CS380
Artificial Intelligence for Games

Dijkstra’s Search In Implicit Graph

Outline

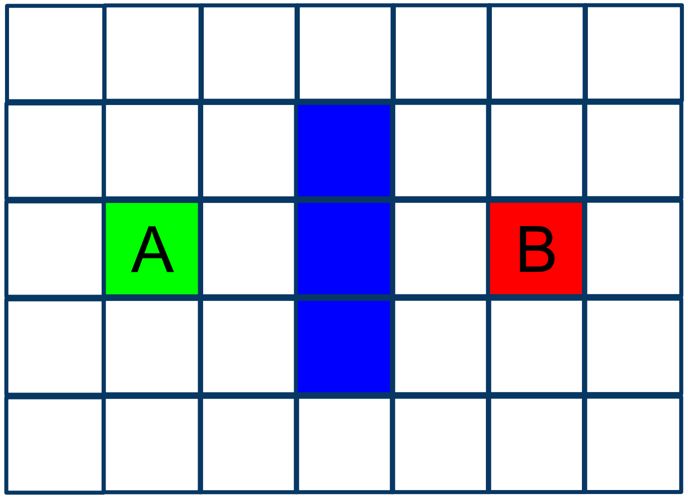
- Introduction
- Dijkstra’s Path Finding Algorithm Outline
- Starting The Search
- Path Scoring
  - How To Calculate G?
- Notations
- Searching Loop
- When To Stop?
- Summary

Copyright © 2015 DigiPen Corporation

Introduction

- Let's assume that
  - we have someone who wants to get from starting point in square A to target point in B
  - obstacles (forest, wall, river) separate A and B

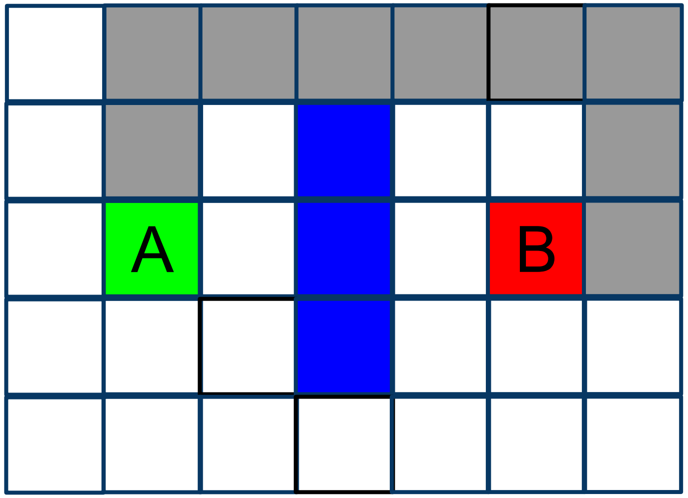
- This is illustrated here, with green being the starting point A, and red being the ending point B, and the blue squares being the obstacle in between

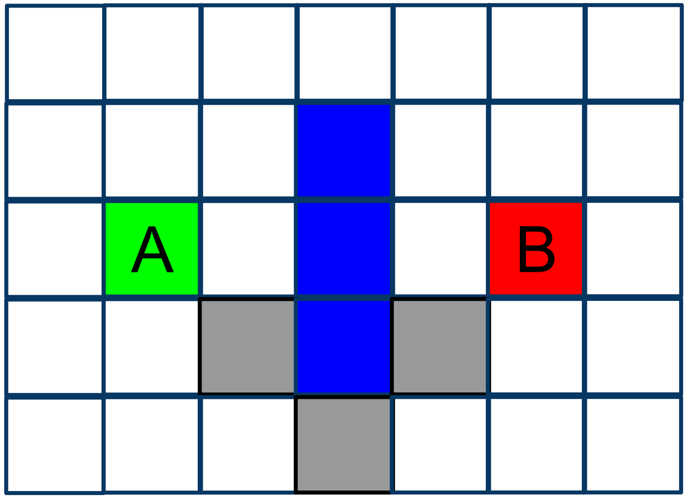


Copyright © 2015 DigiPen Corporation

Introduction (contd)

- Two types of algorithms:
    - Any path finding
        - there is no need for animated walking, ex: logic board game
    - Best path finding
        - cheapest
        - fastest
        - safest





Copyright © 2015 DigiPen Corporation

Dijkstra’s Best Path Finding Algorithm Outline

- Dijkstra’s Algorithm is a path searching algorithm that takes "cost of path" to define the best path
  - When searching all neighbors follows the best (lowest score) path
- At any instance there can be more than one best paths with the same score, in particularly, at the beginning
- If at any point the path being followed has a higher score than other encountered paths, the higher score path is abandoned and a lower score path considered instead
- Searching continues until
  - the goal is reached or
  - there is no more available moves

Copyright © 2015 DigiPen Corporation

Starting The Search

We begin the search by doing the following:

1. Begin at the starting point A and add it to an **open list** of squares to be considered

    - The **open list** contains squares that might fall along the path you want to take, but maybe not
        - It is kind of a shopping list

    - Right now there is just one item (A) on the list, but we will have more later

Copyright © 2015 DigiPen Corporation

Starting The Search (contd)

