

## Junioraufgabe 2: Kacheln

### J2.1 Lösungsidee

Man stellt fest, dass immer vier Teilquadrate von vier im Quadrat zusammenliegenden Kacheln entweder alle aus Land oder alle aus Wasser bestehen müssen. Ansonsten würden zwei Kacheln in einer nicht erlaubten Weise aneinander liegen.

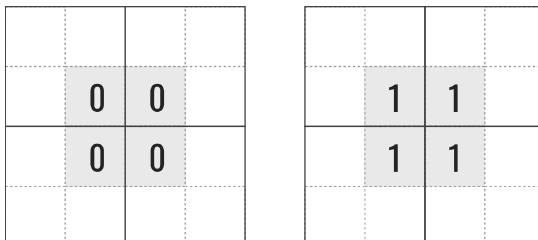


Abbildung J2.1: Mögliche Belegungen für die „inneren“ Teilquadrate von vier aneinander liegenden Kacheln

Wie oft, hilft auch hier zunächst ein Blick in die Beispieldaten. Da in den Eingabedaten für jedes Teilquadrat der Kacheln angegeben ist, ob es Land, Wasser oder noch unbestimmt ist, kann direkt für jede zusammengehörende Gruppe von Teilquadranten geprüft werden, ob sie

- a) nur Land (und ggf. unbestimmte Teilquadrate) oder nur Wasser (und ggf. unbestimmte Teilquadrate) enthalten,
- b) nur unbestimmte Teilquadrate enthalten, oder
- c) mindestens einmal Land und einmal Wasser enthalten.

Im letzten Fall lässt sich die Landschaft nicht vervollständigen, ohne am Ende eine nicht-erlaubte Anordnung der Kacheln zu erhalten. Selbst wenn das Land und das Wasser nicht direkt nebeneinander liegen, lassen sich die anderen Felder nicht mehr belegen, ohne eine nicht-erlaubte Anordnung zur erzeugen. In diesem Fall kann das Programm also sofort abgebrochen werden.

Im ersten Fall kann die Belegung der ggf. vorhandenen noch unbestimmten Teilquadrate einfach entsprechend der schon belegten Teilquadrate festgelegt werden. Da alle Möglichkeiten, Kacheln aus Teilquadranten zu erzeugen, als vorgegebene Kacheln vorhanden sind, gibt es keine Einschränkung bei der Erstellung der Kacheln aus den Teilquadranten.

Nur im Fall b) muss man kurz überlegen, was man tun kann. Auch hier stellt man wieder fest: Solange alle vier Teilkacheln in der selben Art belegt werden, können keine Einschränkungen verletzt werden. Ob man alle vier Teilquadrate mit Wasser belegt oder alle mit Land, spielt also keine Rolle.

Am Ende muss noch berücksichtigt werden, dass es am Rand der Landschaft auch zu Fällen kommt, bei denen nur jeweils zwei Teilquadrate auf Gleichheit geprüft werden müssen. Bei den vier Teilquadranten in den Ecken der Landschaft ist jede beliebige Belegung möglich.

### J2.2 Lösung mit Graph

*Vorbemerkung: Diese Lösungsidee verwendet Methoden, die weit über die Anforderungen einer Junioraufgabe hinausgehen.*

Unsere Karten bestehen aus Kacheln, die wiederum aus  $2 \times 2$  kleineren Teilkacheln besetzen. Wir beschreiben jede Teilkachel durch ein Koordinatenpaar  $(x, y)$ . Ist die Karte  $w$  Kacheln breit und  $h$  Kacheln hoch, so bezeichnet  $(0, 0)$  die Teilkachel oben links und  $(2 \cdot w - 1, 2 \cdot h - 1)$  die Teilkachel unten rechts. Insgesamt gibt es somit  $4 \cdot w \cdot h$  Teilkacheln. Analog zu den Teilkacheln können wir den Kacheln Koordinaten von  $(0, 0)$  bis  $(w - 1, h - 1)$  zuordnen. Beschreibt  $(x, y)$  eine Teilkachel, so bezeichne  $\mu(x, y) := (\lfloor \frac{x}{2} \rfloor, \lfloor \frac{y}{2} \rfloor)$  die zugehörige Kachel.

