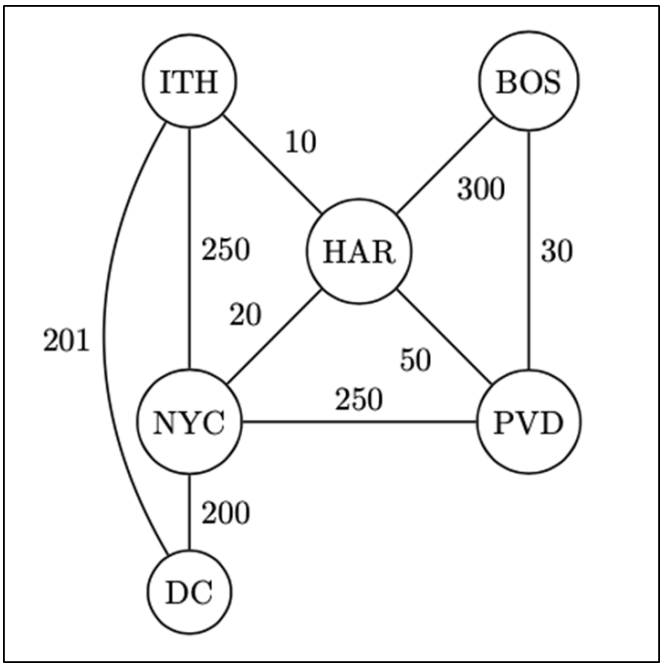
Dijkstra’s algorithm



Find the cheapest route from Boston to NYC

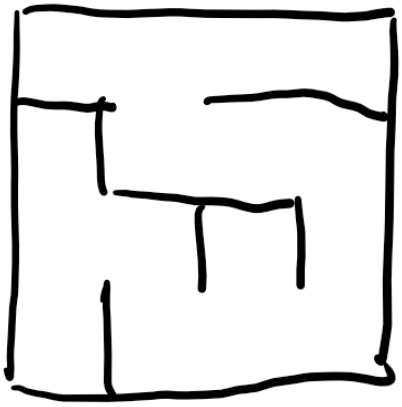
toCheckPQ

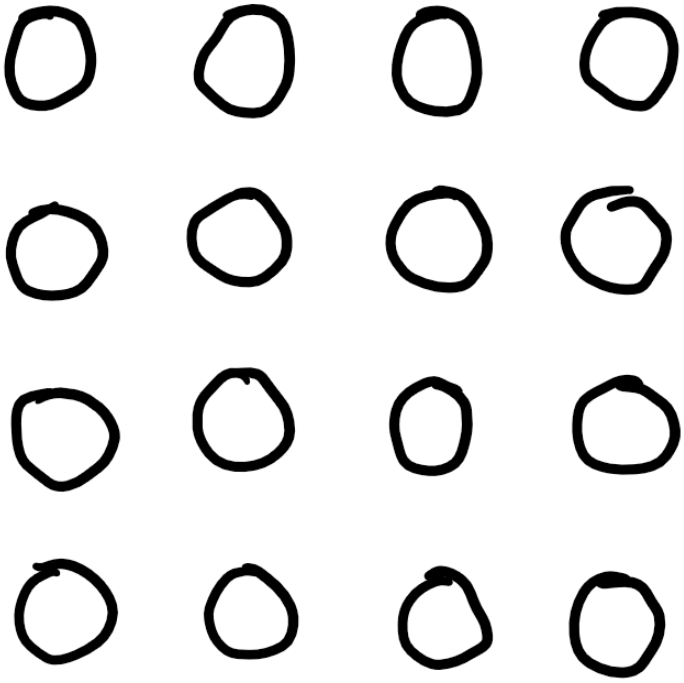
Name	Route Distance

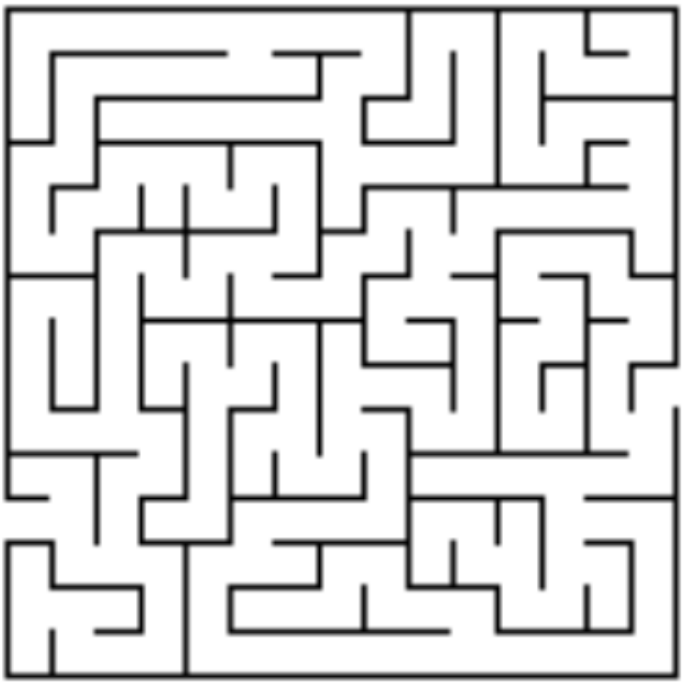
Order in which items were removed from toCheckPQ:

cameFrom

Another application of queue vs stack vs priority queue







BFS/DFS Pseudocode

```
while (! toCheck.isEmpty()) {
  Vertex<T> checkingVertex = toCheck.removeLast(); // removeFirst() for BFS
  if (dest.equals(checkingVertex)) {
    return true;
  }
  for (Vertex<T> neighbor : checkingVertex.getOutgoing()) {
    if (!visited.contains(neighbor)) {
      visited.add(neighbor);
      toCheck.addLast(neighbor);
    }
  }
}
```

Dijkstra Pseudocode

```
toCheckQueue = V (prioritized on routeDist)
cameFrom = empty map

for v in V:
  v.routeDist = inf
source.routeDist = 0

while toCheckQueue is not empty:
  checkingV = toCheckQueue.removeMin()
  for neighbor in checkingV's neighbors:
    if checkingV.routeDist + cost(checkingV, neighbor) < neighbor.routeDist:
      neighbor.routeDist = checkingV.routeDist + cost(checkingV, neighbor)
      cameFrom.add(neighbor -> checkingV)
      toCheckQueue.decreaseValue(neighbor)

backtrack from dest to source through cameFrom
```