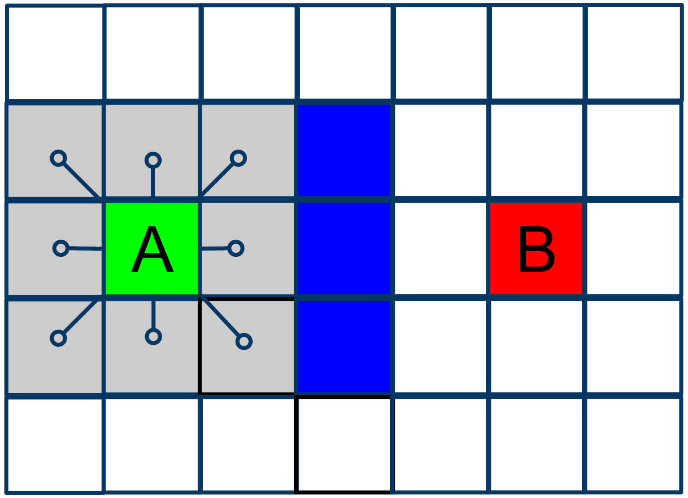
2. Look at all the reachable or walkable squares adjacent to the starting point, ignoring obstacles
   - Add them to the open list, too
   - For each of these squares, save point A as its **parent** square
     - Knowing the parent square is important when we will trace the best path at the end of algorithm

3. Drop the starting square A from the open list, and add it to the **closed list** of squares that you don't need to look at again during the path searching

Copyright © 2015 DigiPen Corporation

Starting The Search (contd)

- At this point, you should have something like on the following illustration
- In this illustration, green color indicate that the square has been added to the closed list
- All of the adjacent squares are now on the open list of squares to be checked, and they are colored in gray
- Each square in open list has a pointer that points back to its parent A



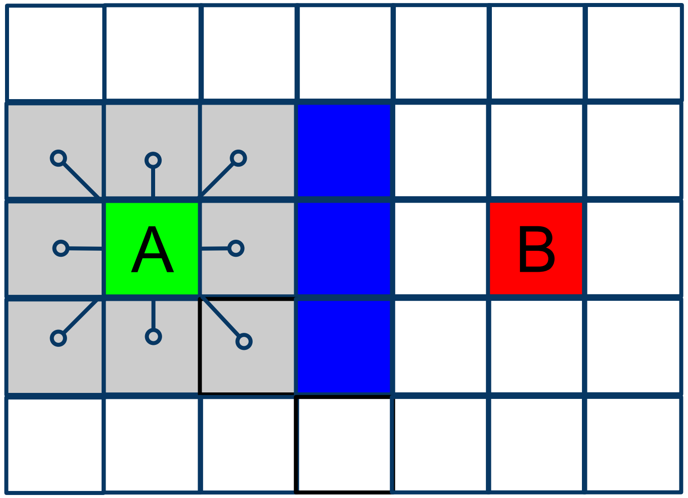
Copyright © 2015 DigiPen Corporation

Path Scoring

- What next?
  - We choose one of the adjacent squares on the open list and repeat the process

- But which square do we choose?
  - The one with the lowest G cost

- What the G?



Copyright © 2015 DigiPen Corporation

Path Scoring (contd)

- G - the cost to move from A to a given square, following the path to get there
  - can be calculated unambiguously traversing all parents squares back

Copyright © 2015 DigiPen Corporation

How To Calculate G?

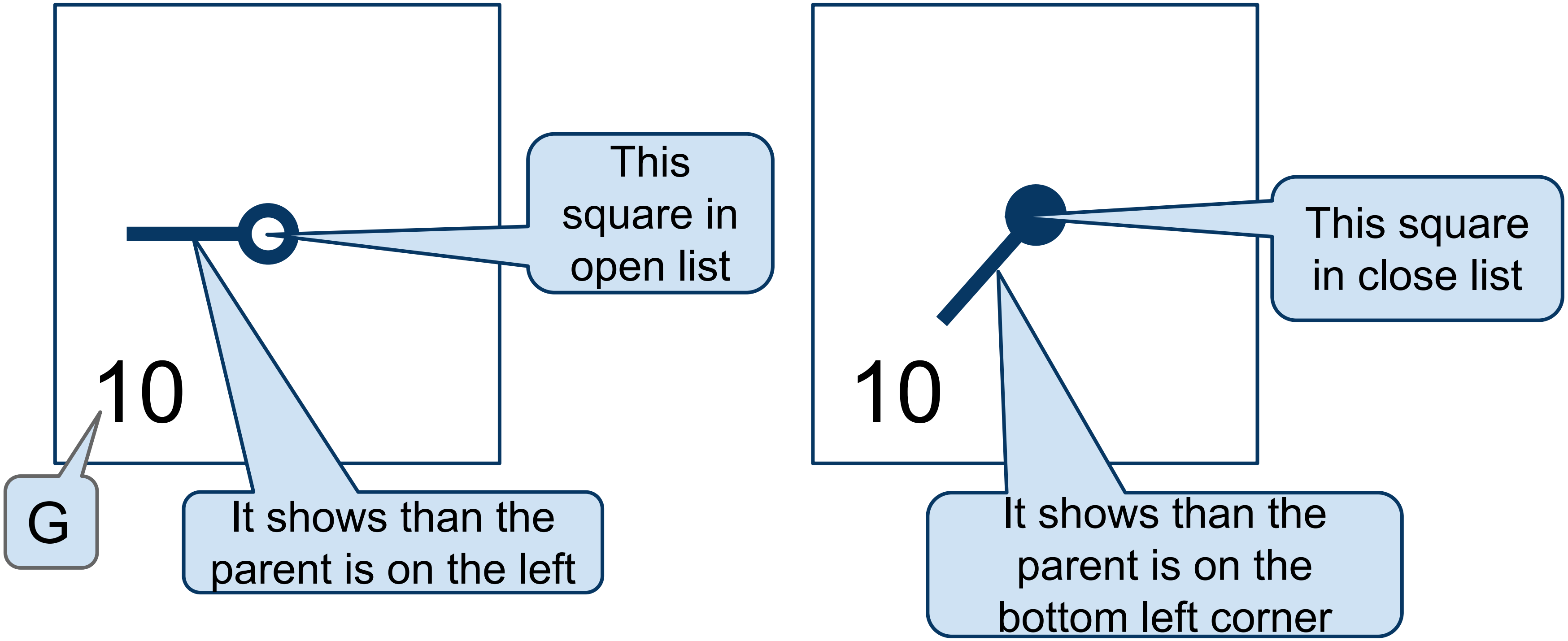
- We have to assign a cost to each horizontal, vertical, and diagonal square move
  - For example, 10 points for horizontal or vertical moves, and 14 for a diagonal move

