J1 Landnahme

J1.1 Festlegungen zur Eingabe und deren Verarbeitung

Für die Lösung dieser Aufgabe gibt es verschiedene Strategien. Welche man verwenden kann, hängt auch davon ab, mit welchen Eingaben man zurecht kommen muss, und zu welcher Zeit man auf einzelne Eingaben reagiert. Zunächst zur Eingabe: Die besteht laut Aufgabenstellung aus einer Liste von Rechtecken. Ein Rechteck ist durch vier Zahlen gegeben: die Koordinaten zweier sich diagonal gegenüberliegender Ecken. Bezüglich dieser Zahlen sind zwei Punkte nicht genau festgelegt, nämlich das Zahlenformat und der Zahlenbereich, so dass man hier mehrere Möglichkeiten hat:

Zahlenformat Auch wenn die Beispieleingabe nur natürliche Zahlen enthält, schließt die Aufgabenstellung nicht ausdrücklich aus, dass die Koordinaten der Punkte auch negative (ganze) Zahlen oder Fließkommazahlen sein können. Man muss also selbst einen sinnvollen Datentyp festlegen. Je nachdem verwendet man entweder einen Datentyp für ganze Zahlen, wie *int* oder *long*, oder einen für Fließkommazahlen, wie *float* oder *double*.[1]

Zahlenbereich Wichtig ist auch, dass man eine Obergrenze für die Größe der Zahlen festlegt. Verwendet man z.B. eine 32-Bit-Ganzzahl (*int*), so dürfen die Koordinaten nicht größer sein als $2^{31} - 1 = 2147483647$. Erwartet man auch größere Koordinaten, so sollte man auf 64-Bit-Ganzzahlen (*long*) zurückgreifen. Man kann sich aber überlegen, dass auch im wilden Westen Entfernungen nicht in Millimeter angegeben wurden, oder dass niemand mehrere 1000km in 24 Stunden zu Fuß schaffen kann. Einem alten Commodore reicht also auch ein 16-Bit-Ganzzahl, die immerhin Koordinaten bis $2^{15} - 1 = 32768$ speichern kann.

Online oder nicht Ein dritter offener Punkt ist, wie der Commodore auf die Eingabe reagieren muss. Es ist nicht angegeben, ob er sofort nach der Umrundung des Rechteckes entscheiden muss, ob es genehmigt ist, oder ob er erst entscheidet, wenn er alle Rechtecke kennt. Wählt man eine Strategie, mit welcher der Commodore sofort nach jedem Rechteck entscheidet, so spricht man von einem Online-Algorithmus. Für andere Lösungsstrategien wiederum ist es wichtig, dass man alle Rechtecke schon zu Beginn kennt, da das Vorberechnungen ermöglicht. Online oder nicht – beides ist vertretbar.

J1.2 Lösungsstrategien

Prinzipiell muss man die Rechtecke in der Reihenfolge abarbeiten, in der sie in der Eingabe stehen. Man kann nur entscheiden, ob ein Rechteck genehmigt wird oder nicht, wenn diese Frage schon für alle Rechtecke davor entschieden wurde. Daher ist die grundlegende Idee, sich die Menge der schon genehmigten Rechtecke zu speichern und dann immer das nächste Rechteck zu überprüfen. Algorithmus 1 drückt diese grundlegende Verfahrensweise in Pseudocode aus.

[1] Die Namen von Datentypen variieren je nach Programmiersprache. In einigen Programmiersprachen, wie z.B. Python, muss kein Datentyp angegeben werden.

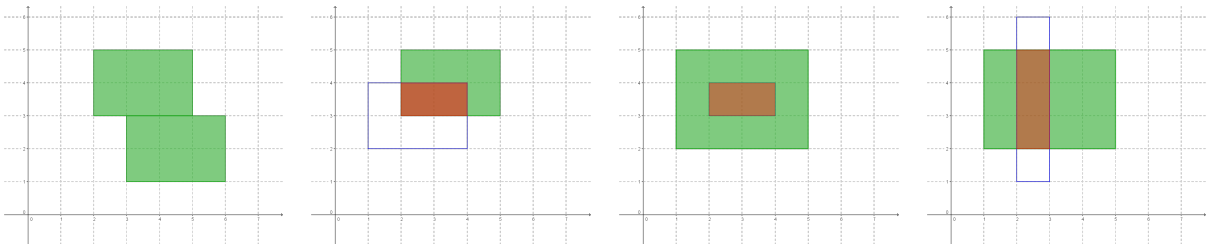
Algorithmus 1 Arbeitsanweisung für den alten Commodore

```
function LANDNAHME
    M ← ∅                          ▷ Menge der genehmigten Rechtecke; zu Beginn leer
    for Rechteck r do
        if r überschneidet sich mit einem Rechteck aus M then
            PRINT('abgelehnt')
        else
            PRINT('genehmigt')
            füge r zu M hinzu
        end if
    end for
end function
```

Interessant ist nun, wie man die Menge M speichert und wie man überprüft, ob sich ein Rechteck mit einem anderen überschneidet. Dazu werden im Folgenden zwei Varianten vorgestellt.

