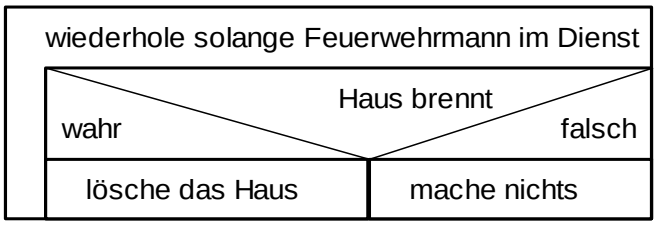
Phase 3 – Personensuche: Spielbericht

Struktogramme helfen uns Menschen zwar die Algorithmen zu verstehen, aber ein Computer kann diese nicht verarbeiten. Er braucht zeilenweise Befehle, die er der Reihe nach bearbeiten kann. Diese zeilenweise Befehle heißen **Code**. Diesen Code gibt es dann in verschiedenen Sprachen(z.B. Java, C++). Umgangssprachlichen Code, den wir verwenden möchten, nennt man **Pseudocode**. Praktischerweise ist es jedoch möglich aus Struktogrammen Code bzw. Pseudocode zu machen. Hier ein Beispiel:

Pseudocode	Struktogramm

`Wiederhole(solange Feuerwehrmann im Dienst){`
`  wenn(Haus brennt = WAHR){`
`  lösche das Haus`
`  }`
`  wenn(Haus brennt = FALSCH){`
`  mache nichts;`
`  }`
`}`
	

Befehle des Pseudocodes:

```
wenn (xx=WAHR) dann
{}
wiederhole (solange)
{}
```

Zu Beachten:
Jede Klammer, die geöffnet wird, muss auch wieder schließen.

Aufgabe 3.1:
Guckt euch diesen Pseudocode (der sogenannten Tiefensuche) an und versucht gemeinsam diesen Algorithmus auf dem Spielfeld durchzuführen.

```
mache alle Zimmer weiß
wiederhole(solange es weiße Zimmer gibt){
      färbe aktuelles Zimmer schwarz
      wenn(es gibt weißes Nachbarzimmer = WAHR){
            gehe in das weiße Nachbarzimmer mit kleinster Nummer
      }
      wenn(es gibt weißes Nachbarzimmer = FALSCH){
            gehe ein Zimmer entsprechend deines gegangenen Weges zurück
      }
}
```

Aufgabe 3.2:
Versucht selber einen Algorithmus zu finden, mit dem man die vermisste Person suchen könnte. Wenn ihr eine Idee habt notiert diese im Pseudocode.

_____

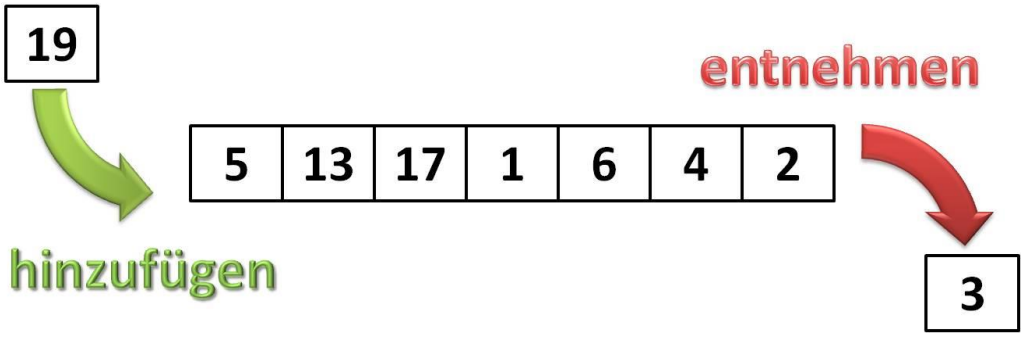
_____

_____

Nutzt auch die Rückseite des Blattes!

Phase 3 – Personensuche: Spielbericht

Ein weiterer Algorithmus ist die Breitensuche. Um diese zu benutzen wird eine Warteschlange benutzt, die genauso funktioniert wie im Supermarkt.
Einer Warteschlange können Elemente (z.B. Zimmernummern) hinzugefügt (19) und entnommen (3) werden. Wie könnt ihr der folgenden Grafik entnehmen:



Algorithmus der Breitensuche:

```
mache alle Zimmer weiß
Speichere Nummer des Startzimmers in der Warteschlange
wiederhole(solange Warteschlange nicht leer und Person nicht gefunden){
gehe in das erste in der Warteschlange gespeicherte Zimmer
entferne dieses aus der Warteschlange
      färbe dieses schwarz

wenn(es gibt weiße Nachbarzimmer = WAHR){
füge diese weißen Nachbarzimmer der Größe nach zur Warteschlange hinzu
      }
}
```

Weiter geht es mit folgendem Gruppenspiel:

1) Spieler 1 zieht eine AlgoCard, um seinen Suchalgorithmus zu erfahren. Sein linker Nebenspieler würfelt in welchem Raum sich die zu suchende Person befindet. Alle Mitspieler, außer Spieler 1, dürfen wissen, wo sich die verletzte Person befindet.
2) Alle anderen Mitspieler raten wie viele Zimmer/Knoten besucht werden müssen, um die Person zu finden. Die geschätzte Zahl muss im Spielbericht
3) Spieler 1 sucht entsprechend des Algorithmus die verletzte Person. Dabei werden die besuchten Zimmer gezählt.
4) Punkte:
   Spieler 1 erhält 10 Punkte für das Finden der Person.
   Die Spieler die am Anfang die Zahl richtig geschätzt haben, bekommen fünf Punkte.
5) Alle Spieler tragen die Schrittanzahl und ihre Punkte auf dem Spielbericht ein.
6) Es geht im Uhrzeigersinn weiter.
7) Der Gewinner ist der, der am meisten Punkte am Ende des Spiels hat.

Phase 3 – Personensuche: Spielbericht

Übersichtstabelle

	Benötigte Schritte							Durchschnitt
Breitensuche								
Tiefensuche								
Eigener Algorithmus								

Spielstand

Name des Spielers	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12	13.	14.	Endstand
Maximilian	*25* *5*	*X* *10*	*14* *5*	*30* *0*											*20*

Erklärung: Spieler Maximilian hat in der ersten Runde 25 Schritte geschätzt. Da er der beste Schätzer war, bekam er in Runde 1 5 Punkte. In Runde 2 war er selber am Zug und konnte deshalb nicht schätzen. Dafür, dass er am Zug war, bekam er jedoch 10 Punkte.