- Since we are calculating the G cost along a specific path to a given square, the way to figure out the G cost of that square is to take the G cost of its parent, and then add 10 or 14 depending on whether it is orthogonal (non-diagonal) or diagonal from that parent square

Copyright © 2015 DigiPen Corporation

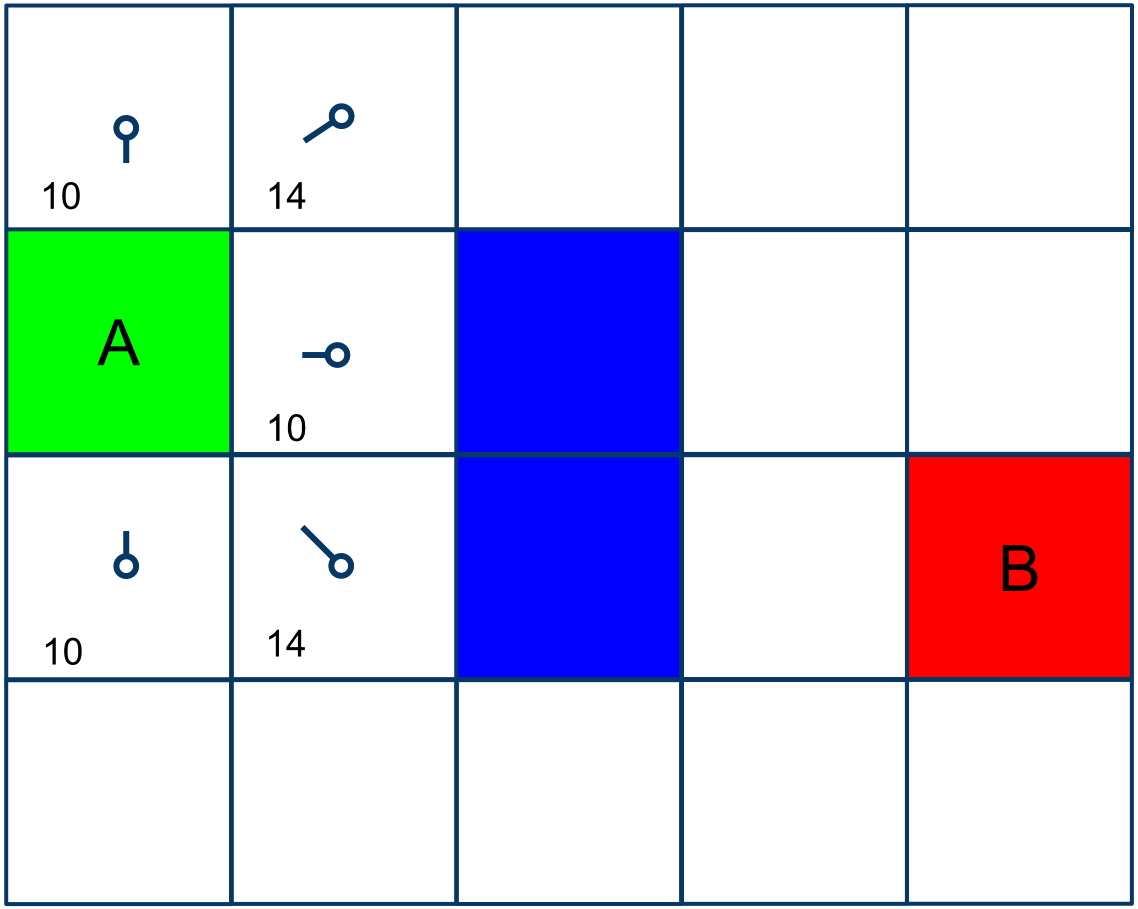
Notations

- For convenience, we will use the following notations to keep track of all costs during the path searching:


This square in open list
10
G
It shows than the parent is on the left
This square in close list
10
It shows than the parent is on the bottom left corner

Copyright © 2015 DigiPen Corporation

Starting Example



Copyright © 2015 DigiPen Corporation

Searching Loop

- To continue the search, we simply choose the lowest G score square from all those that are on the open list
- We then do the following with the selected square:
    - Drop it from the open list and add it to the closed list
    - Check all of the adjacent squares ignoring those that are on the closed list or obstacle
      (See next slide for more details for this step)

Copyright © 2015 DigiPen Corporation

How To Check An Adjacent Square?

- Add the square to the open list if they are not on the open list already and make the selected square the "parent" of the new squares
- If an adjacent square is already on the open list, check to see if this path to that square is a better one
  - In other words, check to see if the G score for that square is lower if we use the current square to get there. If not, don't do anything
  - On the other hand, if the G cost of the new path is lower, change the parent of the adjacent square to the selected. Finally, recalculate both the F and G scores of that square

Copyright © 2015 DigiPen Corporation

When To Stop?

