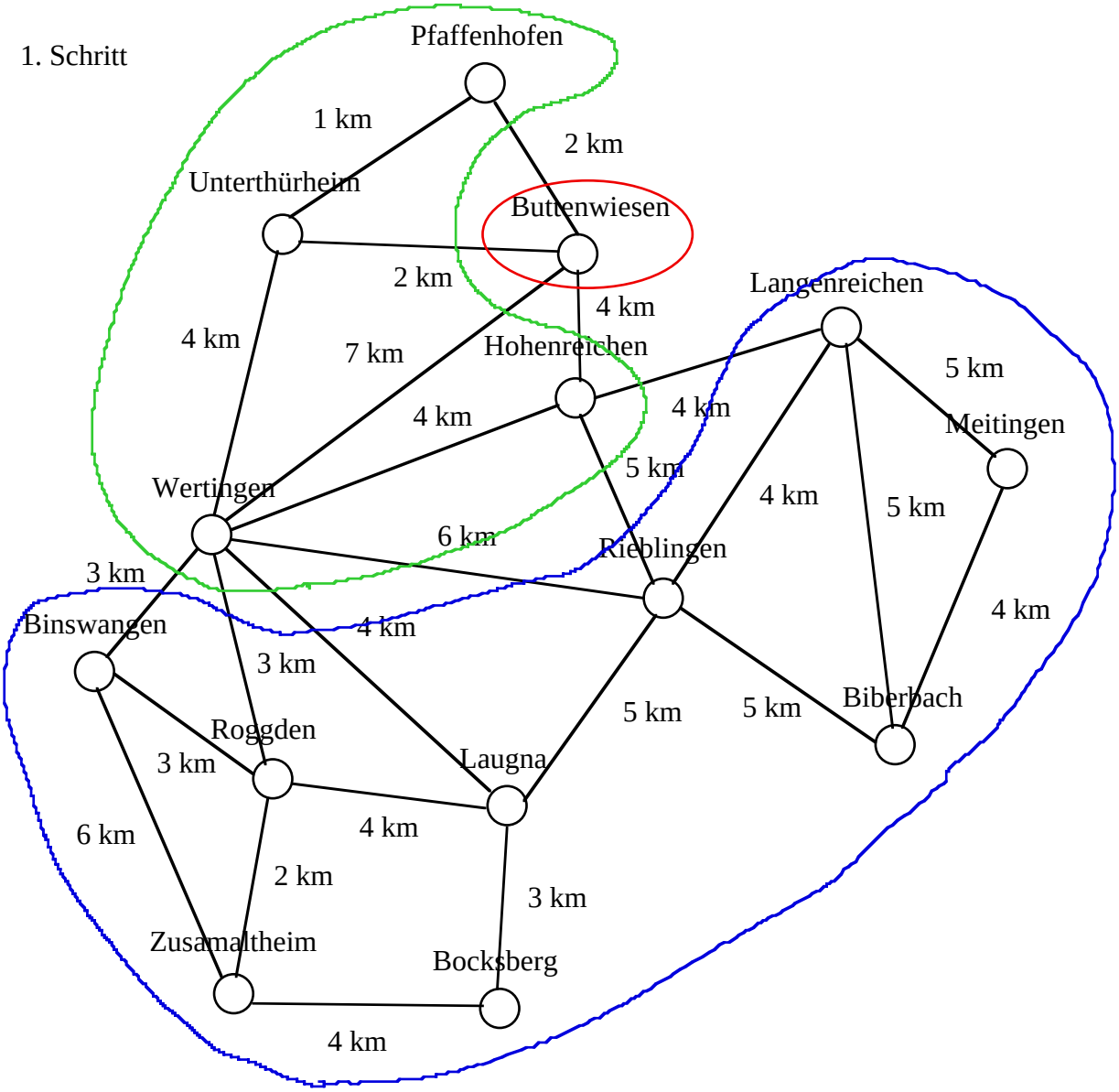
I.3 Dijkstra - Algorithmus

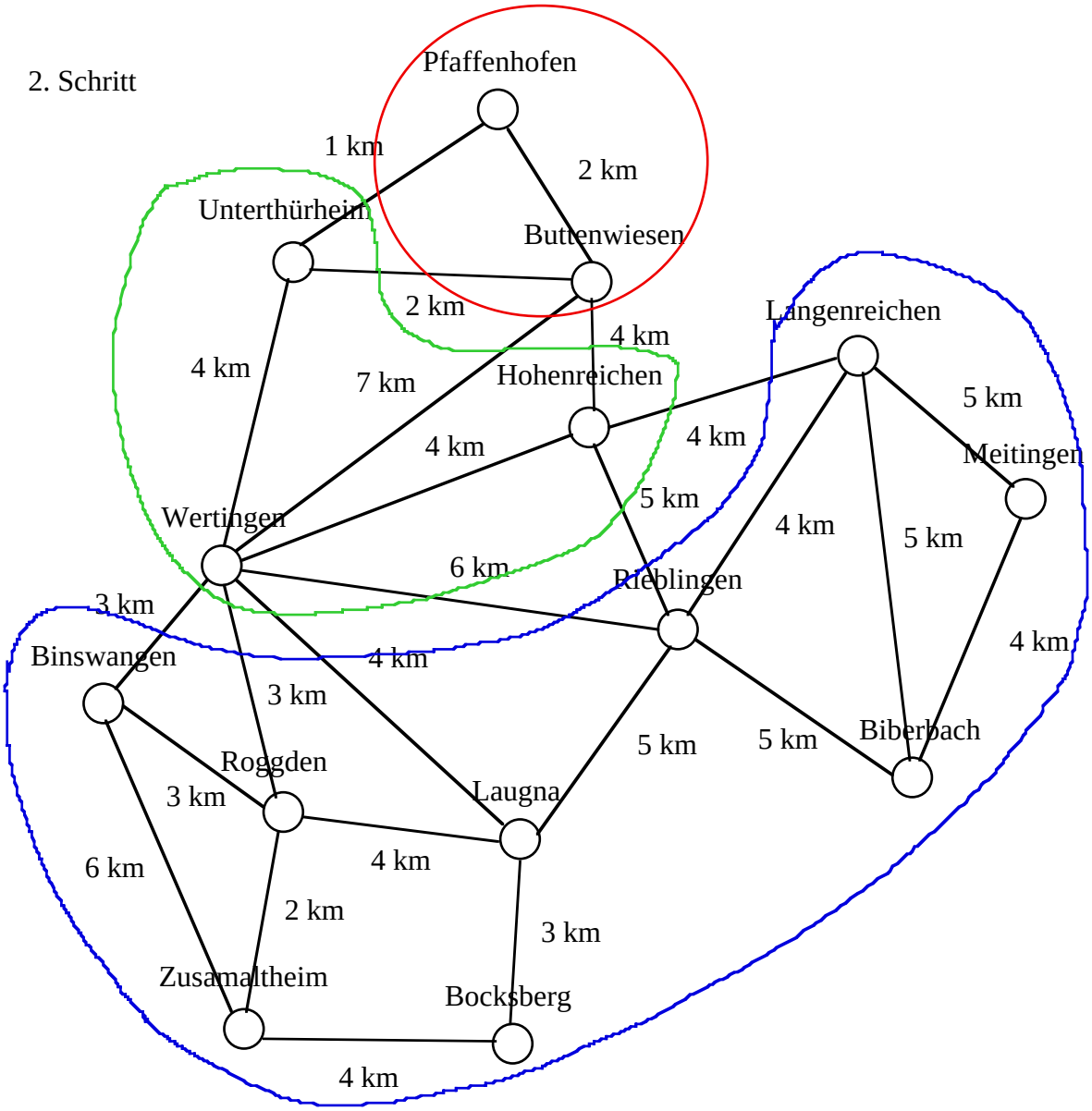
Nachdem wir nun gesehen haben, dass das Backtracking - Verfahren sehr aufwendig und für große Ortszahlen ungeeignet ist, wollen wir nun ein effektiveres Verfahren vorstellen. Der Dijkstra - Algorithmus ermöglicht es mit geringerem Aufwand den schnellsten Weg zu finden. Dagegen kann er nicht für andere Probleme (Acht-Damen-Problem oder Springer-Problem) eingesetzt werden. Das Backtracking - Verfahren ist allgemeiner einsetzbar.

