Digital Image Processing
(CPE-415)

Assignment #1 – Detailed Solutions

Topics: Image Sensors, Sampling & Quantization, Pixel Connectivity

Reference: *Digital Image Processing*, 3rd Edition
by Gonzalez & Woods

Problems: 2.1, 2.2, 2.5, 2.10, 2.11, 2.12, 2.13, 2.15, 2.16, 2.17

Name: _____

Roll No: _____

Section: _____

Assignment Date: 20th February 2026

Submission Due Date: 27th February 2026

Total Marks: 10

Contents

2.1 – Smallest Dot the Eye Can Discern

Problem Statement

Using the background information provided in Section 2.1, and thinking purely in geometric terms, estimate the diameter of the smallest printed dot that the eye can discern if the page on which the dot is printed is 0.2 m away from the eyes. Assume for simplicity that the visual system ceases to detect the dot when the image of the dot on the fovea becomes smaller than the diameter of one receptor (cone) in that area of the retina. Assume further that the fovea can be modeled as a square array of dimensions 1.5 mm × 1.5 mm, and that the cones and spaces between the cones are distributed uniformly throughout this array.

Step 1: Understand What We Know (Given Information)

From the textbook (Section 2.1), we are given the following facts about the human eye:

- The fovea is the central part of the retina where cones are most dense. This is where sharp, detailed vision happens.
- The fovea is modeled as a square array of size 1.5 mm × 1.5 mm.
- The density of cones in the fovea is approximately 150,000 elements per mm^2.
- The distance from the center of the lens to the retina (along the visual axis) is approximately 17 mm.
- The page is at a distance of $d = 0.2$ m $= 200$ mm from the eye.

Step 2: Find the Total Number of Cones in the Fovea

The fovea has area:

$$A_{\text{fovea}} = 1.5 \times 1.5 = 2.25 \text{ mm}^2$$

Total number of cones:

$$N = 150{,}000 \times 2.25 = 337{,}500 \text{ cones}$$

Step 3: Find the Size of One Cone (One Receptor)

Since the fovea is modeled as a square array and the cones are uniformly distributed, the number of cones along one side of the fovea is:

$$n = \sqrt{337{,}500} \approx 581 \text{ cones per side}$$

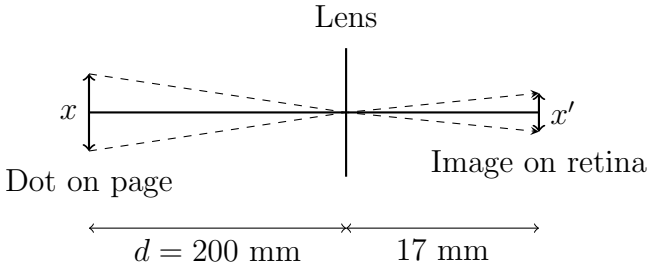
So the diameter (size) of one cone is:

$$d_{\text{cone}} = \frac{1.5 \text{ mm}}{581} \approx 0.00258 \text{ mm} = 2.58 \ \mu\text{m}$$

What this means: Each cone receptor on the fovea occupies roughly a 2.58 μm × 2.58 μm square.

Step 4: Use Geometry to Find the Smallest Dot

Now we use similar triangles. The eye works like a pinhole camera — an object in the real world creates an image on the retina. The geometry looks like this:



By similar triangles:

$$\frac{x}{d} = \frac{x'}{17}$$

where:

- x = diameter of the dot on the page (what we want to find)
- $d = 200$ mm (distance from page to eye)
- x' = size of the image on the retina = d_{cone}
- 17 mm = distance from lens to retina

We want the image of the dot to be exactly the size of one cone (any smaller and the eye can't see it). So:

$$\frac{x}{200} = \frac{0.00258}{17} \implies x = \frac{200 \times 0.00258}{17} \approx 0.0304 \text{ mm} \approx 30.4\ \mu\text{m}$$

Answer

The smallest dot the eye can discern at a distance of 0.2 m is approximately **0.0304 mm** (or about **30.4 μm**). This is roughly 1/3 of the thickness of a human hair.
In plain English: We figured out how big one "pixel" on our retina is (one cone receptor), then used simple geometry to calculate what real-world size that corresponds to at a distance of 20 cm. If the dot is smaller than this, its image on our retina would be smaller than one cone, and we simply cannot detect it.

2.2 – Dark Theater Adaptation

Problem Statement

When you enter a dark theater on a bright day, it takes an appreciable interval of time before you can see well enough to find an empty seat. Which of the visual processes explained in Section 2.1 is at play in this situation?

Step 1: Understand the Scenario

- You are outside in bright sunlight. Your eyes are adapted to high light intensity (photopic vision — cones are active).
- You walk into a dark movie theater. Suddenly, the light level drops dramatically.
- For several minutes, you can barely see anything. Then gradually, your vision improves and you can start to see seats, people, etc.

Step 2: Identify the Visual Process

The process at play is called **brightness adaptation** (also called *dark adaptation*). From Section 2.1.3 of the textbook:

- The human visual system can handle an enormous range of light intensities — about 10^{10} from the dimmest to the brightest.
- However, the eye cannot operate over this entire range simultaneously. It works within a smaller window of intensity at any given time.
- The eye adjusts this window by changing its overall sensitivity — this is brightness adaptation.
- When you go from bright to dark, the eye needs time to shift its adaptation level from a high-intensity setting to a low-intensity setting.