Unser Ziel besteht darin, jeder Teilkachel ein Bit (0 = überwiegend Wasser, 1 = überwiegend Land) zuzuordnen, sodass zwei *benachbarte* Teilkacheln *verschiedener* Kacheln das gleiche Bit erhalten. Hierbei sind zwei Teilkacheln  $(x, y)$  und  $(x', y')$  benachbart, wenn sie eine gemeinsame Seite haben, d. h. wenn  $|x - x'| + |y - y'| = 1$  gilt. So stellen wir sicher, dass stets zwei Kacheln in ihrer angrenzenden Seite übereinstimmen.

Bevor wir jeder Teilkachel ein Bit zuordnen, konstruieren wir einen ungerichteten Graphen  $G = (V, E)$  mit  $V = \{0, 1, \dots, 2 \cdot w - 1\} \times \{0, 1, \dots, 2 \cdot h - 1\}$  als Knotenmenge. Es gibt also einen Knoten pro Teilkachel. Zwei Knoten  $(x, y), (x', y') \in V$  verbinden wir genau dann durch eine Kante, wenn  $(x, y)$  und  $(x', y')$  benachbart sind und zu verschiedenen Kacheln gehören. Formal:  $\{(x, y), (x', y')\} \in E \iff |x - x'| + |y - y'| = 1$  und  $\mu(x, y) \neq \mu(x', y')$ . Schließlich färben wir die Knoten wie folgt:

- Knoten mit vorgegebenem Bit 0 bzw. überwiegend Wasser werden **blau** gefärbt.
- Knoten mit vorgegebenem Bit 1 bzw. überwiegend Land werden **rot** gefärbt.
- Knoten, deren Bit nicht vorgegeben ist, werden gar nicht gefärbt und bleiben schwarz.

Abbildung J2.2 veranschaulicht unsere Konstruktion.

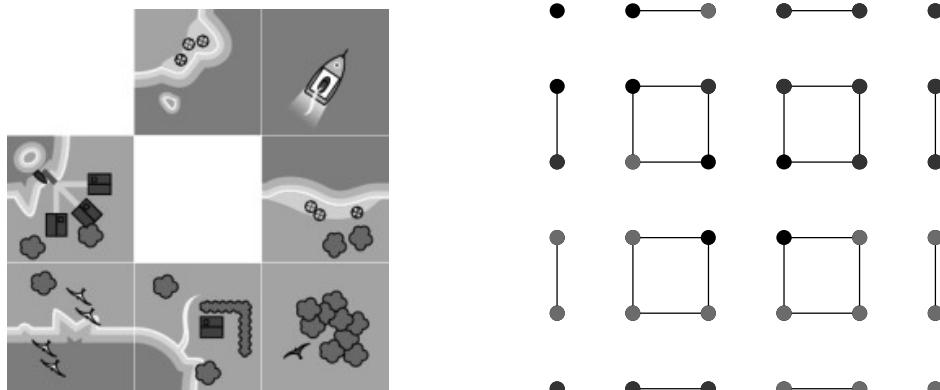


Abbildung J2.2: Konstruktion des Graphen  $G$  aus einer Karte der Größe  $w = h = 3$

Der Graph besteht aus drei Typen von Zusammenhangskomponenten: Isolierte Knoten, durch eine Kante verbundene Knotenpaare und Zyklen der Länge 4. Allgemein enthält er  $4 \cdot w \cdot h$  Knoten und  $4 \cdot w \cdot h - 2 \cdot w - 2 \cdot h$  Kanten.

**Beobachtung:** Allen Knoten einer Zusammenhangskomponente müssen wir die gleiche Farbe zuordnen. Ist wie in Abbildung J2.2 ein blauer und ein roter Knoten in derselben Zusammenhangskomponente, so existiert für das Beispiel keine Lösung.

In allen anderen Fällen gibt es eine Lösung, die sich folgendermaßen bestimmen lässt.

## Algorithmus

Es ist zwar möglich, aber keineswegs erforderlich,  $G$  explizit zu konstruieren und mit einem allgemeinen Algorithmus (z. B. Tiefensuche) die Zusammenhangskomponenten zu berechnen.