- The target square is added to the closed list
  - In this case, there is the "best" path
  - To get it go backwards from the target square to parent square of the parent square and so on until you reach the starting square

or

- The open list is empty
  - In this case, there is no path
  - Return "No path found" flag

Copyright © 2015 DigiPen Corporation

Summary

1. Add the starting square to the open list. Calculate its G as 0.

2. Repeat the following:

2.1. Look for the lowest G cost square on the open list. We refer to this as the current square

2.2. Move the current square to the closed list

2.3. For each of the 8 squares adjacent to this current square (clockwise) …

- If it is an obstacle or on the closed list, ignore it
- If it isn't on the open list, add it to the open list. Make the current square the parent of this square. Calculate and record the G cost of the square
- If it is on the open list already, check to see if this path to that square is better, using G cost as the measure. A lower G cost means that this is a better path. If so, change the parent of the square to the current square, and recalculate the G scores of the square

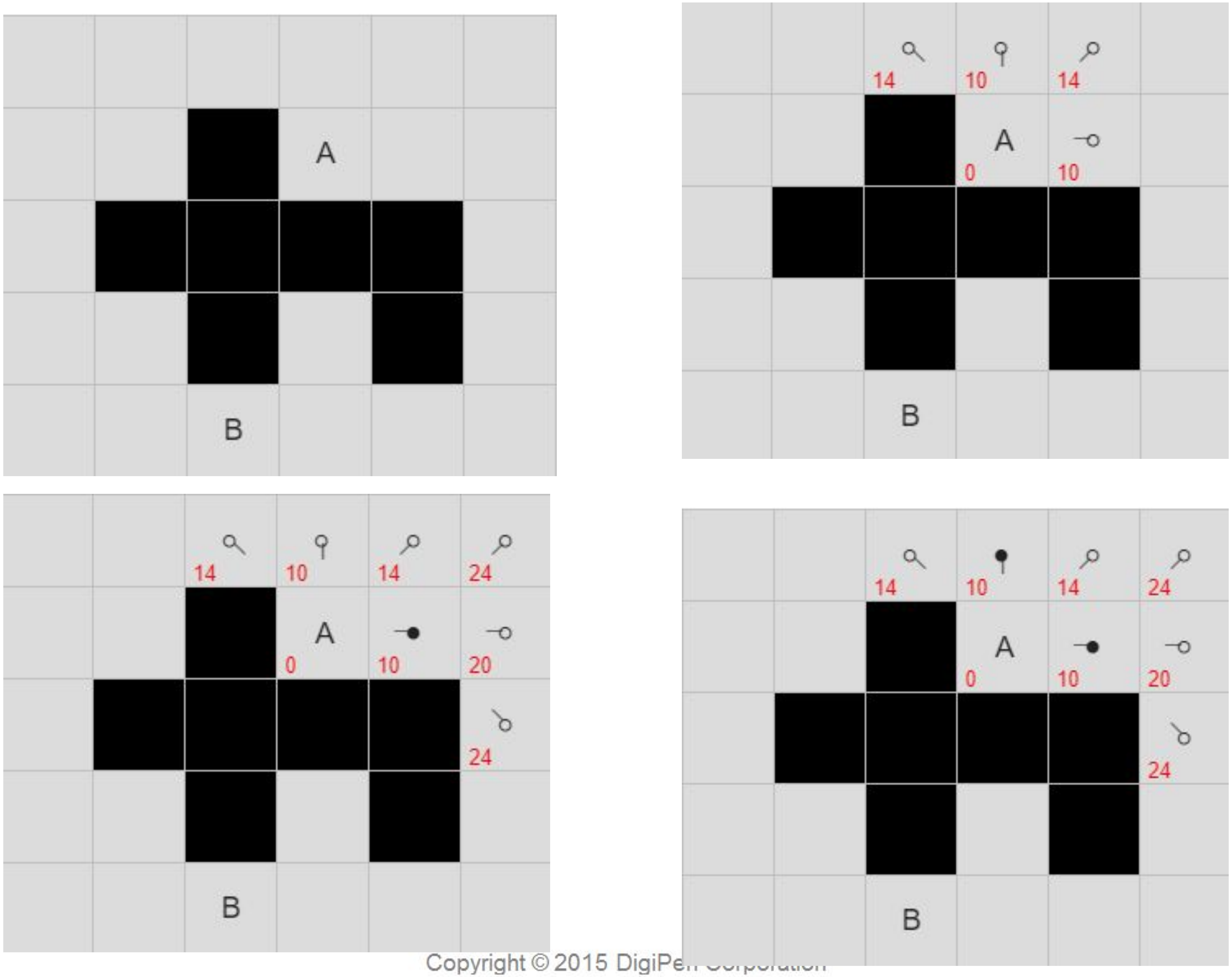
2.4. Stop when you:

- add the target square to the closed list, in which case the path has been found, or
- fail to find the target square, and the open list is empty. In this case, there is no path

3. Save the path. Working backwards from the target square, go from each square to its parent square until you reach the starting square.