Variante 1: Online-Algorithmus für beliebige Zahlen

Man speichert die Rechtecke mit ihren Koordinaten in einem Array, in einer verketteten Liste oder einer ähnlichen Datenstruktur. Aus den gegebenen Koordinaten eines Rechtecks gehen die minimale und maximale x-Koordinate x_{min} und x_{max} bzw. die minimale und die maximale y-Koordinate y_{min} und y_{max} des Rechtecks direkt hervor. Damit lassen sich die Koordinaten aller Eckpunkte eines Rechtecks problemlos konstruieren.



(a) angrenzend (genehmigt) (b) überlappend (abgelehnt) (c) enthalten (abgelehnt) (d) überkreuz (abgelehnt)

Abbildung 1: Wichtige Fälle bei der Überschneidungsprüfung

Um zu ermitteln, ob ein neu beantragtes Rechteck sich mit einem schon genehmigten überschneidet, überprüft man für jedes bereits genehmigte Rechteck einzeln, ob das beantragte sich mit ihm überschneidet. Eine solche Prüfung muss alle Fälle korrekt behandeln, insbesondere auch die in Abb. 1 gezeigten. Eine naheliegende Idee, nämlich die Überprüfung, ob mindestens ein Eckpunkt des beantragten Rechtecks im bereits genehmigten Rechteck enthalten ist, scheitert am Fall „überkreuz“ und am Umkehrfall von „enthalten“ (wenn das genehmigte Rechteck im beantragten enthalten ist).

Ein sicheres Kriterium ist hingegen das folgende: Ein Rechteck r_1 überschneidet sich dann nicht mit einem Rechteck r_2, wenn es links, rechts, oben oder unten von r_2 liegt. Das bedeutet z.B., dass die rechte Seite von r_1 links von der linken Seite von r_2 liegen könnte, also: $r_1.x_{max} <$

$r_2.x_{min}$. Betrachtet man alle Seiten, dann kommt man darauf, dass sich zwei Rechtecke genau dann nicht überschneiden, wenn gilt ($\vee$ ist das logische „oder“):

$$(r_1.x_{max} \leq r_2.x_{min}) \vee (r_1.x_{min} \geq r_2.x_{max}) \vee (r_1.y_{max} \leq r_2.y_{min}) \vee (r_1.y_{min} \geq r_2.y_{max}) \quad (1)$$

Indem man das Ergebnis dann noch logisch negiert, kann man ermitteln, ob sich die beiden Rechtecke schneiden. Statt mit den Koordinaten der Eckpunkte kann man auch mit Mittelpunkten und Seitenlängen rechnen, um zu klären, ob zwei Rechtecke ohne Überlappung neben- bzw. übereinander liegen.

Der Algorithmus ist online, da man keine Informationen über spätere Rechtecke benötigt. Weiterhin ist es egal, welches Zahlenformat man verwendet, da man nur Zahlen vergleichen muss. Zur Laufzeit kann man sich Folgendes überlegen: Im schlimmstem Fall muss man jedes Rechteck mit jedem anderen vergleichen, also bei n Rechtecken $0+1+2+...+(n-1) = \frac{n(n-1)}{2} \in O(n^2)$ Vergleiche ausführen[2]. Bei einer Siedlerschwemme könnte der alte Commodore mit diesem Verfahren also durchaus ins Schwitzen kommen: Bei 1000 Rechtecken stehen etwa 1 Million Vergleiche, bei 1 Million Rechtecke etwa 1 Billion Vergleiche an.

Da man die Koordinaten nur miteinander vergleicht, ist es für die Betrachtung der Laufzeit egal, welche Werte die Koordinaten haben. Diese Lösung eignet sich also für eine nicht allzu große Zahl von Rechtecken, deren Koordinaten dafür beliebig geartet und beliebig groß sein dürfen.

