1. (4 Punkte) Ein Suchproblem hat die Zustände A bis G. Die Tabelle gibt an, welche Folgezustände `nextstates` für einen Zustand liefert, und zwar in dieser Reihenfolge. Startzustand ist A, Zielzustand ist G.

Zustand	Folgezustände
A	B, C
B	A, D, E
C	A, F
D	B, G
E	B, G
F	C, G
G	D, E, F

   (a) In welcher Reihenfolge werden die Zustände von der Breitensuche untersucht?

   (b) Gib die frontier und das Dictionary `prev` an, nachdem der Zustand C untersucht wurde.

   (c) Welchen Weg liefert `reconstructPath` am Ende?

2. (3 Punkte) (a) Warum findet die Breitensuche einen Weg mit möglichst wenigen Zügen?

   (b) In `bfs` wird `frontier.popleft()` durch `frontier.pop()` ersetzt. Was ändert sich?

   (c) Das Dictionary `prev` speichert die Vorgänger. Wozu wird es in `bfs` außerdem gebraucht?

3. (1 Punkt) Zeichne den Weg ein, den die Breitensuche von S nach E findet. (Markiere die entsprechenden Punkte oder verbinde sie mit einer Linie.) Die Breitensuche erzeugt die Folgezustände in der Reihenfolge oben, unten, links, rechts.

```
XXXXXXXXXXXXXXXXXX
X.............E...X
X..................X
X..XXXX..XXXXX...X
X..................X
X..S...............X
XXXXXXXXXXXXXXXXXX
```

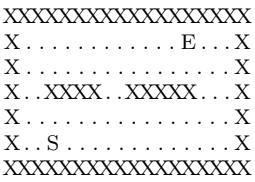
4. (2 Punkte) Ein Feld hat die Koordinaten (Zeile, Spalte), das X oben links hat die Koordinaten (0, 0). Die Breitensuche erzeugt die Folgezustände in der Reihenfolge oben, unten, links, rechts. (Die Punkte sollen das Abzählen erleichtern.)

   Die Breitensuche wird angehalten, nachdem der Zustand (4, 1) untersucht wurde. Wie viele Zustände wurden untersucht, wie viele befinden sich in der frontier? Zeichne in das große Labyrinth die Zustände in der frontier mit einer Tilde ein, die untersuchten Zustände mit einem Punkt.

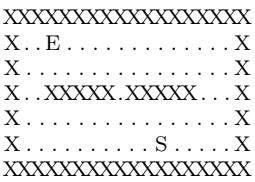
```
XXXXXXXXXX
X........X
X..E.....X
XXXXXXX..X
X....S...X
XXXXXXXXXX
```

```
XXXXXXXXXX
X........X
X........X
X........X
X........X
XXXXXXXXXX
```

5. (2 Punkte) Zeichne den Weg ein, den die Tiefensuche von S nach E findet, und gib seine Länge an. (Markiere die entsprechenden Punkte oder verbinde sie mit einer Linie.) Die Tiefensuche legt die Folgezustände so auf den Stack, dass sie in der Reihenfolge oben, unten, links, rechts wieder herausgenommen werden.



6. (2 Punkte) Zeichne den Weg ein, den die Tiefensuche von S nach E findet, und gib seine Länge an. (Markiere die entsprechenden Punkte oder verbinde sie mit einer Linie.) Die Tiefensuche legt die Folgezustände so auf den Stack, dass sie in der Reihenfolge oben, unten, links, rechts wieder herausgenommen werden.



7. (2 Punkte) Ein Feld hat die Koordinaten (Zeile, Spalte), das X oben links hat die Koordinaten (0, 0). Die Tiefensuche legt die Folgezustände so auf den Stack, dass sie in der Reihenfolge oben, unten, links, rechts wieder herausgenommen werden. (Die Punkte sollen das Abzählen erleichtern.)

   Die Tiefensuche wird angehalten, nachdem der Zustand (1, 1) untersucht wurde, seine Folgezustände liegen also schon auf dem Stack. Wie viele Zustände wurden untersucht, wie viele befinden sich in der frontier? Zeichne in das große Labyrinth die Zustände in der frontier mit einer Tilde ein, die untersuchten Zustände mit einem Punkt.

```
XXXXXXXXXX
X........X
X..E.....X
X..XXXX..X
X....S...X
XXXXXXXXXX
```

```
XXXXXXXXXX
X........X
X........X
X........X
X........X
XXXXXXXXXX
```

8. (1 Punkt) Zeichne ein Labyrinth, in dem die Tiefensuche einen Weg der Länge 7 findet.

