgreich beenden.
    if (c == 4) {
        return 1;
    }

    /* Rekursiver Aufruf:
     * x und y werden um eine Schrittweite erhöht
     * dx und dy bleiben unverändert
     * c muss um eins erhöht werden, damit der nächste Buchstabe gefunden
werden kann.
     * Wir benötigen für c an dieser Stelle den pre-increment Aufruf (++ VOR
dem Variablennamen).
     * Dies stellt sicher, dass die Zahl wirklich SOFORT erhöht wird und
erst danach die Rekursion aufgerufen wird.
     */
    return searchXmas(x+dx, y+dy, dx, dy, ++c);
}
```

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:java:aoc:aoc2024:day04:start?rev=1733299044

Last update: **04.12.2024 07:57**