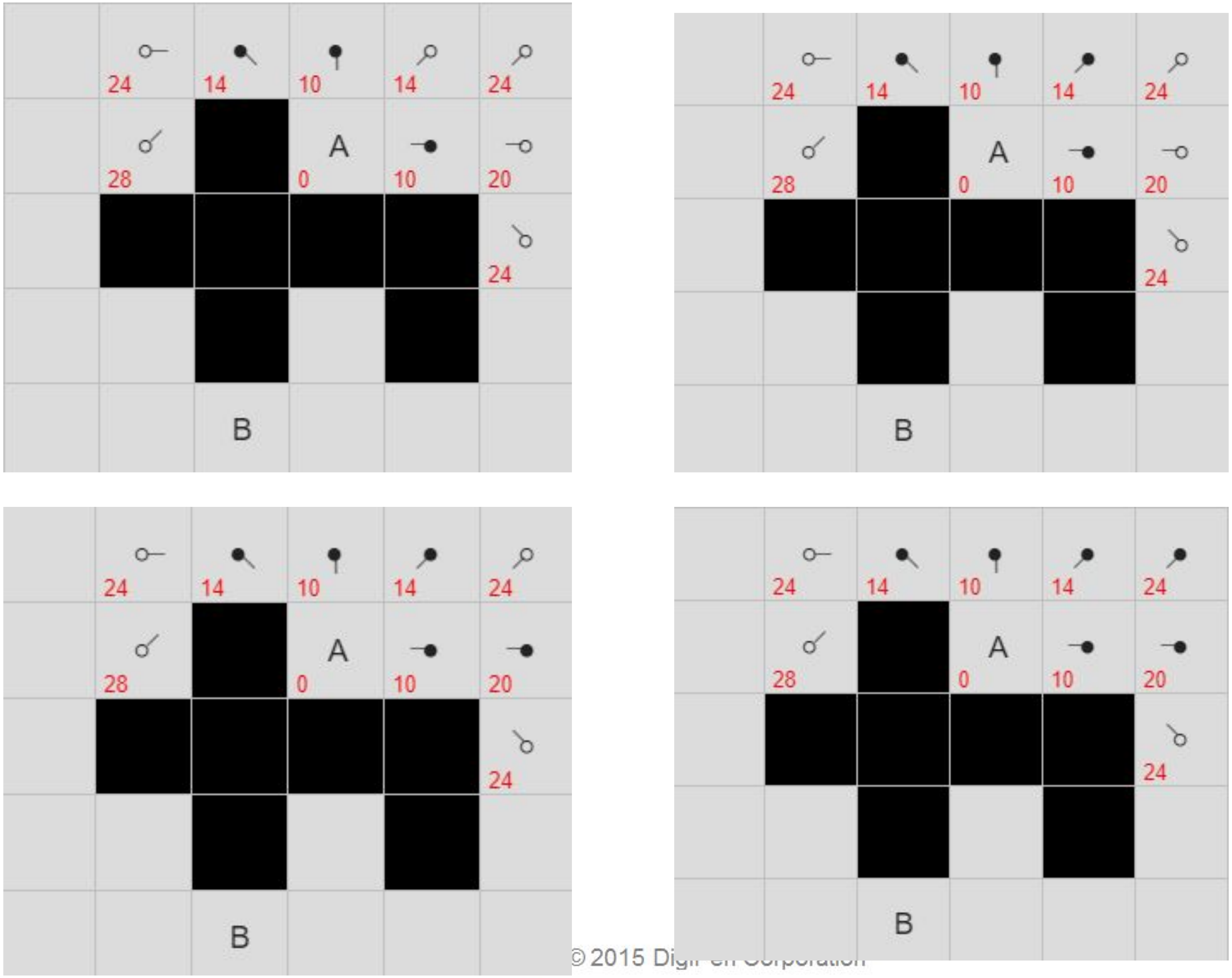
Copyright © 2015 DigiPen Corporation

A* Algorithm Example

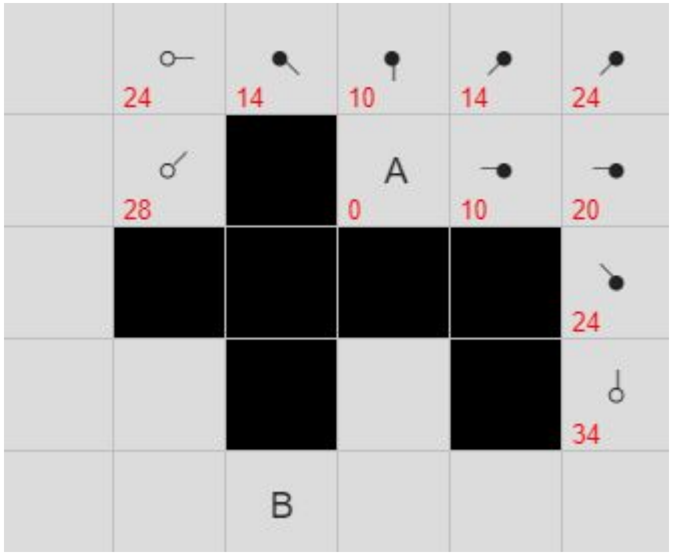


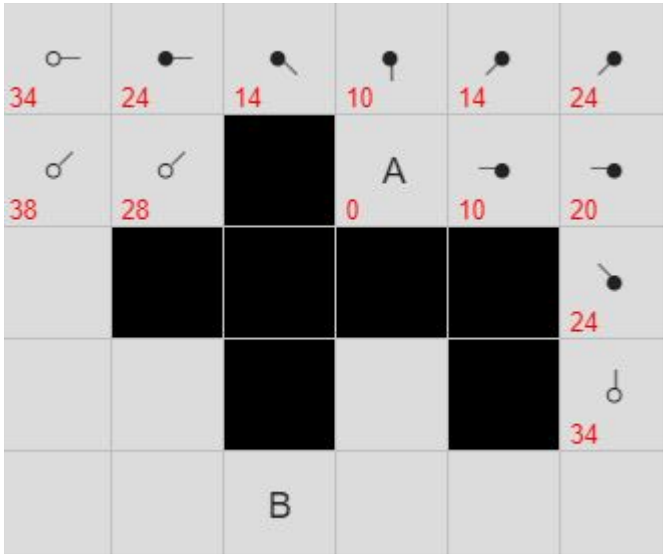