Variante 2: Für kleinere, ganzzahlige Koordinaten

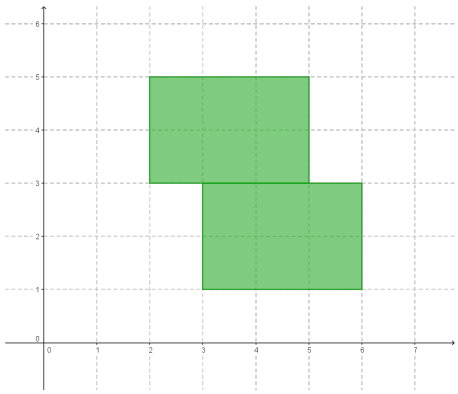
Wenn man davon ausgeht, dass die Koordinaten nur natürliche Zahlen sind, so kann auch folgende Variante sehr effektiv sein. An Stelle einer Liste von Rechtecken speichert man eine Landkarte. Diese Landkarte unterteilt man in Quadrate der Größe 1×1 und speichert für jedes Quadrat, ob es Teil eines Rechteckes ist. Das geht nur, wenn die Eingabe nur aus natürlichen Zahlen besteht[3]. Die Landkarte kann man dann zum Beispiel als zweidimensionales Array A speichern. Dabei ist der Eintrag $A[x][y]$ gleich 1, wenn das Einheitsquadrat an der Position (x,y) schon in einem Rechteck enthalten ist, und 0 sonst. Abb. 2 zeigt ein Beispiel für diese Umsetzung.

Da man vor dem Anlegen eines Arrays schon die benötigte Größe wissen muss, kann man diesen Algorithmus nur verwenden, wenn man weiß, welche Koordinaten in der Eingabe auftreten werden. Dazu müsste man theoretisch alle Rechtecke von Beginn an kennen, so dass der Algorithmus nicht online ist. Man kann aber spezielle Datenstrukturen wie dynamische Arrays verwenden, die man bei Bedarf in ihrer Größe verändern kann, um aus dem Algorithmus doch wieder einen Online-Algorithmus zu machen. Es ist aber – gerade für eine Junioraufgabe – auch in Ordnung, selbst eine Obergrenze für die Koordinaten festzulegen; dann funktioniert der Algorithmus auch bei einem Array fester Größe online.

Will man ermitteln, ob ein neues Rechteck genehmigt wird oder nicht, geht man alle Einheitsquadrate durch, die in dem neuen Rechteck liegen. Ist der Eintrag eines dieser Einheitsquadrate gleich 1, also das Quadrat schon belegt, so wird das Rechteck nicht genehmigt. Ansonsten wird

[2] Die sogenannte O-Notation $O(f(n))$ gibt das charakteristische Verhalten der Funktion $f(n)$ an.

[3] Durch eine sogenannte Koordinatenkompression kann man auch Fließkommazahlen verwenden. Dabei verwendet man anstatt von Einheitsquadraten Rechtecke unterschiedlicher Größe. Allerdings ist dies deutlich komplizierter zu implementieren.



Rechtecke des Commodore

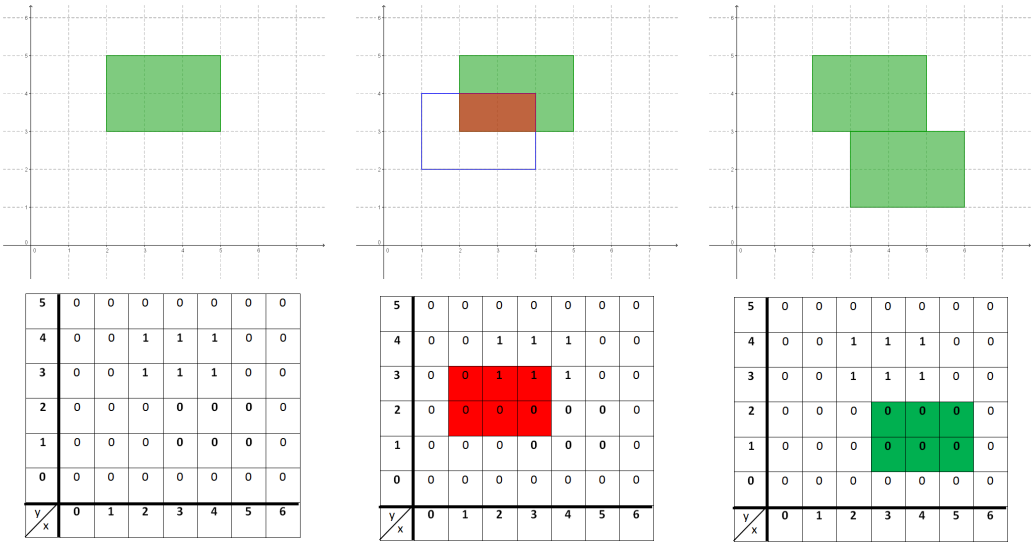
y / x	0	1	2	3	4	5	6
5	0	0	0	0	0	0	0
4	0	0	1	1	1	0	0
3	0	0	1	1	1	0	0
2	0	0	0	1	1	1	0
1	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

Zweidimensionales Array A

Abbildung 2: Repräsentation von Rechtecken in einem Array

das Rechteck genehmigt, und die Einträge der dadurch (neu) belegten Einheitsquadrate werden auf 1 gesetzt.