---

**Algorithmus 2** Lösung mit Graph

---

1. Konstruiere den Graphen  $G$  inklusive Färbung der Knoten.
  2. Für jede Zusammenhangskomponente  $Z$  von  $G$ ...
    - a) ... überprüfe, ob zwei Knoten in  $Z$  die gleiche Farbe haben und breche ggf. ab, denn es ist keine Lösung möglich.
    - b) ... überprüfe, ob  $Z$  nur schwarze Knoten enthält und färbe ggf. einen beliebigen Knoten blau (oder rot).
    - c) ... suche in  $Z$  einen bereits gefärbten Knoten und färbe alle Knoten in  $Z$  mit dieser Farbe.
  3. Übertrage die Färbung des Graphen  $G$  in eine Landkarte und gebe diese im vorgegebenen Ausgabeformat aus.
- 

Stattdessen können wir die in Abbildung J2.2 ersichtliche spezielle Struktur des Graphen ausnutzen und drei Fälle betrachten: Eckknoten, Randknoten (jene Knoten, die mit genau einem anderen Knoten verbunden sind) und innere Knoten, die in  $2 \times 2$ -Quadranten auftreten. Die vier Eckknoten können wir beliebig einfärben, sofern diese schwarz sind. Die Fälle der Randknoten und inneren Knoten lassen sich mit einigen Schleifen schnell erledigen.

**Laufzeit:** Die  $\leq 4$  Knoten einer Zusammenhangskomponente lassen sich in Zeit  $\mathcal{O}(1)$  färben. Da wir jede Zusammenhangskomponente genau einmal betrachten und es höchstens so viele Zusammenhangskomponenten wie Knoten gibt, liegt die Laufzeit insgesamt in  $\mathcal{O}(w \cdot h)$ , ist also linear in der Anzahl der Knoten.

### J2.3 Weitere Lösungswege

Es sei eine Karte der Breite  $w$  und Höhe  $h$  gegeben, wobei die Funktion

$$\tau : \{0, 1, \dots, w-1\} \times \{0, 1, \dots, h-1\} \rightarrow \{\perp, 0, 1, \dots, 15\} \quad (\text{J2.1})$$

den Kacheln  $(x, y)$  ihren Typ  $\tau(x, y)$  zuordnet. Hierbei markiert  $\perp$  nicht vorgegebene Kacheln und die Zahlen 0 bis 15 bezeichnen die Kacheltypen von 0 = nur Wasser bis 15 = nur Land. Die Gestalt der Teilkacheln ist der Binärdarstellung dieser Nummer wie folgt zu entnehmen: Ist  $\tau(x, y) = t_3 \cdot 2^3 + t_2 \cdot 2^2 + t_1 \cdot 2 + t_0$  mit  $t_0, t_1, t_2, t_3 \in \{0, 1\}$ , dann hat Teilkachel  $(2 \cdot x, 2 \cdot y)$  den Typ  $t_0$ , Teilkachel  $(2 \cdot x + 1, 2 \cdot y)$  den Typ  $t_1$ , Teilkachel  $(2 \cdot x, 2 \cdot y + 1)$  den Typ  $t_2$  und Teilkachel  $(2 \cdot x + 1, 2 \cdot y + 1)$  den Typ  $t_3$ . Das niedrigwertigste Bit entspricht also der Teilkachel oben links und das höchswertige Bit der Teilkachel unten rechts.

### Brute-Force-Lösung

Eine Brute-Force-Lösung wählt nach und nach für jede unbekannte Kachel einen passenden Typen aus und bricht ab, wenn alle Kacheln gesetzt sind oder es für eine Kachel keinen Typ mehr gibt. Im letzteren Fall wird der gesamte Algorithmus neu ausgeführt – in der Hoffnung, dass eine Lösung gefunden wird. Ist nach einer großen Zahl von  $C$  Iterationen (z. B.  $C = 20000$ ) keine Lösung gefunden, gibt der Algorithmus aus, dass keine Lösung existiert.