Copyright © 2015 DigiPen Corporation

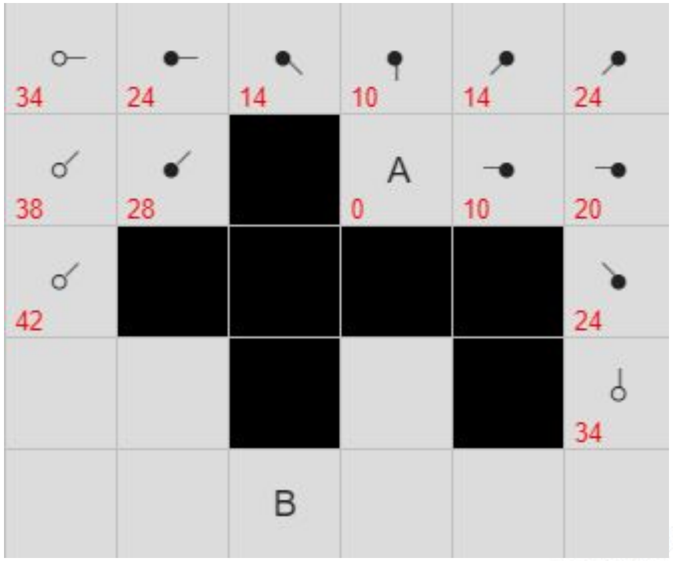
A* Algorithm Example (cont.)

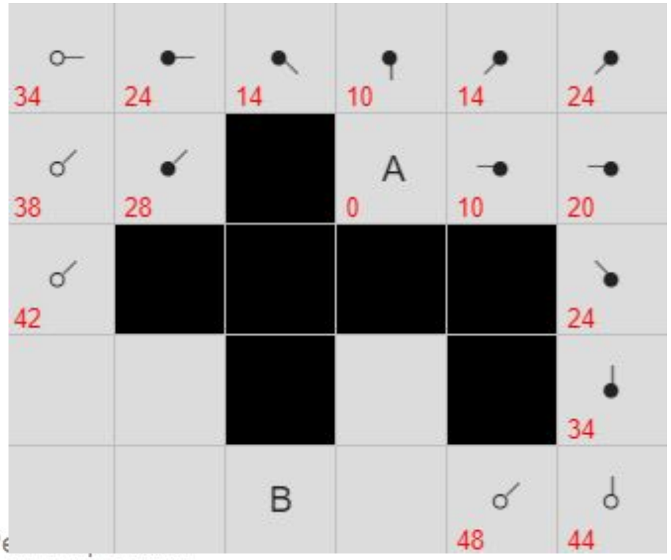


A* Algorithm Example (cont.)