Verfahren

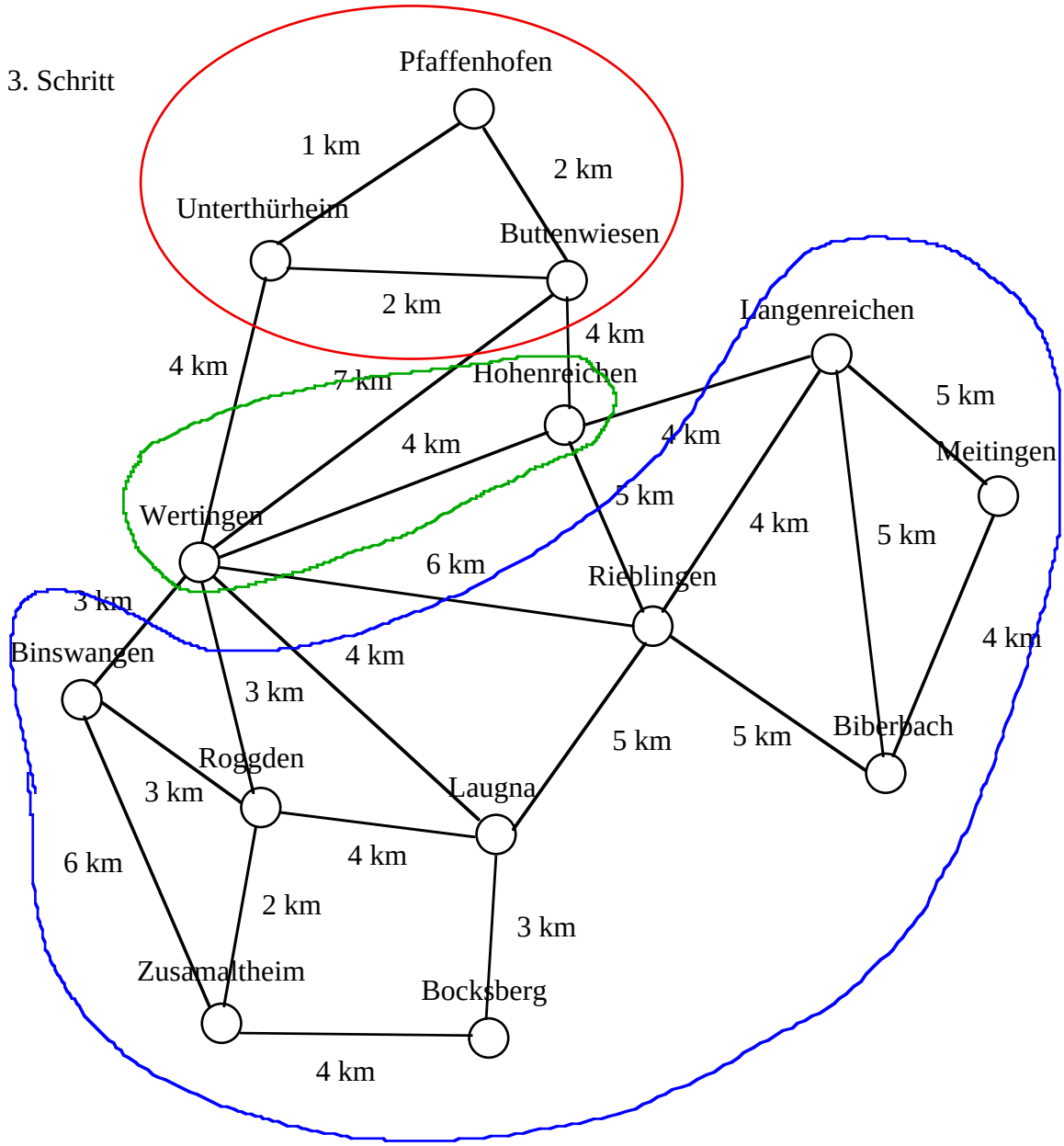
Dijkstras Algorithmus findet den minimalen Weg zwischen zwei Knoten eines Graphen. Die zugrundeliegende Idee besteht darin, die Knoten in der Reihenfolge ihrer Abstände zu betrachten. Dazu wird die Menge der Orte in drei Teile eingeteilt. Zum einen die Orte, die bereits erreicht sind, dann die Orte die im nächsten Schritt von den erreichten Orten aus erreichbar sind. Die letzte Menge bilden die Orte die im nächsten Schritt nicht erreichbar sind. Der Vorteil des Dijkstra Algorithmus besteht darin, dass ein Ort automatisch auf dem kürzesten Weg erreicht wird.



Betrachten wir nun das Verfahren an einem Beispiel.
Es soll der kürzeste Weg von Buttenwiesen nach Biberbach gesucht werden.
Am Anfang ist nur der kürzeste Weg nach Buttenwiesen gefunden. D.h. die erste Menge der erreichten Orte (rot) besteht nur aus Buttenwiesen. Die zweite Menge der im nächsten Schritt direkt erreichbaren Orte besteht aus Pfaffenhofen, Unterthürheim, Hohenreichen und Wertingen (grün). Alle anderen Orte sind in diesem Schritt nicht erreichbar und bilden damit die dritte Menge (blau).
Wir beginnen wieder in Buttenwiesen. Nun sucht man den kürzesten Weg. Dies wäre hier der Weg nach Pfaffenhofen oder nach Unterthürheim. Man wählt nun einen der beiden und hat damit den kürzesten Weg zu diesem Ort gefunden, z.B. Pfaffenhofen. Pfaffenhofen wird nun in die Menge der erreichten Orte aufgenommen. In der zweiten Menge sind nun alle Orte, die von Buttenwiesen und Pfaffenhofen direkt erreicht werden.

