Vorgehensweise (Dijkstra)

1. Die direkte Implementation basierend auf dem Pseudocode. Die OK, Entf und Vorg werden entsprechend Array-artig gehalten und aktualisiert. Diese Implementation ist vor allem für Erlang sehr ineffizient, stellt aber erst einmal eine gute Basis dar.
2. Um nicht mehr die Vergleiche aus 1 zu haben, ob ein Vertex laut OK = false ist, kombiniere ich diese Information in einer Liste [{V, Entf, Vorg}, {V, Entf, Vorg}, ...]. Das Setzen eines Elements auf true wird dann durch Löschung aus der Liste realisiert. Dadurch sinkt die Anzahl der Vergleiche bei A1 von |V| auf |V|-i, wobei i die Anzahl der bisherigen Iterationen darstellt.
3. Da die Abfrage aus A3 nur die Vertices aktualisiert, die laut OK = false sind, es somit einen kürzeren Weg zu diesen geben kann, muss bisher, nachdem eine Edge vh gefunden hat, geprüft werden, ob es (in 2) in der Liste ist. Um sich diesen Abgleich zu sparen, löscht man beim OK = true setzen nicht nur aus der Liste, sondern auch aus dem Graphen. Wichtig ist dabei, dass die Löschung erst nach A3 stattfindet, weil man unmittelbar nach dem Setzen von OK = true ja mit H weiterarbeitet.
4. Bei A3 wird bisher die Entf aus der Liste gesucht. Die Optimierung hier ist nun dies direkt in den Graphen assoziiert mit dem Vertex zu speichern, wobei hier der Gewinn davon abhängt, ob das getValV und setValV schneller ist, als das Iterative in der Liste und da ich keinen Zugriff auf die Implementation des Graphen habe, werden die Messungen dies zeigen.
5. Da die Liste aus 2 zu Anfang nur ein Element, nämlich beim StartVertex, mit Entf = 0 hat und die anderen bei Entf = inf liegen, ist es nicht nötig die gesamte Liste für einen Vertex H zu durchsuchen. Auch beim zweiten Durchgang ändert sich bei der Liste nur für die vh-Edges die Entf, somit kommen nur diese als nächstes H infrage. Die Idee ist nun, dass man die Liste aus 2 mit [{11, 0, 11}] initialisiert (für graph\_03) und dann jeweils Elemente einfüllt, sodass weniger Vergleiche für H nötig sind.
6. Auch wenn die Liste kleiner ist, so muss sie dennoch komplett durchsucht werden, um H zu finden. Weniger Vergleiche sind z.B. mit einem MinHeap möglich.