---

**Algorithmus 3** Brute-Force-Lösung

---

1. Initialisiere die Karte und setze  $k \leftarrow 0$ .
  2. Wenn  $k \geq C$ , brich ab und gebe „Nein“ aus.
  3. Wenn es keine Kachel  $(x, y)$  mit  $\tau(x, y) = \perp$  gibt, brich ab und gebe „Ja“ aus.
  4. Wähle eine beliebige Kachel  $(x, y)$  mit  $\tau(x, y) = \perp$  aus.
  5. Ermittle für alle  $i, j \in \{0, 1\}$  alle möglichen Typen der Teilkachel  $(2 \cdot x + i, 2 \cdot y + j)$  und wähle – sofern beides möglich ist – zufällig Land oder Wasser aus. Falls es für eine Teilkachel keinen Typen gibt, setze  $k \leftarrow k + 1$  und gehe zu Schritt 2.
  6. Weise der Kachel den ermittelten Typ zu.
  7. Gehe zu Schritt 3.
- 

Diese Lösung hat den großen Nachteil, nicht korrekt zu sein. Es gibt einen einseitigen Fehler: Wenn keine Lösung existiert, wird dies in jedem Fall richtig erkannt. Wenn es hingegen eine Lösung gibt, so besteht die Gefahr, dass der Algorithmus trotzdem „Nein“ ausgibt.

**Rekursives Backtracking**

Eine einfache und auf den BwInf-Beispielen gut funktionierende Lösung ist mit Backtracking möglich. Dazu iterieren wir über alle Kacheln und überspringen solche mit bereits vorgegebenem Typ. Hat eine Kachel den Typ  $\perp$ , so prüfen wir für jeden Typ, ob wir ihn der Kachel zuordnen können. Das ist genau dann der Fall, wenn die Kachel mit den benachbarten Kacheln in ihrer angrenzenden Seite übereinstimmt (d. h. die zwei Teilkacheln sind jeweils vom gleichen Typ). In diesem Fall ordnen wir der Kachel den Typ zu und fahren mit der nächsten Kachel fort. Sobald wir zum Ende kommen, ist eine Lösung gefunden.

Es kann sein, dass wir einer Kachel keinen der 16 Typen zuordnen können. Dann machen wir solange Schritte rückgängig, bis wir einer Kachel einen anderen Typen zuordnen können und setzen die Suche nach einer Lösung an dieser Stelle fort. Ist auch das nicht möglich, so haben wir alle Möglichkeiten ausprobiert und können die Existenz einer Lösung ausschließen.

Algorithmus 4 beschreibt das Vorgehen formal. Hierbei kann die mit  $(*)$  gekennzeichnete Zeile mittels einer Fallunterscheidung umgesetzt werden. Die Funktionen  $\alpha$  und  $\beta$  sind durch

$$\alpha(x, y) = \begin{cases} x + 1 & \text{falls } x < w - 1 \\ 0 & \text{falls } x = w - 1 \end{cases} \quad \beta(x, y) = \begin{cases} y & \text{falls } x < w - 1 \\ y + 1 & \text{falls } x = w - 1 \end{cases} \quad (\text{J2.2})$$

definiert, sodass  $(\alpha(x, y), \beta(x, y))$  nach  $(x, y)$  die nächste Kachel beschreibt.

Ein Nachteil dieser Lösung ist ihre im schlechtesten Fall exponentielle Laufzeit.

**J2.4 Beispiele**

Die folgenden Abbildungen zeigen jeweils zwei Karten. Die linke (unvollständige) Karte ist die Eingabe, die rechte Karte eine mögliche Lösung. Hervorzuheben ist das unlösbare Beispiel in

**Algorithmus 4** Lösung mit Backtracking

---

```

procedure BACKTRACKING( $x, y$ )
  if  $y = h$  then
    return 1
  end if
  if  $\tau(x, y) = \perp$  then
    for  $t = 0$  to 15 do
      if Kachel ( $x, y$ ) kann Typ  $t$  zugeordnet werden then ▷ (*)
         $\tau(x, y) \leftarrow t$ 
        if BACKTRACKING( $\alpha(x, y), \beta(x, y)$ ) = 1 then
          return 1
        end if
      end if
    end for
    return 0 ▷ keine Lösung
  else
    return BACKTRACKING( $\alpha(x, y), \beta(x, y)$ )
  end if
end procedure

```

---

Abbildung J2.8: Hier ist es z. B. nicht möglich, die Kachel (3, 1) festzulegen, denn ihre linke untere Teilkachel stimmt weder mit Wasser noch mit Land mit den angrenzenden Teilkacheln überein. Somit existiert keine Lösung.