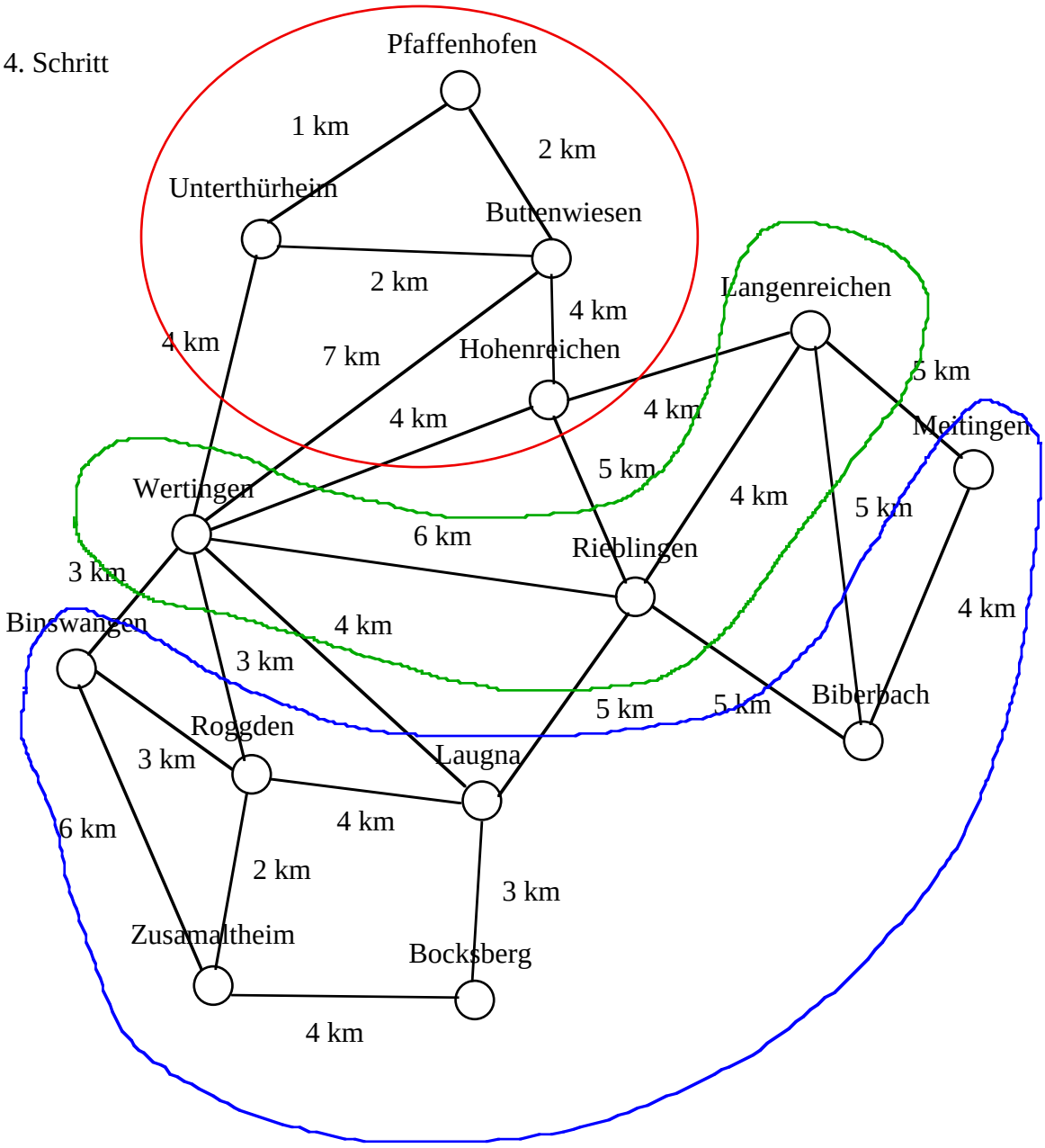


Wieder wählt man den Ort aus, der von Buttenwiesen auf der kürzesten Strecke erreichbar ist. In diesem Fall Unterthürheim.