Step 3: What Happens Physically

1. **Outside (bright):** Your cones are active (photopic vision). Cones need lots of light to work well. The rods are essentially "bleached out" (overwhelmed by the bright light and inactive).
2. **Entering the theater (dark):** The light level drops below what cones can handle effectively. Your vision must switch from cone-based (photopic) to rod-based (scotopic) vision.
3. **The delay:** The rods need time (typically 20–30 minutes for full adaptation) to regenerate the visual pigment (rhodopsin) that was bleached by the bright light. During this regeneration period, your sensitivity to low light gradually increases.
4. **After adaptation:** The rods become fully functional, and you can see in the dim light of the theater. However, you won't see colors well because rods are not sensitive to color — everything looks grayish (scotopic vision).

Answer

The visual process at play is **brightness adaptation** (specifically, dark adaptation). When you move from a brightly lit environment to a dark one, the eye

must shift its sensitivity from photopic (cone-based, bright-light) vision to scotopic (rod-based, dim-light) vision. This transition takes time because the rods need to regenerate their visual pigments (rhodopsin) that were bleached by the bright light. This is why you are essentially "blind" for several minutes until your eyes adjust.

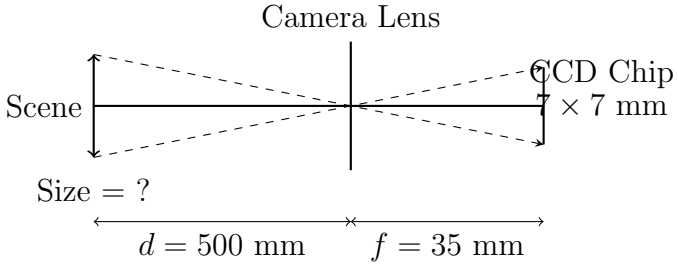
2.5 – CCD Camera Resolution (Line Pairs per mm)

Problem Statement

A CCD camera chip of dimensions 7×7 mm, and having 1024×1024 elements, is focused on a square, flat area, located 0.5 m away. How many line pairs per mm will this camera be able to resolve? The camera is equipped with a 35-mm lens. (Hint: Model the imaging process as in Fig. 2.3, with the focal length of the camera lens substituting for the focal length of the eye.)

Step 1: Understand the Setup

Think of this like the eye model from Section 2.1.2 (Fig. 2.3), but instead of an eye, we have a CCD camera:



Given:

- CCD chip size: 7×7 mm
- CCD resolution: 1024×1024 elements (pixels)
- Distance to scene: $d = 0.5$ m $= 500$ mm
- Focal length of lens: $f = 35$ mm

Step 2: Find the Size of the Scene Area Being Imaged

Using similar triangles (same geometry as the eye in Fig. 2.3):

$$\frac{\text{Scene size}}{d} = \frac{\text{Chip size}}{f} \implies \frac{\text{Scene size}}{500} = \frac{7}{35} \implies \text{Scene size} = 500 \times \frac{7}{35} = 500 \times 0.2 = 100 \text{ mm}$$

So the camera sees a 100×100 mm area of the scene.

Step 3: Find the Resolution on the Scene

The CCD has 1024 pixels spread across 100 mm of the scene. So:

$$\text{Pixels per mm (on the scene)} = \frac{1024}{100} = 10.24 \text{ pixels/mm}$$

Step 4: Convert to Line Pairs per mm

What is a "line pair"? A line pair consists of one dark line and one bright line (one white + one black stripe). To resolve one line pair, you need at least 2 pixels — one pixel for

the dark line and one for the bright line. This is essentially the Nyquist criterion: to resolve a spatial frequency, you need at least 2 samples per cycle.

$$\text{Line pairs per mm} = \frac{\text{Pixels per mm}}{2} = \frac{10.24}{2} = 5.12 \text{ lp/mm}$$

Answer

The camera can resolve approximately **5.12 line pairs/mm** on the scene.
In plain English: The camera sees a 100×100 mm area with 1024 pixels across. That gives us about 10 pixels per mm. Since you need 2 pixels to see one "stripe" (one dark + one light = one line pair), we can resolve about 5 line pairs per mm.

2.10 – HDTV Movie Storage

Problem Statement

High-definition television (HDTV) generates images with 1125 horizontal TV lines interlaced (where every other line is painted on the tube face in each of two fields, each field being 1/60th of a second in duration). The width-to-height aspect ratio of the images is 16:9. A company has designed an image capture system that generates digital images from HDTV images. The resolution of each TV (horizontal) line in their system is in proportion to vertical resolution, with the proportion being the width-to-height ratio of the images. Each pixel in the color image has 24 bits of intensity resolution, 8 bits each for a red, a green, and a blue image. How many bits would it take to store a 2-hour HDTV movie?

Step 1: Find the Image Dimensions (in Pixels)

Vertical resolution:
There are 1125 horizontal lines, so the vertical resolution is 1125 pixels (each line = one row of pixels).

Horizontal resolution:
The problem says: horizontal resolution is proportional to vertical resolution, with the proportion being the aspect ratio (16:9).

$$\text{Horizontal pixels} = 1125 \times \frac{16}{9} = 1125 \times 1.7778 = 2000 \text{ pixels}$$

So each frame is 2000×1125 pixels.