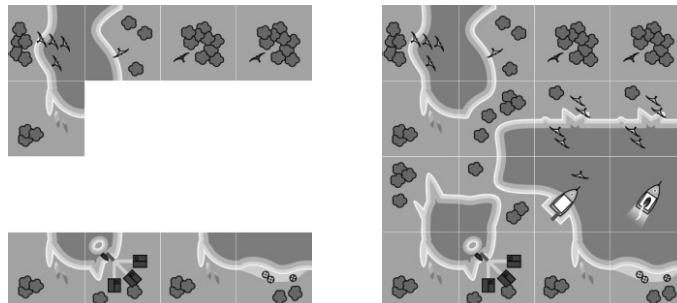


Abbildung J2.3: Eigenes Beispiel;  $4 \times 4$  Landschaft

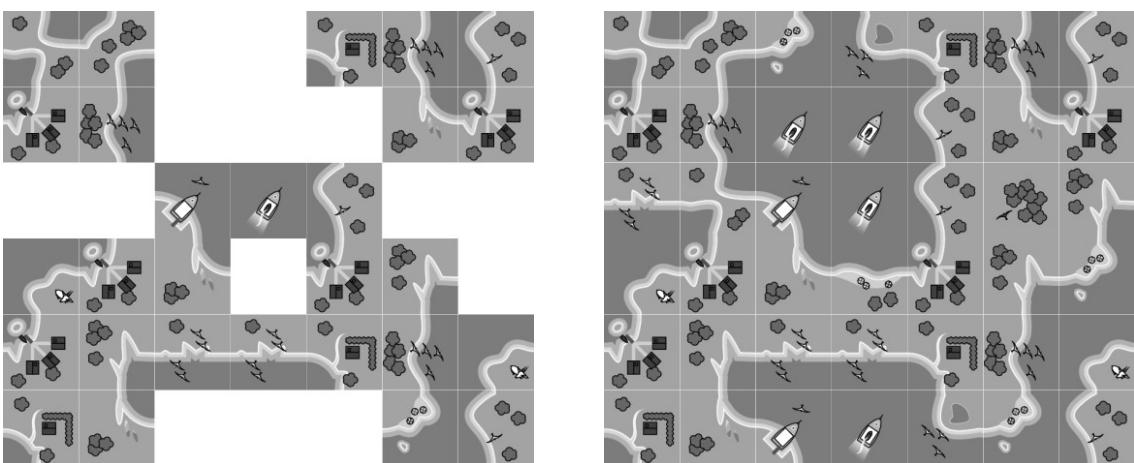
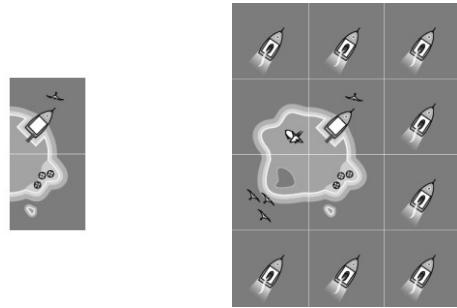