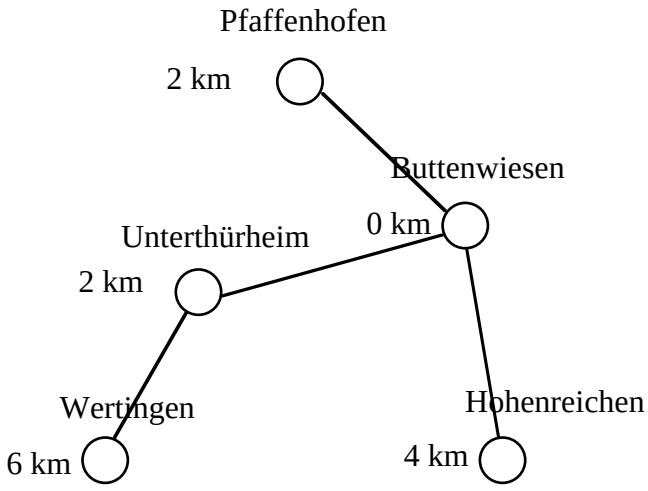


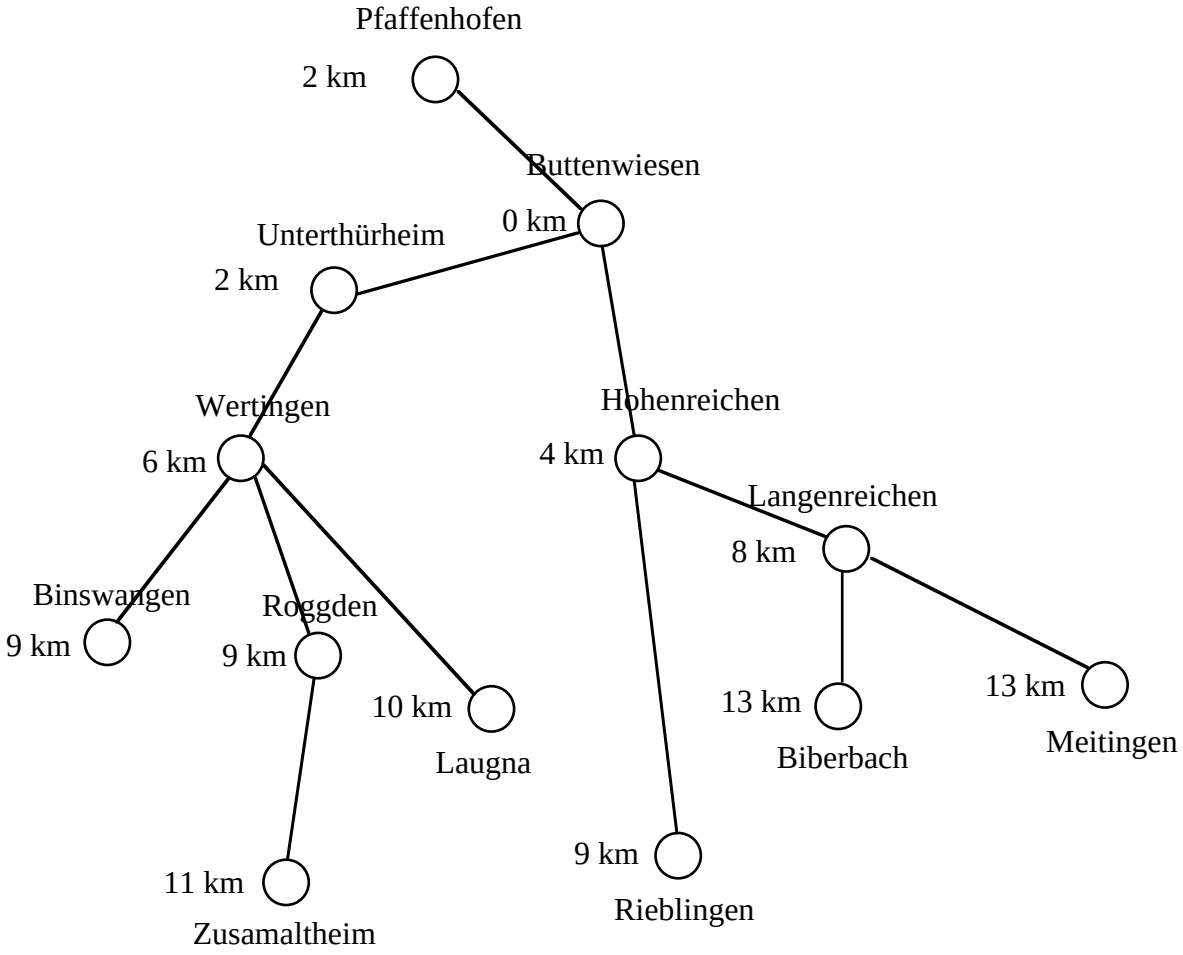
Der untersuchte Bereich besteht nun aus den Orten Buttenwiesen, Unterthürheim und Buttenwiesen. Der nächsten Ort der nun in die erreichte Menge aufgenommen wird ist Hohenreichen. Eigentlich ist der Weg nach Unterthürheim über Pfaffenhofen kürzer als der Weg nach Hohenreichen, aber Unterthürheim gehört bereits zu den erreichten Orten und muss daher nicht mehr betrachtet werden (im Gegensatz zum Backtracking Verfahren). Hohenreichen wird nun in die Menge der erreichten Orte aufgenommen und es ergibt sich die nachfolgende Situation.

4. Schritt



Der nächste Ort ist nun Wertingen, der von Unterthürheim aus erreicht wird. Damit hat der nun aufgebaute Graph nebenstehendes Aussehen.