Step 2: Find the Number of Bits per Frame

Each pixel has 24 bits (8 bits for Red + 8 bits for Green + 8 bits for Blue).

$$\text{Bits per frame} = 2000 \times 1125 \times 24 = 54{,}000{,}000 \text{ bits/frame} = 5.4 \times 10^7 \text{ bits/frame}$$

Step 3: Find the Frame Rate

The system uses interlaced scanning:

- Each "field" paints every other line and takes 1/60th second.
- It takes 2 fields to make one complete frame.
- So: frame rate = 60/2 = 30 frames per second.

Step 4: Find the Total Duration in Seconds

$$\text{Duration} = 2 \text{ hours} = 2 \times 60 \times 60 = 7{,}200 \text{ seconds}$$

Step 5: Find Total Number of Frames

$$\text{Total frames} = 30 \times 7{,}200 = 216{,}000 \text{ frames}$$

Step 6: Calculate Total Bits

Total bits = bits/frame×total frames = 54,000,000×216,000 = 11,664,000,000,000 bits ≈ 1.1664×10^{13}

To put this in perspective:

$$\frac{1.1664 \times 10^{13}}{8} = 1.458 \times 10^{12} \text{ bytes} \approx 1.458 \text{ terabytes (TB)}$$

Answer

It would take approximately $\mathbf{1.1664 \times 10^{13}}$ **bits** (about **1.458 TB**) to store a 2-hour HDTV movie.

Breakdown:

- Frame size: 2000×1125 pixels
- Bits per pixel: 24 (color)
- Frame rate: 30 fps (interlaced: 2 fields per frame)
- Duration: 7200 seconds
- Total: $2000 \times 1125 \times 24 \times 30 \times 7200 = 1.1664 \times 10^{13}$ bits

This is why video compression (like H.264 or H.265) is absolutely necessary — without it, even a modern hard drive would struggle to store a single movie!

2.11 – Adjacency of Two Image Subsets

Problem Statement

Consider the two image subsets, S_1 and S_2, shown in the following 5×10 figure. For $V = \{1\}$, determine whether these two subsets are (a) 4-adjacent, (b) 8-adjacent, or (c) m-adjacent.

S_1 = pixels in rows 0–3, columns 1–4 (left dashed box).
S_2 = pixels in rows 0–3, columns 5–8 (right dashed box).

Step 1: Identify the Two Subsets

Using (row, column) coordinates starting from $(0, 0)$ at top-left, and reading the 5×10 grid from the textbook figure:

S_1 pixels with value 1 (i.e., value $\in V$): $(1, 3),\ (2, 3),\ (3, 2),\ (3, 3),\ (3, 4)$.
S_2 pixels with value 1 (i.e., value $\in V$): $(0, 7),\ (0, 8),\ (1, 6),\ (2, 5),\ (2, 6)$.

Step 2: Find the "Bridge" Pixels

The key question is: are any pixels from S_1 adjacent to any pixels from S_2? Looking at the pixels closest to the boundary between S_1 and S_2:

- Pixel $p = (3, 4) \in S_1$ (value = 1)
- Pixel $q = (2, 5) \in S_2$ (value = 1)

This is the closest V-valued pair across the boundary.

Step 3: Check 4-Adjacency

Two pixels are 4-adjacent if one is in the N_4 (4-neighborhood) of the other. The 4-neighbors are the pixels directly above, below, left, and right.

Check $(3, 4)$ and $(2, 5)$:

$$D_4\big((3, 4),\ (2, 5)\big) = |3 - 2| + |4 - 5| = 1 + 1 = 2 \neq 1$$

They are diagonal neighbors, **not** 4-neighbors. No other S_1–S_2 pair has $D_4 = 1$.

(a) 4-Adjacency

NO. S_1 and S_2 are not 4-adjacent. No pixel from S_1 is a direct horizontal or vertical neighbor of any pixel from S_2 (with both having values in V). The closest pair $(3, 4)$ and $(2, 5)$ are diagonal, not 4-adjacent.

Step 4: Check 8-Adjacency

Two pixels are 8-adjacent if one is in the N_8 (8-neighborhood) of the other. The 8-neighbors include all surrounding pixels (horizontal, vertical, *and* diagonal).

Check $(3, 4)$ and $(2, 5)$:

$$D_8\big((3, 4),\ (2, 5)\big) = \max(|3 - 2|,\ |4 - 5|) = \max(1, 1) = 1 \quad \checkmark$$

They are diagonal neighbors $\Rightarrow$ 8-adjacent. Both have value $1 \in V$.

(b) 8-Adjacency

YES. S_1 and S_2 are 8-adjacent via the diagonal pair $(3,4) \leftrightarrow (2,5)$. Both pixels have value $1 \in V$, and $D_8 = 1$.

Step 5: Check m-Adjacency

Two pixels p and q with values from V are m-adjacent if:

1. $q \in N_4(p)$, OR
2. $q \in N_D(p)$ (diagonal neighbor) AND $N_4(p) \cap N_4(q)$ has no pixels with values from V.