9. (3 Punkte) Beim 8-Puzzle ist ein Zustand ein Tupel mit den Zahlen von links oben nach rechts unten, die Leerstelle ist die 0.

```
def nextstates(state):
    tmp = []
    i = state.index(0)
    for d in [-3, 3, -1, 1]:
        j = i + d
        if 0 <= j < 9 and (i % 3 == j % 3 or i // 3 == j // 3):
            a = list(state)
            a[i], a[j] = a[j], a[i]
            tmp.append(tuple(a))
    return tmp
```

   (a) Gib die Folgezustände von `(3, 1, 2, 0, 4, 5, 6, 7, 8)` in der Reihenfolge an, in der `nextstates` sie liefert.

   (b) Für $d = -1$ ist $j = 2$ ein gültiger Index. Warum wird trotzdem kein Folgezustand erzeugt?

10. (3 Punkte) Ein String aus den Zeichen M und O soll mit möglichst wenigen Operationen in den String MOO umgewandelt werden. Erlaubt sind: den ersten oder letzten Buchstaben durch den anderen ersetzen (M durch O oder O durch M) und den ersten oder letzten Buchstaben löschen.

   (a) Gib alle Folgezustände von MOMO an.

   (b) Wie viele Operationen braucht man mindestens für OMOMO? Gib die Zwischenschritte an.

   (c) Warum kann MMM nie zu MOO werden?

11. (3 Punkte) Es gibt einen 3-Liter-Krug und einen 5-Liter-Krug, beide sind leer. Ein Zug ist: einen Krug ganz füllen, einen Krug ganz leeren oder Wasser von einem Krug in den anderen gießen, bis der eine leer oder der andere voll ist. Ein Zustand ist das Paar (Liter im 3-Liter-Krug, Liter im 5-Liter-Krug).

   (a) Gib alle Folgezustände von (3, 2) an.

   (b) Wie viele Züge braucht man mindestens, bis im 5-Liter-Krug genau 4 Liter sind? Gib die Zustände an.

12. (4 Punkte) Auf der Westseite eines Flusses sind ein Mensch, ein Wolf, eine Ziege und ein Kohl. Nur der Mensch kann die Fähre steuern und höchstens ein Tier oder den Kohl mitnehmen. Ohne den Menschen dürfen Wolf und Ziege nicht zusammen auf einer Seite bleiben, Ziege und Kohl auch nicht. Alle sollen auf die Ostseite.
    (a) Wie kann man einen Zustand modellieren? Gib Start- und Zielzustand an.
    (b) Gib alle erlaubten Folgezustände des Startzustands an.
    (c) Wie viele Überfahrten sind mindestens nötig? Gib eine mögliche Reihenfolge an.

13. (5 Punkte) Auf der Westseite eines Flusses sind 3 A und 3 B. In das Boot passen höchstens 2, bei jeder Überfahrt muss mindestens ein A oder B im Boot sein. Ist auf einer Seite mindestens ein A, dann dürfen dort nicht mehr B als A sein. Alle sollen auf die Ostseite.

    Löse die Aufgabe mit der Breitensuche aus dem Unterricht (`bfs`, `reconstructPath`). Modelliere einen Zustand und schreibe die Funktionen `nextstates` und `goaltest`. Wie viele Überfahrten sind mindestens nötig?

14. (5 Punkte) Gegeben sind drei Strings `s1`, `s2` und `s3` aus den Zeichen 1 und 2. Ein Zug besteht darin, das letzte Zeichen eines Strings zu entfernen und hinten an einen anderen String anzuhängen. Ist das neue Zeichen gleich dem bisher letzten Zeichen, verschmelzen die beiden zu einem.

```
s1, s2, s3 = '121', '1', ''
# Zug: letztes Zeichen von s1 an s2 anhängen
s1, s2, s3 = '12', '1', ''    # die beiden 1 sind verschmolzen
```

Ziel ist, dass `s1` und `s2` nur noch aus einem Zeichen bestehen und `s3` leer ist.

Modelliere einen Zustand und schreibe die Funktionen `nextstates` und `goaltest` für die Breitensuche aus dem Unterricht. Wie viele Züge braucht man mindestens für `s1, s2, s3 = '121', '12', ''`?