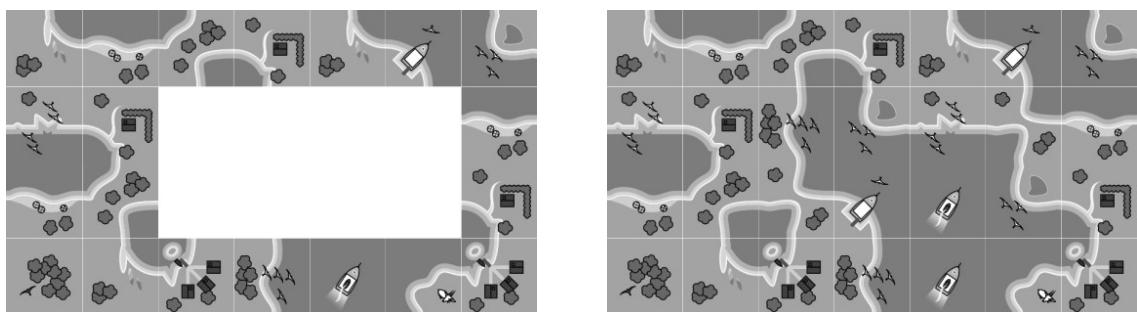
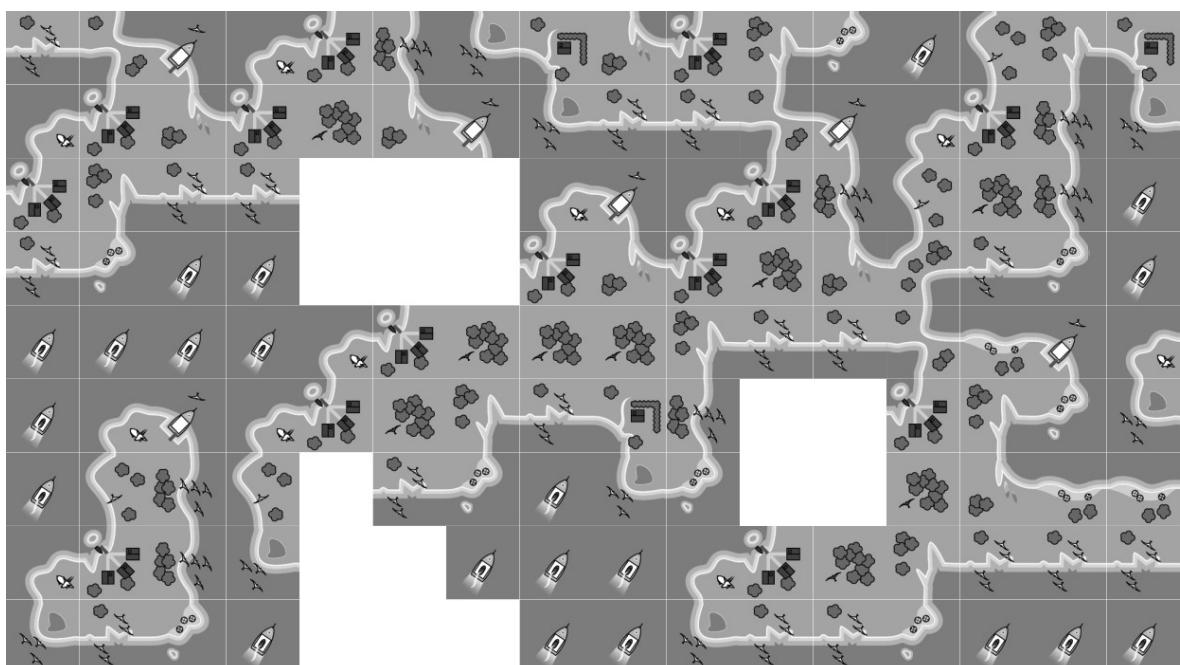
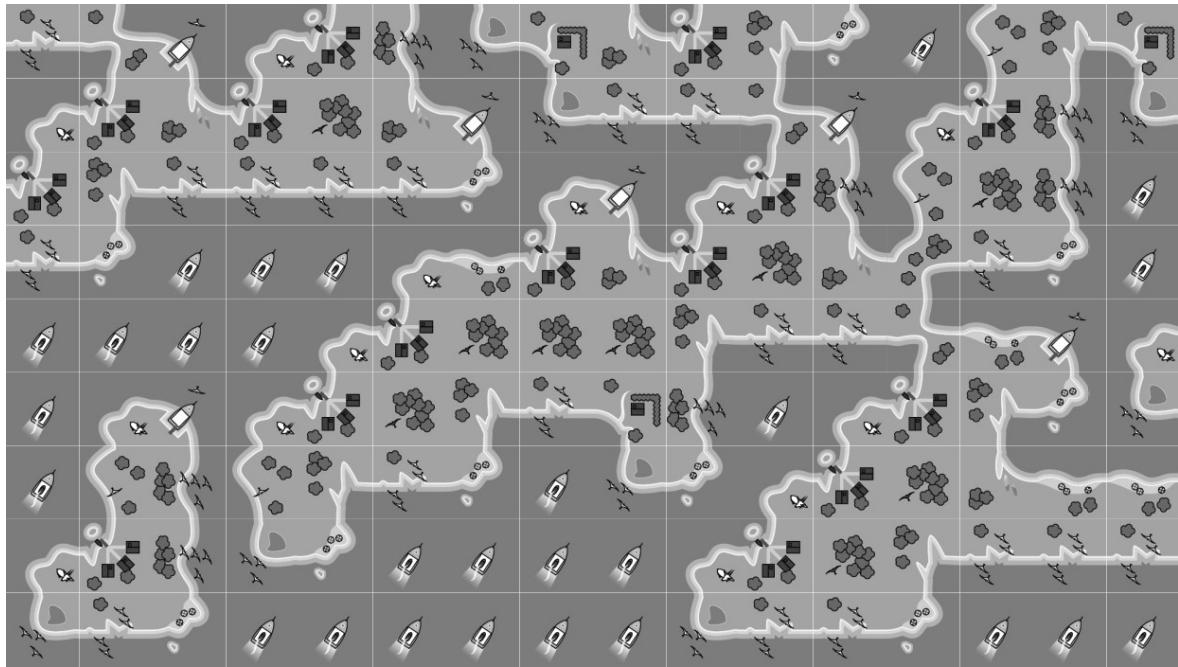
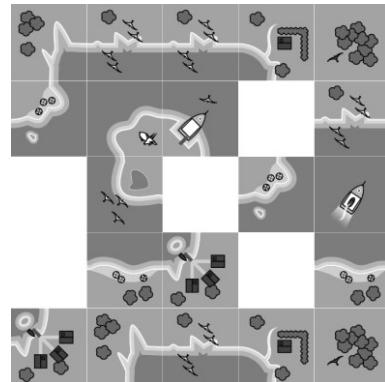