$(3,4)$ and $(2,5)$ are diagonal, so condition (i) fails. Check condition (ii):

$$N_4(3,4) \cap N_4(2,5) = \{(2,4),\ (3,5)\}$$

- grid[2][4] $= 0 \notin V$ ✓
- grid[3][5] $= 0 \notin V$ ✓

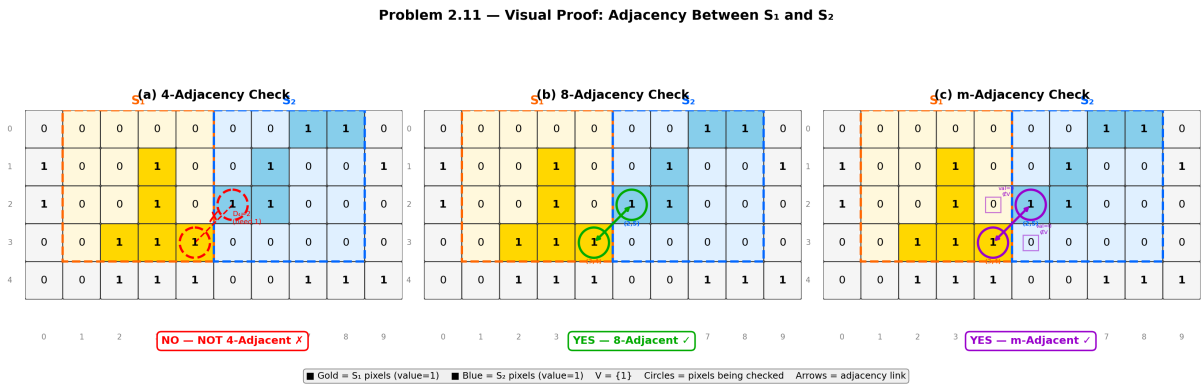
Neither shared 4-neighbor has a value in V, so the diagonal connection is allowed.

(c) m-Adjacency

YES. S_1 and S_2 are m-adjacent via the diagonal pair $(3,4) \leftrightarrow (2,5)$. The diagonal is permitted because neither shared 4-neighbor ($(2,4) = 0$ and $(3,5) = 0$) has a value in $V = \{1\}$.

Summary: S_1 and S_2 are (a) NOT 4-adjacent, (b) 8-adjacent, and (c) m-adjacent.

Visual Proof



2.12 – Converting 8-Path to 4-Path

Problem Statement

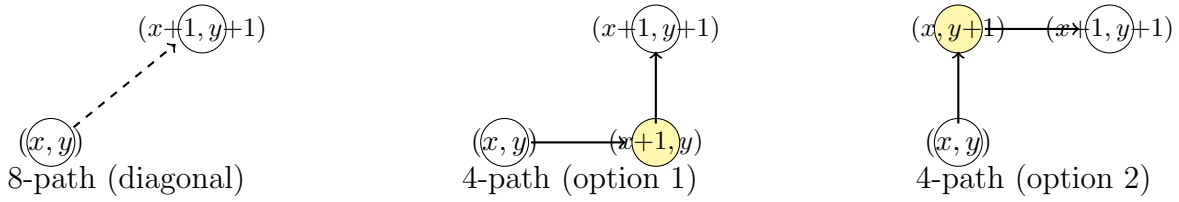
Develop an algorithm for converting a one-pixel-thick 8-path to a 4-path.

Step 1: Understand the Problem

- **What is an 8-path?** A sequence of pixels where each consecutive pair is 8-adjacent (they can be connected horizontally, vertically, or diagonally).
- **What is a 4-path?** A sequence of pixels where each consecutive pair is 4-adjacent (they can ONLY be connected horizontally or vertically — NO diagonals allowed).
- **The Goal:** Take any 8-path and convert it to a 4-path that goes between the same start and end points.

Step 2: The Key Idea

Whenever we encounter a diagonal step in the 8-path, we need to replace it with two steps: one horizontal and one vertical. A diagonal move goes from (x, y) to $(x+1, y+1)$ — we can break this into: first go to $(x+1, y)$ then to $(x+1, y+1)$, OR first go to $(x, y+1)$ then to $(x+1, y+1)$.



Step 3: The Algorithm

Algorithm: Convert 8-Path to 4-Path

Input: An 8-path $P_8 = \{p_0, p_1, p_2, \ldots, p_n\}$ where $p_i = (x_i, y_i)$
Output: A 4-path P_4

1. Initialize $P_4 = \{p_0\}$ (start with the first pixel).
2. For each consecutive pair (p_i, p_{i+1}) in the 8-path:
   (a) Compute: $\Delta x = x_{i+1} - x_i$ and $\Delta y = y_{i+1} - y_i$
   (b) If $|\Delta x| + |\Delta y| = 1$ (already a 4-connected step, i.e., purely horizontal or purely vertical):
   Add p_{i+1} to P_4.
   (c) Else if $|\Delta x| = 1$ AND $|\Delta y| = 1$ (diagonal step):
   Insert an intermediate pixel: add $(x_i + \Delta x,\ y_i)$ to P_4.
   Then add $p_{i+1} = (x_{i+1}, y_{i+1})$ to P_4.
3. Return P_4.