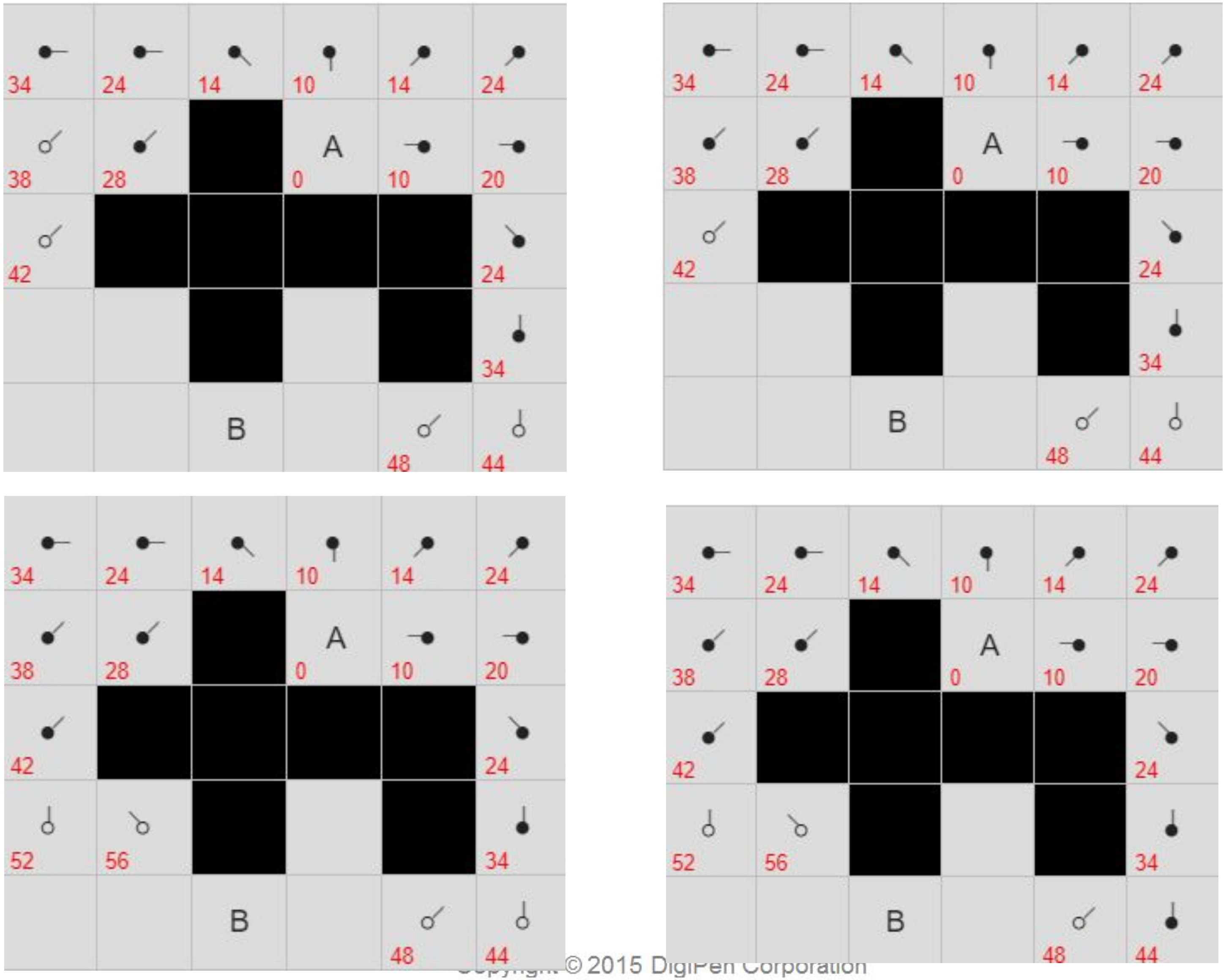






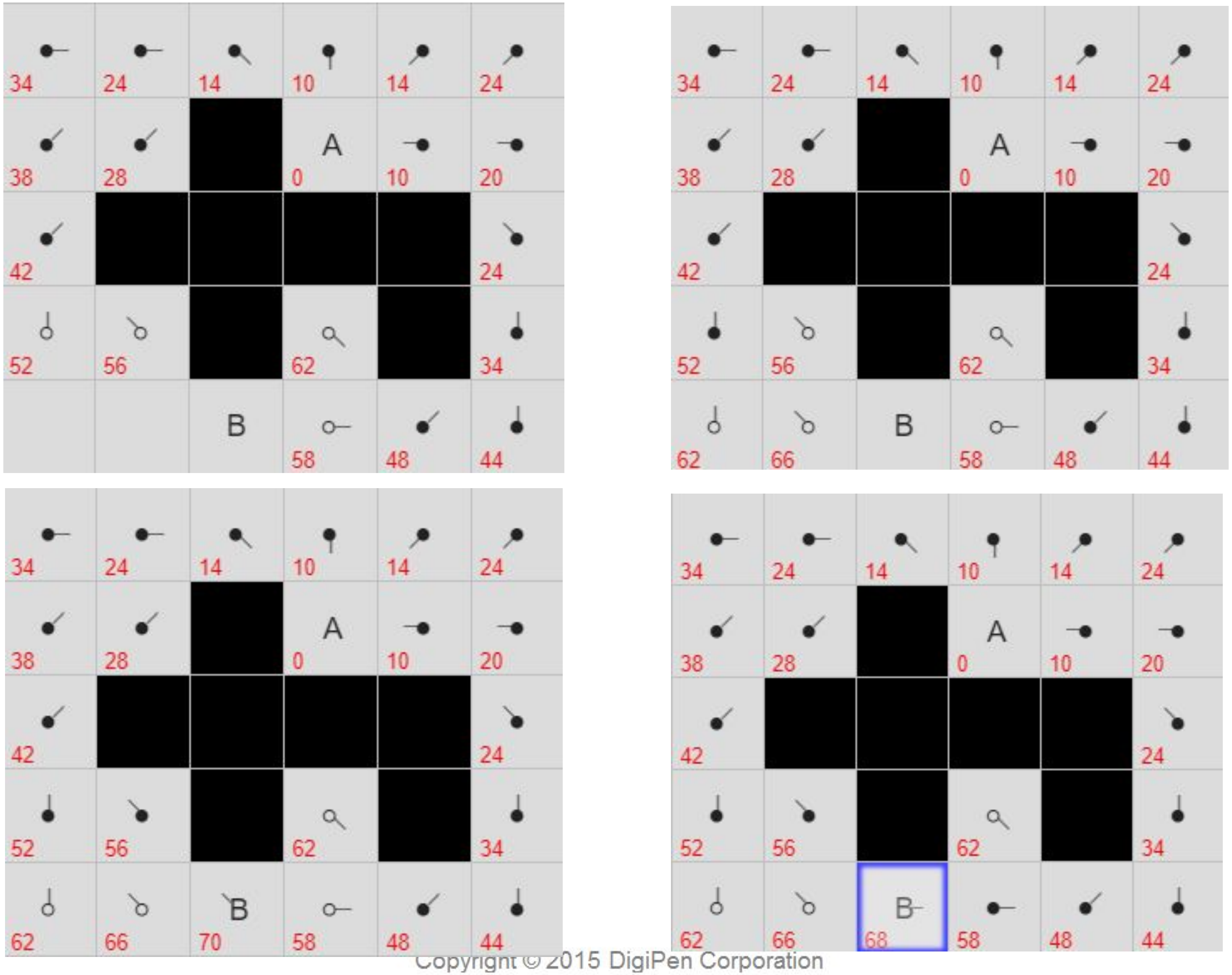
© 2015 DigiPe

A* Algorithm Example (cont.)



Copyright © 2015 DigiPen Corporation

A* Algorithm Example (cont.)