Nun ist Langenreichen mit einer Entfernung von 8 km an der Reihe. Danach folgen mit 9 km Binswangen, Roggden und Rieblingen. Als nächstes wird mit 10 km Laugna eingefügt und dann mit 11 km Zusamaltheim. Schließlich mit 13 km Meitingen und Biberbach. Da man nun am Zielort angekommen ist hat man auch den kürzesten Weg gefunden. Dies folgt direkt aus dem Verfahren von Dijkstra, da immer mit steigender Wegstrecke die Orte gesucht werden, die mit dieser Kilometerzahl erreicht werden können. Hat man damit zum ersten Mal einen Ort erreicht, so ist dies auch die kürzeste Strecke.
Vergleicht man nun den obigen Graphen, so enthält er wesentlich weniger Knoten als der Suchbaum eines Backtracking - Verfahrens, Damit wird das Ziel auch wesentlich schneller erreicht. Die Knotenzahl kann hier die Zahl der Orte nicht übersteigen.

Benötigte Datenstrukturen

Um nun das Verfahren auf den Rechner zu übertragen, muss zuerst die benötigte Datenstruktur festgelegt werden. Zuerst braucht man die drei Mengen. Einmal die ereichten Orte, dann die erreichbaren Orten und die nicht erreichbaren Orte. Um dies zu realisieren, speichert man bei jedem Ort nicht nur die Kennung ab, sondern auch die Zugehörigkeit zur Menge. Dazu verwenden wir die drei booleschen Variablen „erreicht“, „erreichbar“ und „nicht_erreichbar“. Der Weg kann diesmal nicht im Voraus gespeichert werden, da nicht bekannt ist, welcher Weg zum Ziel führt. Auch ist bei einem Ort mit mehreren Zweigen nicht klar, welcher gewählt werden muss. Daher merkt man sich bei jedem Ort den Vorgänger, d.h. den Ort von dem man zu diesem Ort kommt. Daneben muss bei jedem Ort auch noch die Entfernung zum Startpunkt abgespeichert werden.

Dazu benötigen wir einen neuen Datentyp, der diese fünf Daten umfasst. Die Kennung kann bereits als Index des Feldes verwendet werden.

```
TYPE

    tKnoten = RECORD
                Gesamt     : INTEGER;
                Vorgaenger : INTEGER;
                erreicht : BOOLEAN;
                erreichbar : BOOLEAN;
                nicht_erreichbar : BOOLEAN;
              END; (* tKnoten *)

    tKnotenfeld : ARRAY cMaxOrte OF tKnoten;

VAR

    Mengen : tKnotenfeld;
        (* Die drei Mengen des Dijkstra - Algorithmus *)
```

Benötigte Prozeduren

In Kurzer_Weg müssen der nun Start- und Zielort eingelesen und die Variablen wieder vorbelegt werden. Das Feld besucht besteht am Anfang nur aus den Anfangsort, während alle anderen Ortskennungen in dem Feld nicht_besucht stehen. Am Schluss erfolgt die Ausgabe des Weges.

PROCEDURE Kurzer_Weg (Entfernungsmatrix : tMatrix, Orte : tOrte,
Ortszahl : INTEGER);

Initialisiere (Mengen, Ortszahl)
Eingabe (Startort)
Eingabe (Zielort)
Startkennung := Kennung (Startort, Ortszahl, Orte)
Zielkennung := Kennung (Zielort, Ortszahl, Orte)
Suche_Weg (Startkennung, Zielkennung)
Weg_ausgeben (Startkennung, Zielkennung, Mengen)

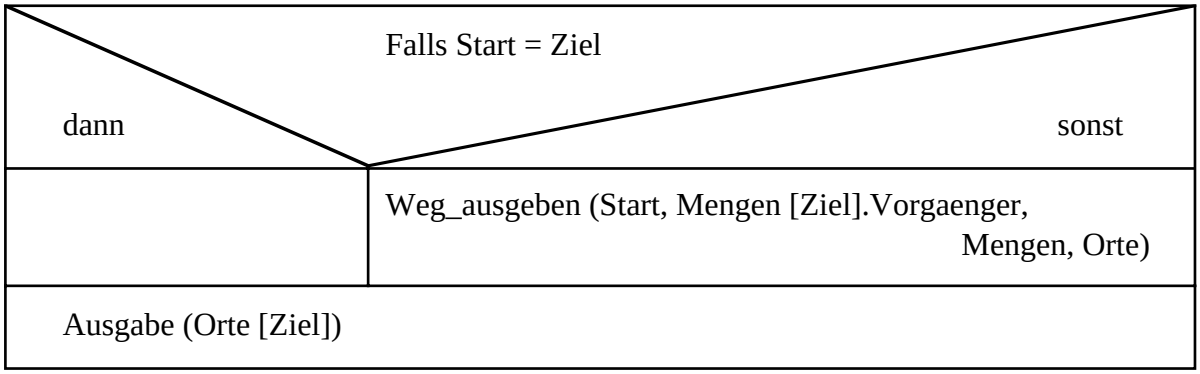
Bei der Initialisierung der Mengen werden alle Orte in die nicht erreichbare Menge eingetragen. Das heisst: „erreicht“ und „erreichbar“ wird auf FALSE gesetzt und „nicht_erreichbar“ auf TRUE. „Gesamt“ wird mit einer möglichst großen Zahl belegt, da dann nach den kleinsten Werten gesucht wird. Der Vorgänger muss nicht eingetragen werden.