Step 4: Example

Consider the 8-path: $(0, 0) \to (1, 1) \to (2, 1) \to (3, 2)$

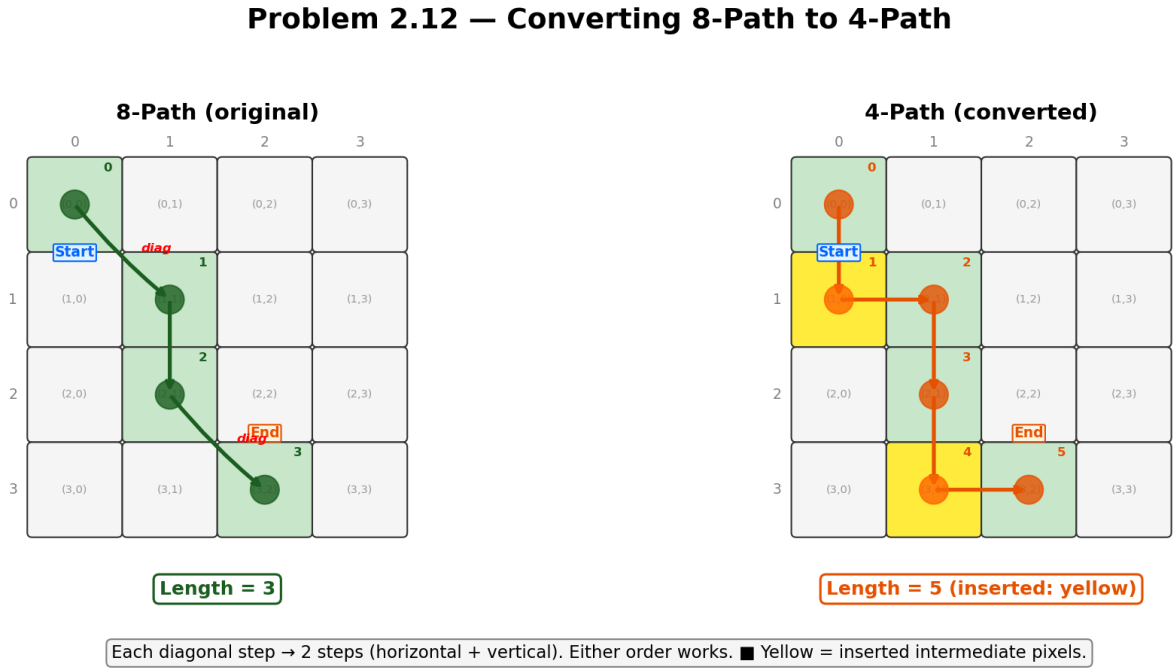
- $(0,0) \to (1,1)$: Diagonal! Insert $(1,0)$. Path becomes: $(0,0) \to (1,0) \to (1,1)$
- $(1,1) \to (2,1)$: Vertical move. Already 4-connected. Just add. Path: $\ldots \to (2,1)$
- $(2,1) \to (3,2)$: Diagonal! Insert $(3,1)$. Path: $\ldots \to (3,1) \to (3,2)$

Final 4-path: $(0,0) \to (1,0) \to (1,1) \to (2,1) \to (3,1) \to (3,2)$

Answer

The algorithm scans the 8-path step by step. For every diagonal step (where both x and y change by 1), it inserts an additional intermediate pixel that decomposes the diagonal into two 4-connected moves (one horizontal + one vertical). Non-diagonal steps are kept as-is. The choice of whether to go horizontal-first or vertical-first is arbitrary (either produces a valid 4-path).

Visual Proof



2.13 – Converting m-Path to 4-Path

Problem Statement

Develop an algorithm for converting a one-pixel-thick m-path to a 4-path.

Step 1: The Critical Insight — What m-adjacency Guarantees

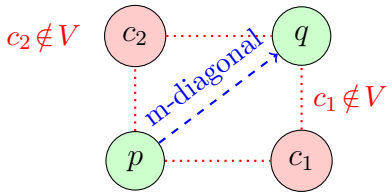
Recall the m-adjacency rule: a diagonal step from $p = (x, y)$ to $q = (x{+}1, y{+}1)$ (both $\in V$) is only permitted when both shared 4-neighbors are **not** in V:

$$(x{+}1,\, y) \notin V \quad \text{AND} \quad (x,\, y{+}1) \notin V$$

This is exactly the opposite of what Problem 2.12 faces. In an 8-path the diagonal might skip over a perfectly valid V-pixel. In an m-path the diagonal is taken *because* no valid V-pixel exists in between.

Step 2: Why This Changes the Conversion

When we try to break the diagonal step $p \to q$ into two 4-connected steps, we must insert an intermediate pixel — either $(x{+}1, y)$ or $(x, y{+}1)$.



Both candidates c_1 and c_2 are **guaranteed to be** $\notin V$ by the m-adjacency definition. Therefore:

- A **strict** V-constrained 4-path *cannot* pass through this diagonal step using only intermediate pixels in V — it is **impossible for that local step**.
- A **geometric** (relaxed) 4-path can still be produced, but it passes through a pixel $\notin V$.

This is the fundamental difference from Problem 2.12, where the intermediate pixel *may* be in V.

Step 3: The Algorithm (Two Interpretations)

Algorithm: Convert m-Path to 4-Path

Input: An m-path $P_m = \{p_0, p_1, \ldots, p_n\}$ where $p_i = (x_i, y_i)$
Output: A 4-path P_4 (geometric, may visit non-V pixels)

1. Initialize $P_4 = \{p_0\}$.
2. For each consecutive pair (p_i, p_{i+1}):
   (a) Compute $\Delta x = x_{i+1} - x_i, \quad \Delta y = y_{i+1} - y_i$.
   (b) If $|\Delta x| + |\Delta y| = 1$ **(4-connected step):**
   Append p_{i+1} to P_4. *Already valid; no change needed.*
   (c) Else ($|\Delta x| = |\Delta y| = 1$, **diagonal step):**

Both $(x_i+\Delta x,\, y_i)$ and $(x_i,\, y_i+\Delta y)$ are $\notin V$.
Append intermediate pixel $(x_i+\Delta x,\, y_i)$ to P_4.
Append p_{i+1} to P_4.
Note: the appended intermediate pixel is $\notin V$.

