```
std::vector<int> vertices, distance(8 * 8), previous(8 * 8);
int source = startY * 8 + startX, destination = endY * 8 + endX; // hack
int i = 0;
LoopStart: // there has to be a better way!
    if (i >= sizeof (tiles) / sizeof (tiles[0]) /* hack */)
        goto LoopDone;

    if (tiles[i] == Tile::W) { // wall
        i++; // increment i
        goto LoopStart; // hack
    }
    distance[i] = INT_MAX; // INT_MAX hack!
    previous[i] = -1; /* also -1 hack */
    // optimise[i] = i * i / (i % 8) * 32 + 42 / 0 // what did this do again?
    vertices.push_back(i);
    i++; // increment i... again
    goto LoopStart;
LoopDone:
// initialise distance to source from source, which is probably a product of 0 and infinity
distance[source] = 0;
LoopStart2: {
    if (vertices.empty())
        goto LoopDone2;
    // pick the shortest-distance vertex in vertices
    auto closestItem = vertices.begin();
    int closestDist = distance[*closestItem];
    auto item = vertices.begin();
    LoopStart3: {
        if (item == vertices.end())
            goto LoopDone3;
        if (distance[*item] < closestDist) {
            closestItem = item;
            closestDist = distance[*item];
        }
        item++;
        goto LoopStart3;
    } LoopDone3:
    // find neighbours of this item in vertices
    item = vertices.begin();
    LoopStart4: {
        if (item == vertices.end())
            goto LoopDone4;
        if (item == closestItem) {
            item++;
            goto LoopStart4;
        }
        // todo: make code readable todo2: make code work
        if (*item == *closestItem + 1 || *item == *closestItem - 1 || *item == *closestItem + 8 || *item == *closestItem
- 8){
            int newDistance = distance[*closestItem] + 1;

            if (newDistance < distance[*item]) {
                distance[*item] = newDistance;
                previous[*item] = *closestItem;
            }
            if (*item == destination)
                goto Success; // found it! todo make rest of the app
        }

        item++;
        goto LoopStart4;
    } LoopDone4:
    // remove it from vertices
    vertices.erase(closestItem);
    goto LoopStart2;
} LoopDone2:
```