34	24	14	10	14	24
38	28	■	A 0	10	20
42	■	■	■	■	24
52	56	■	62	■	34
		B	58	48	44

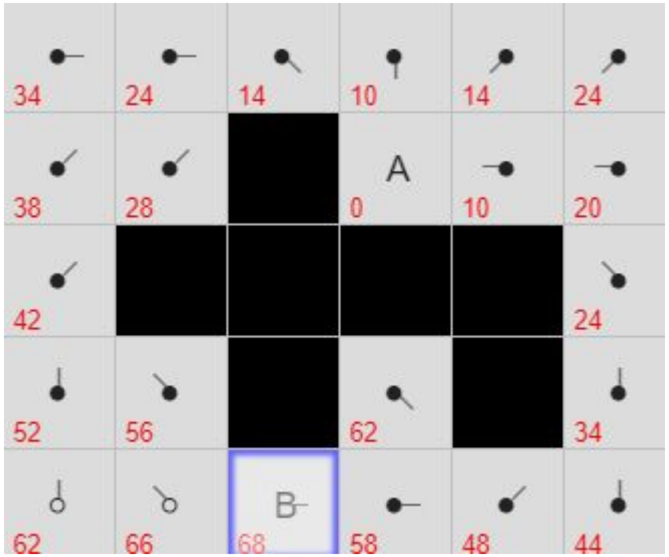
34	24	14	10	14	24
38	28	■	A 0	10	20
42	■	■	■	■	24
52	56	■	62	■	34
62	66	B	58	48	44

34	24	14	10	14	24
38	28	■	A 0	10	20
42	■	■	■	■	24
52	56	■	62	■	34
62	66	B 70	58	48	44

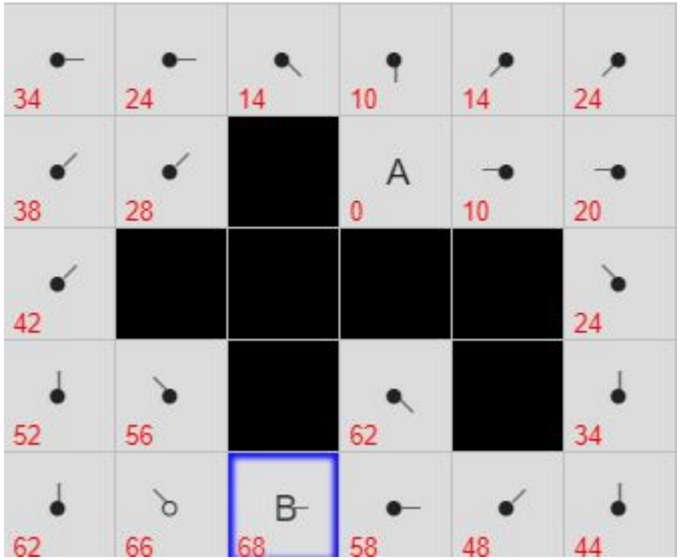
34	24	14	10	14	24
38	28	■	A 0	10	20
42	■	■	■	■	24
52	56	■	62	■	34
62	66	B 68	58	48	44

Copyright © 2015 DigiPen Corporation

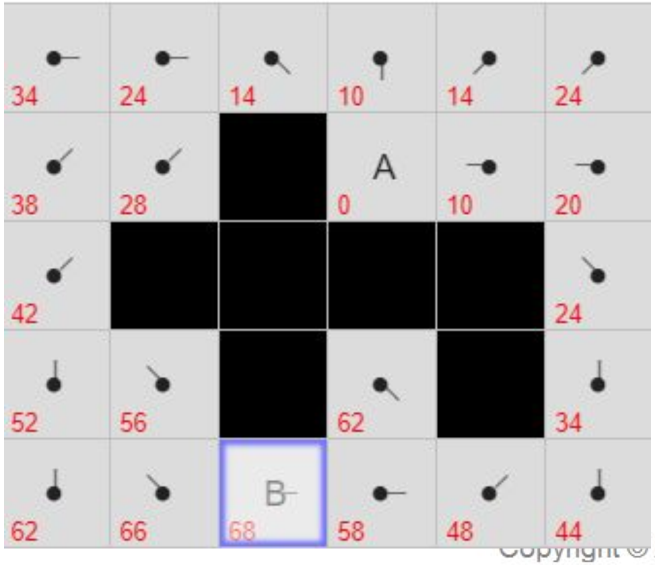
A* Algorithm Example (cont.)



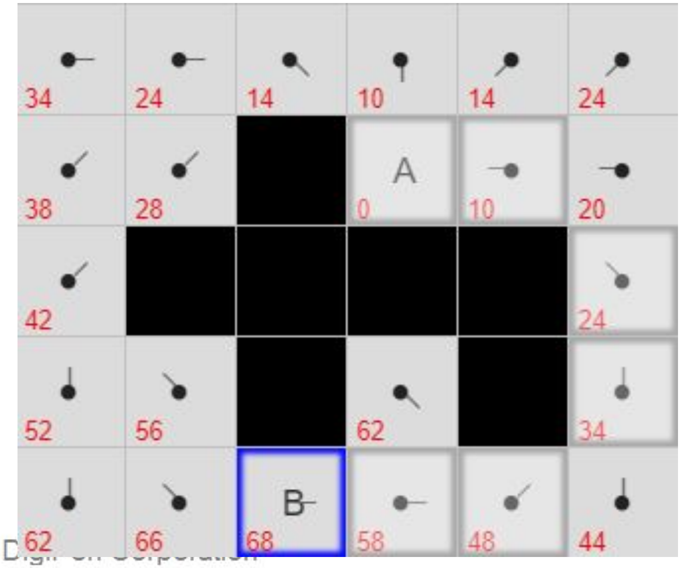
34	24	14	10	14	24
38	28		A 0	10	20
42					24
52	56		62		34
62	66	B 68	58	48	44



34	24	14	10	14	24
38	28		A 0	10	20
42					24
52	56		62		34
62	66	B 68	58	48	44



34	24	14	10	14	24
38	28		A 0	10	20
42					24
52	56		62		34
62	66	B 68	58	48	44



34	24	14	10	14	24
38	28		A 0	10	20
42					24
52	56		62		34
62	66	B 68	58	48	44

Copyright © 2015 DigiPen Corporation