3. Return P_4.

Alternative (strict V-path): If a path that stays entirely in V is required, a diagonal step in the m-path signals that *no local 4-connected bridge through V exists*. You must search for a completely different longer route from p_0 to p_n using BFS/DFS over only V-valued pixels. Such a route may or may not exist.

Step 4: Comparison with Problem 2.12

	8-path → 4-path	**m-path → 4-path**
Geometric algorithm	Insert H+V	Insert H+V (same)
Intermediate pixel $\in V$?	**Possibly yes**	**Guaranteed NO**
Strict V-path always obtainable?	**Yes**	**Not always**
Diagonal reason	Any 8-neighbor in V	Only when corners $\notin V$

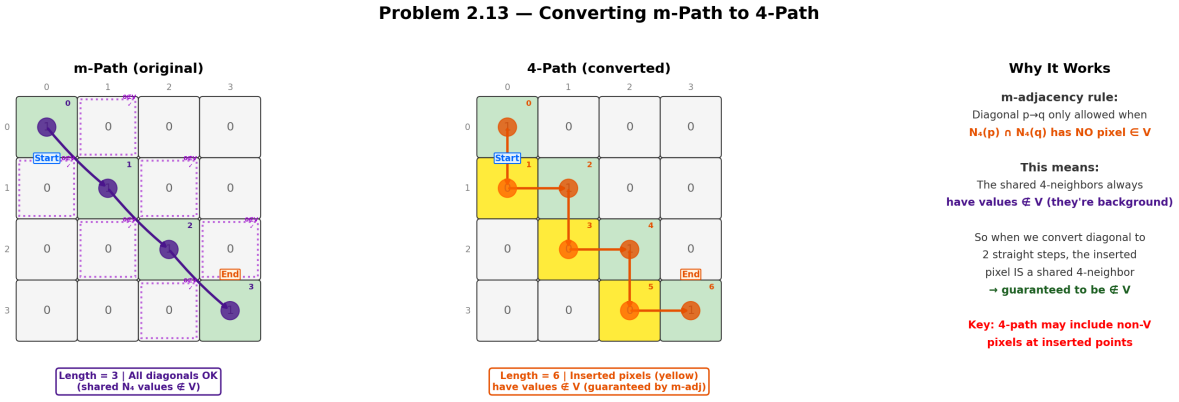
Answer

The geometric algorithm is the same as Problem 2.12 — replace each diagonal step with one horizontal + one vertical step by inserting an intermediate pixel. However, the result is fundamentally different in quality:

- In a **8-path**, the intermediate pixel *might* be in V, so the converted 4-path can remain entirely within V.
- In an **m-path**, the m-adjacency condition *guarantees* both candidate intermediate pixels are $\notin V$ (that is why the diagonal was used in the first place). Therefore the strict V-constrained 4-path **does not exist** for that diagonal step.

The produced 4-path is only a valid **geometric path**. If a strict V-pixel-only 4-path is required, you must search for an entirely different route between the endpoints using BFS/DFS, and it may not exist at all.

Visual Proof



2.15 – Shortest Paths Between p and q

Problem Statement

Consider the image segment shown.

	col 0	col 1	col 2	col 3
row 0	3	1	2	**1 (q)**
row 1	2	2	0	2
row 2	1	2	1	1
row 3	**1 (p)**	0	1	2

(a) Let $V = \{0, 1\}$ and compute the lengths of the shortest 4-, 8-, and m-path between p and q. If a particular path does not exist between these two points, explain why.
(b) Repeat for $V = \{1, 2\}$.

Step 0: Label the Grid

Coordinates (row, col) starting from $(0, 0)$ at top-left:

$$p = (3, 0),\ \text{value} = 1 \qquad q = (0, 3),\ \text{value} = 1$$

Full grid for reference:

$$\begin{pmatrix} 3 & 1 & 2 & 1 \\ 2 & 2 & 0 & 2 \\ 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix} \quad \begin{matrix} \leftarrow \text{row 0 } (q \text{ at col 3}) \\ \leftarrow \text{row 1} \\ \leftarrow \text{row 2} \\ \leftarrow \text{row 3 } (p \text{ at col 0}) \end{matrix}$$

Part (a): $V = \{0, 1\}$

Only pixels with values 0 or 1 can be used.

Usable pixels: $(0,1)$=1, $(0,3)$=1, $(1,2)$=0, $(2,0)$=1, $(2,2)$=1, $(2,3)$=1, $(3,0)$=1, $(3,1)$=0, $(3,2)$=

Shortest 4-path ($V = \{0, 1\}$)

Check the 4-neighbors of $q = (0, 3)$:

- $(0, 2) = 2 \notin V$ (left)
- $(1, 3) = 2 \notin V$ (below)

$q = (0, 3)$ has **no 4-neighbors with values in** V — it is a completely isolated island. No 4-path can ever arrive at q.

Result: No 4-path exists.