PROCEDURE Initialisiere (Mengen)

Für i von 0 bis Ortszahl -1 tue
Mengen[i].Gesamt := 10000
Mengen[i].erreicht := FALSE
Mengen[i].erreichbar := FALSE
Mengen[i].nicht_erreichbar := TRUE

Die Ausgabe des Weges ist nun etwas ungewöhnlich, da der Weg an sich nicht abgespeichert ist. Zu jedem Ort ist nur sein Vorgänger bekannt. Daher muss man von hinten anfangen. D.h. bevor man den Zielort ausgibt muss man den Weg bis zum Vorgänger ausgeben. So etwas nennt man einen rekursiven Aufruf. Der rekursive Aufruf wird beendet, wenn man am Startort angekommen ist.

PROCEDURE Weg_ausgeben (Start, Ziel : INTEGER; Mengen : tKnotenmenge,
Orte : tOrte),



Nun fehlt nur noch die Wegesuche. Betrachten wir dazu noch einmal das Verfahren.

Verfahren der Wegesuche
Nachdem ein Ort in die Menge der erreichten Orte eingetragen wurde, betrachtet man nun alle Orte die eine direkte Verbindung zu diesem Ort haben (Entfernungsmatrix [Neuer_Ort, j] > 0). Ist dieser Ort bereits in der Menge der erreichten Orte, so geschieht nichts. Ist er in der Menge der nicht_erreichbaren Orte, so wird er in die Menge der erreichbaren Orte übernommen, die Gesamtstrecke berechnet und der Vorgänger eingetragen. Ist er in der Menge der erreichbaren Orte, so wird überprüft ob die neue Gesamtstrecke kürzer ist als die bisherige und gegebenenfalls die Gesamtstrecke und der Vorgänger geändert. Anschließend wird aus der Menge der erreichbaren Orte der Ort mit der kürzesten Gesamtstrecke ausgesucht. Dieser wird nun in die Menge der erreichten Orte eingetragen und aus der Menge der erreichbaren Orte entfernt. Dies wiederholt sich bis der neue erreichte Ort der Zielort ist.

Die Wegesuche wird nun in mehreren Teilen realisiert. In der Prozedur Suche_Weg wird zuerst der Startort initialisiert, d.h in die Menge der ereichten Orte eingetragen und die Gesamtstrecke auf null gesetzt. Danach muss die Menge derreichbaren Orte bearbeitet werden. Anschließend wird aus dieser Menge der Ort mit der kürzesten Gesamtstrecke ausgesucht und in die Menge der erreichten Orte übernommen. Dies wiederholt sich bis der ausgewählte Ort der Zielort ist.