Das Problem bei dieser Variante ist, dass die Laufzeit abhängig ist von der Größe der Rechtecke. Um zu ermitteln, ob ein Rechteck mit Breite w und Höhe h genehmigt wird oder nicht, muss man $w \cdot h$ Einheitsquadrate überprüfen. Die Laufzeit ist also nur schwer abschätzbar und kann bei sehr großen Rechtecken sehr hoch sein.[4]



1. Rechteck (genehmigt) 2. Rechteck (abgelehnt) 3. Rechteck (genehmigt)

Abbildung 3: Abarbeitung des Beispiels aus der Aufgabenstellung

[4]Es gibt (äußerst komplizierte) Datenstrukturen, mit denen das Überprüfen und Hinzufügen eines Rechteckes in logarithmischer Laufzeit $O(\log n)$ möglich ist, unabhängig von der Größe der Rechtecke. Eine solche Datenstruktur ist z.B. ein Segmentbaum. Damit kann man insgesamt eine Laufzeit von $O(n \log n)$ erreichen, die sogar besser ist als Variante 1 mit $O(n^2)$.

J1.3 Beispiel

Im Folgenden wird das Vorgehen der beiden vorgestellten Varianten anhand der in der Aufgabenstellung angegebenen Beispieleingabe erläutert (vgl. Abb. 3). Zu Beginn gibt es kein auf Überschneidung zu prüfendes Rechteck, bzw. keine 1-er-Einträge für Einheitsquadrate auf der Landkarte. Daher wird in beiden Varianten das erste Rechteck genehmigt.

Sei r_1 das erste und r_2 das zweite Rechteck, mit $r_1.x_{min} = 2, r_1.x_{max} = 5, r_1.y_{min} = 3, r_1.y_{max} = 5$ und $r_2.x_{min} = 1, r_2.x_{max} = 4, r_2.y_{min} = 2, r_2.y_{max} = 4$. Wenn man in Variante 1 nach Formel 1 überprüft, ob sich beide Rechtecke nicht schneiden, dann erhält man:

$$(5 \leq 1) \vee (2 \geq 4) \vee (5 \leq 2) \vee (3 \geq 4) = \text{false}$$

Demzufolge schneiden sich beide Rechtecke. Analog sind bei Variante 2 die beiden Einträge A[2][3] und A[3][3] schon 1. Daher wird in beiden Varianten das zweite Rechteck abgelehnt.

Das dritte beantragte Rechteck muss in Variante 1 mit dem ersten Rechteck abgeglichen werden. Formel 1 ergibt für die Koordinaten dieser beiden Rechtecke

$$(6 \leq 2) \vee (3 \geq 5) \vee (3 \leq 3) \vee (1 \geq 5) = \text{true}$$

Die Rechtecke überschneiden sich also nicht. In Variante 2 wiederum sind alle Einheitsquadrate des dritten Rechtecks noch frei. Das dritte Rechteck wird also in beiden Varianten genehmigt.

Die Ausgabe zu diesem Beispiel sollte dann, wie in der Aufgabenstellung angegeben, folgendermaßen aussehen:

```
2 3 5 5 genehmigt
1 2 4 4 abgelehnt
3 1 6 3 genehmigt
```

J1.4 Bewertungskriterien

- Ausdrückliche Überlegungen zur Eingabe und zur Frage nach der Online-Abarbeitung der Eingabe, wie sie in diesen Lösungshinweisen angestellt werden, werden in der Bearbeitung einer Junioraufgabe nicht erwartet.
- Die Entscheidung über Genehmigung oder Ablehnung eines Grundstücks darf nur auf der Grundlage vorher genehmigter Grundstücke fallen; bereits abgelehnte Grundstücke oder in der Eingabe erst folgende Grundstücke dürfen keine Rolle spielen.
- Der Kern der Lösung ist die Prüfung zweier Rechtecke auf Überschneidung bzw. Nicht-Überschneidung. Sie soll grundsätzlich korrekt funktionieren, auch für speziellere Fälle. Falsche Ergebnisse in Einzelfällen, z.B. auf Grund von Fehlern in der Implementierung, sollen ebenfalls nicht vorkommen.
- Bewusst wurden bei dieser Aufgabe keine Pflichtbeispiele vorgegeben, um bei der Betrachtung der möglichen Fälle nicht zu viele Hinweise zu geben. Es ist aber eine grundsätzliche Anforderung an jede Dokumentation, dass das Funktionieren des Verfahrens anhand einiger geeigneter Beispiele dokumentiert wird.
- Falls die freiwillige Zusatzaufgabe (Visualisierung der Landnahme) einigermaßen brauchbar bearbeitet wurde, soll das in der Bewertung vermerkt sein, gibt aber keine Pluspunkte.