Shortest 8-path ($V = \{0, 1\}$)

With diagonals allowed, note that $(1, 2) = 0 \in V$ is diagonally adjacent to $q = (0, 3)$:

$$D_8\big((1,2),(0,3)\big) = \max(|1-0|, |2-3|) = 1 \quad \checkmark$$

Working backward from q: $(0,3) \leftarrow (1,2) \leftarrow (2,2) \leftarrow (3,2) \leftarrow (3,1) \leftarrow (3,0)$. This gives the candidate path:

$$(3,0) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,3)$$

Check each step:

- $(3,0) \rightarrow (3,1)$: 4-conn (horizontal), $(3,1) = 0 \in V$. ✓
- $(3,1) \rightarrow (3,2)$: 4-conn (horizontal), $(3,2) = 1 \in V$. ✓
- $(3,2) \rightarrow (2,2)$: 4-conn (vertical), $(2,2) = 1 \in V$. ✓
- $(2,2) \rightarrow (1,2)$: 4-conn (vertical), $(1,2) = 0 \in V$. ✓
- $(1,2) \rightarrow (0,3)$: diagonal, $D_8 = 1$, $(0,3) = 1 \in V$. ✓

Length = 5. But can we do better? Notice $(3,1)$ is 8-adjacent to $(2,2)$ diagonally:

$$(3,0) \rightarrow (3,1) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,3)$$

- $(3,1) \rightarrow (2,2)$: diagonal, $D_8 = \max(1,1) = 1$, $(2,2) = 1 \in V$. ✓

Length = 4. Is there a shorter (length-3) path? $D_8(p,q) = \max(3,3) = 3$, and the only length-3 route would require stepping through $(2,1) = 2 \notin V$. **Shortest 8-path length = 4.**

Shortest m-path ($V = \{0,1\}$)

Start with the 8-path $(3,0) \rightarrow (3,1) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,3)$ and check each diagonal:

Step $(3,1) \rightarrow (2,2)$ *(diagonal):* Shared 4-neighbors are $(3,2)$ and $(2,1)$:

$$(3,2) = 1 \in V \quad \Rightarrow \text{diagonal } \textbf{BLOCKED}$$

Must route through the shared pixel: $(3,1) \rightarrow (3,2) \rightarrow (2,2)$. Insert $(3,2)$.

Step $(1,2) \rightarrow (0,3)$ *(diagonal):* Shared 4-neighbors are $(0,2)$ and $(1,3)$:

$$(0,2) = 2 \notin V \quad \text{and} \quad (1,3) = 2 \notin V \quad \Rightarrow \text{diagonal } \textbf{ALLOWED}✓$$

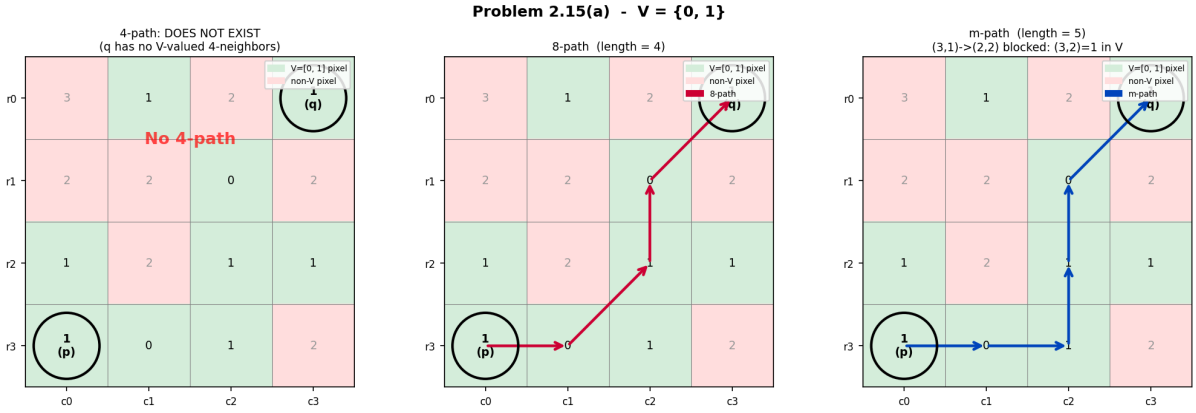
Final m-path (with one extra step inserted):

$$(3,0) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,3)$$

All steps: 4-conn, 4-conn, 4-conn, 4-conn, diagonal (allowed). **Length = 5.**

Part (a): V

- **4-path:** Does **NOT** exist. $q = (0,3)$ has no 4-neighbors in V — it is unreachable by any 4-connected path.
- **8-path:** Shortest length = **4**. Path: $(3,0) \rightarrow (3,1) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,3)$.
- **m-path:** Shortest length = **5**. Path: $(3,0) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (2,2) \rightarrow (1,2) \rightarrow (0,3)$.
  The diagonal $(3,1) \rightarrow (2,2)$ is blocked because shared neighbor $(3,2) = 1 \in V$, so one extra step is needed. The final diagonal $(1,2) \rightarrow (0,3)$ is allowed since both shared neighbors $(0,2) = 2$ and $(1,3) = 2$ are $\notin V$.



Part (b): $V = \{1, 2\}$

Usable pixels have values 1 or 2.

Non-usable: $(0,0) = 3 \notin V$, $(1,2) = 0 \notin V$, $(3,1) = 0 \notin V$.

All remaining 13 pixels are usable.

Shortest 4-path ($V = \{1, 2\}$)

From $p = (3,0)$: only 4-neighbor in V is $(2,0) = 1$ (since $(3,1) = 0 \notin V$).