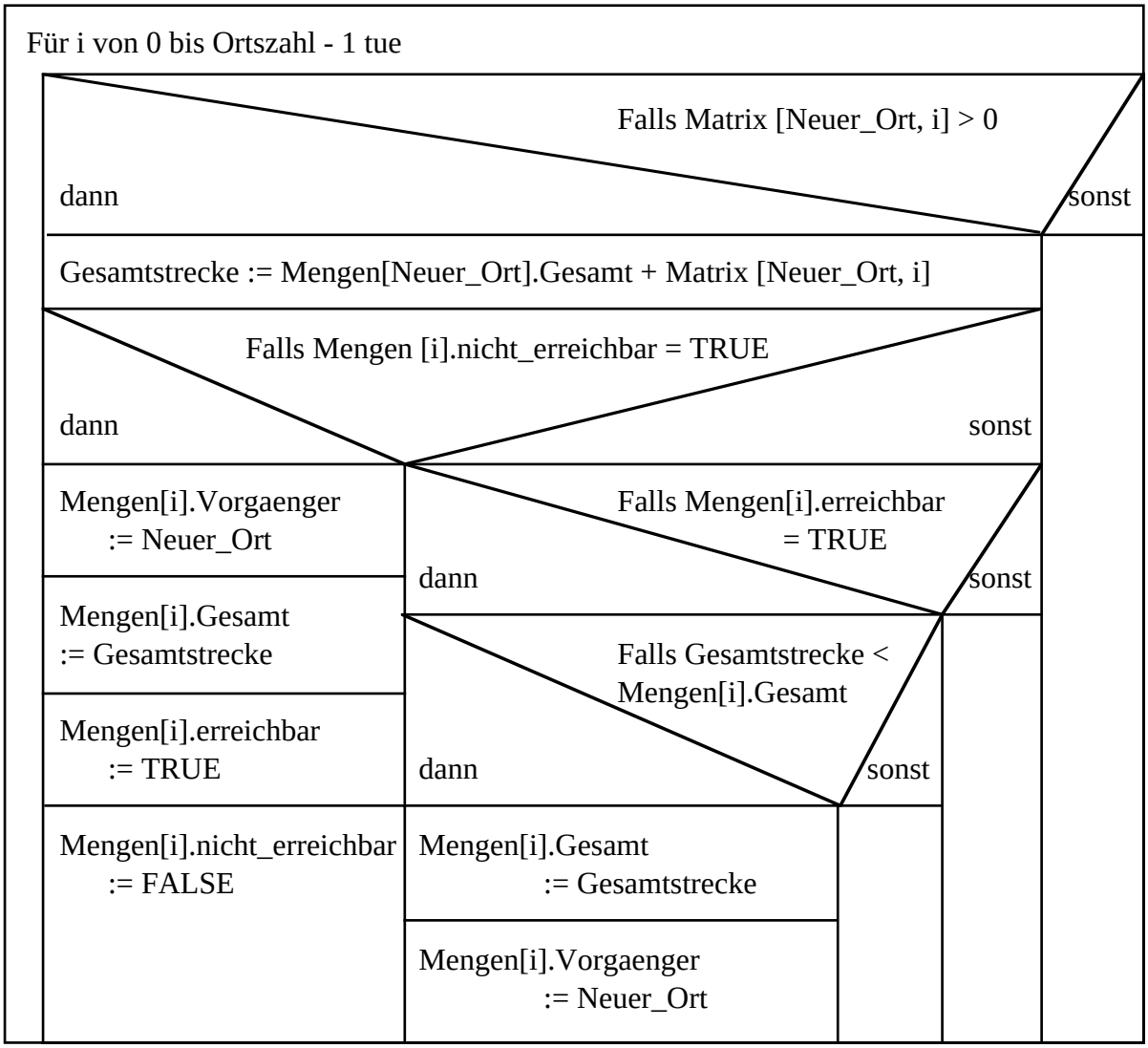
PROCEDURE Suche_Weg (Start, Ziel : INTEGER; Mengen : tKnotenfeld);

```
Mengen [Start].Gesamt := 0
Mengen [Start].erreicht := TRUE
Mengen [Start]. nicht_erreichbar := FALSE
Neuer_Ort := Start
Solange Neuer_Ort ≠ Ziel tue
    Erreichbare_Orte_bestimmen (Neuer_Ort, Ortszahl, Entfernungsmatrix)
    Neuer_Ort := Minimale_Gesamtstrecke_finden (Ortszahl)
    Mengen [Neuer_Ort].erreicht := TRUE
    Mengen [Neuer_Ort].erreichbar := FALSE
```

Die Prozedur darf nicht mit einer „Wiederhole bis …“ - Wiederholung realisiert werden. Bei gleichem Start- und Zielort gibt es sonst Probleme. Daher muss vor der Wiederholung überprüft werden, ob das Ziel erreicht wurde.

Betrachten wir nun die Bearbeitung der erreichbaren Menge nach dem oben beschriebenen Verfahren.

PROCEDURE Erreichbare_Orte_bestimmen (Neuer_Ort, Ortszahl : INTEGER;
Matrix : Def.tMatrix);



Als letztes muss nun aus der Menge der erreichbaren Orte der mit der kürzesten Gesamtstrecke ausgewählt werden.

Die Bestimmung des schnellsten Weges erfolgt dann nach dem gleichen Prinzip. Nur wird statt der Entfernungsmatrix die Zeitmatrix verwendet.

Bewertung des Dijkstra - Verfahrens.

Das Dijkstra - Verfahren führt wesentlich schneller zum Ziel, da nur ein Weg gefunden wird und dieser bereits der kürzeste ist. Es müssen nicht wie beim Backtracking-Verfahren alle Wege durchsucht werden. Um nun eine Abschätzung zu erhalten, zählen wir wieder die betrachteten Knoten.

Beim Djikstra - Verfahren werden nacheinander die Orte in die erreichte Menge einsortiert. Bei n Orten bedeutet dies n-mal das gleiche Verfahren. Danach müssen jedesmal alle Orte betrachtet werden und ihre Einsortierung in die Menge der erreichbaren Orte überprüft werden. Im schlechtesten Fall müssen dazu alle Orte betrachtet werden. Danach muss der Ort mit der minimalen Gesamtstrecke aus der erreichbaren Menge ausgewählt werden. Dabei müssen wieder alle Orte betrachtet werden. Dies könnte aber auch beim Einsortieren erfolgen. Daher werden pro Durchgang n Orte betrachtet. Für den gesamten Dijkstra-Algorithmus sind es dann $n \cdot n = n^2$ Orte.

Dies ist wesentlich besser als der Aufwand beim Backtracking - Verfahren. Allerdings ist das Backtracking - Verfahren allgemeiner und auf mehr Probleme anzuwenden.