Abbildung J2.4: Beispiel 1;  $6 \times 7$  Landschaft

Abbildung J2.5: Beispiel 2;  $4 \times 3$  LandschaftAbbildung J2.6: Beispiel 3;  $4 \times 7$  Landschaft

Abbildung J2.7: Beispiel 4;  $9 \times 16$  LandschaftAbbildung J2.8: Beispiel 5 (unlösbar);  $5 \times 5$  Landschaft

## J2.5 Bewertungskriterien

Die Bewertungskriterien vom Bewertungsbogen werden hier erläutert (Punktabzug in []).

- [-1] **Lösungsverfahren fehlerhaft**

Ist keine Lösung möglich, so sollte dies erkannt werden. Ansonsten ist eine korrekte Lösung zu berechnen, d. h. benachbarte Kacheln stimmen in ihrer angrenzenden Seite überein und die bereits vorgegebenen Kacheln werden nicht verändert.

- [-1] **Verfahren bzw. Implementierung unnötig aufwendig**

Es sind fast alle (korrekten) Verfahren erlaubt, solange sie keine offensichtlich unnötigen Dinge tun. Insbesondere sollte das Programm abbrechen, wenn es eine Stelle findet, an der bereits die in der Eingabekarte vorgegebenen Kacheln eine Vervollständigung unmöglich machen. So ist etwa ein Backtrackingverfahren, das bereits in der Eingabekarte vorhandene Konflikte nicht erkennt, nicht ausreichend.

- [−1] **Ausgabe schlecht nachvollziehbar**

Die Programmausgabe muss die Korrektheit der Lösung leicht überprüfbar machen. Dazu reicht ein Ausgabeformat, das dem Eingabeformat entspricht und binäre Werte (0 und 1, W und L, ...) in einem passenden Raster ausgibt.

- [−1] **Beispiele fehlerhaft bzw. zu wenige oder ungeeignete Beispiele**

Die Ergebnisse zu den vorgegebenen Beispielen sollen mehrheitlich (mindestens drei von fünf, darunter das unlösbare Beispiel 5) dokumentiert und korrekt sein.