A valid path:

$$(3,0) \to (2,0) \to (2,1) \to (2,2) \to (2,3) \to (1,3) \to (0,3)$$

Values: $1, 1, 2, 1, 1, 2, 1$ — all $\in V$. Manhattan distance $= |3-0| + |0-3| = 6$, so this achieves the minimum. **Length = 6.**

Shortest 8-path ($V = \{1, 2\}$)

Using diagonals:

$$(3,0) \to (2,0) \to (1,1) \to (0,2) \to (0,3)$$

- $(3,0) \to (2,0)$: 4-conn (vertical), $(2,0) = 1 \in V$. ✓
- $(2,0) \to (1,1)$: diagonal, $D_8 = 1$, $(1,1) = 2 \in V$. ✓
- $(1,1) \to (0,2)$: diagonal, $D_8 = 1$, $(0,2) = 2 \in V$. ✓
- $(0,2) \to (0,3)$: 4-conn (horizontal), $(0,3) = 1 \in V$. ✓

Length = 4. The only length-3 path would need $(1,2) = 0 \notin V$ as a step, so length-3 is impossible. **Length = 4.**

Shortest m-path ($V = \{1, 2\}$)

Check both diagonals in the 8-path above:

Step $(2,0) \to (1,1)$ *(diagonal):* Shared 4-neighbors: $(1,0)$ and $(2,1)$:

$$(1,0) = 2 \in V \quad \Rightarrow \text{diagonal } \textbf{BLOCKED}$$

Step $(1,1) \to (0,2)$ *(diagonal):* Shared 4-neighbors: $(0,1)$ and $(1,2)$:

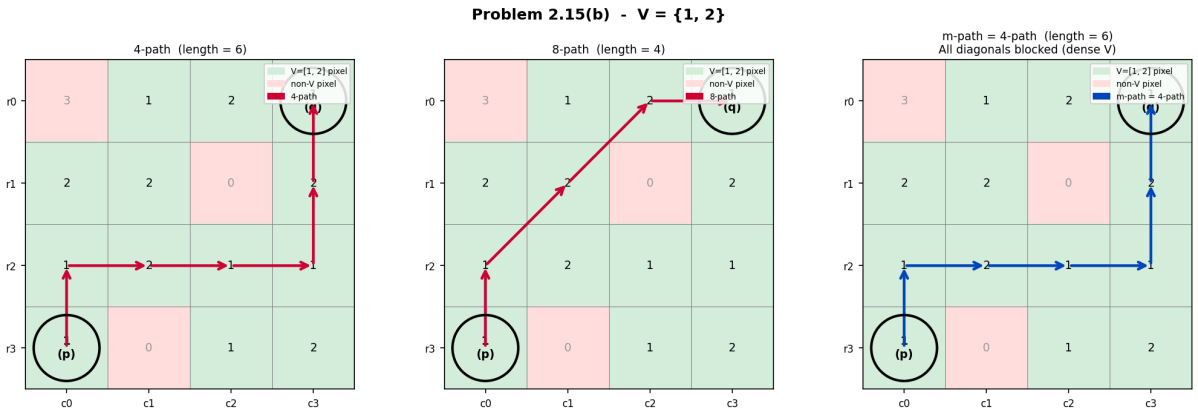
$$(0,1) = 1 \in V \quad \Rightarrow \text{diagonal } \textbf{BLOCKED}$$

Exhaustive check of all other diagonal pairs in V: every candidate diagonal has at least one shared 4-neighbor in V (since most pixels have value 1 or 2 with $V = \{1, 2\}$). The only V pixels that could serve as "safe corners" are $(1, 2) = 0$ and $(3, 1) = 0$, but neither creates a valid all-V diagonal path toward q.

No m-adjacent diagonal step exists in any path from p to q. The m-path must use only 4-connections, giving the same length as the 4-path. **Length = 6.**

Part (b): V

- **4-path:** Shortest length = **6**. E.g., $(3, 0) \to (2, 0) \to (2, 1) \to (2, 2) \to (2, 3) \to (1, 3) \to (0, 3)$.
- **8-path:** Shortest length = **4**. Path: $(3, 0) \to (2, 0) \to (1, 1) \to (0, 2) \to (0, 3)$.
- **m-path:** Shortest length = **6**. All diagonal steps are blocked because $V = \{1, 2\}$ is dense — every candidate diagonal has a shared 4-neighbor $\in V$, so no diagonal shortcut is valid under m-adjacency rules.



2.16 – D_4 Distance and 4-Path

Problem Statement

(a) Give the condition(s) under which the D_4 distance between two points p and q is equal to the shortest 4-path between these points.
(b) Is this path unique?

Step 1: Review D_4 Distance

The D_4 distance (city-block distance or Manhattan distance) between two pixels $p = (x, y)$ and $q = (s, t)$ is:

$$D_4(p, q) = |x - s| + |y - t|$$

This is literally: how far apart are they horizontally + how far apart are they vertically.

Example: If $p = (1, 3)$ and $q = (5, 6)$, then $D_4 = |1 - 5| + |3 - 6| = 4 + 3 = 7$.

Step 2: What is the Shortest 4-Path?

A 4-path is a sequence of pixels where consecutive pixels are 4-adjacent (share an edge — horizontal or vertical neighbors only). The "shortest 4-path" is the one with the fewest steps (fewest intermediate pixels).

Key observation: The shortest 4-path between p and q depends on which pixels are allowed to be in the path (i.e., which pixels have values in V). The D_4 distance, on the other hand, is a pure geometric/mathematical formula that ignores pixel values entirely.

Part (a): When Does D_4 = Shortest 4-Path?

(a)

The D_4 distance between p and q equals the length of the shortest 4-path between them when the path is **unconstrained** — meaning:

1. All pixels along the path (between p and q) have values in V (no obstacles), OR equivalently, the V set is unrestricted or all pixel values belong to V.
2. The path can proceed freely without being blocked by pixels with values outside V.

In other words, when every pixel is a valid step (no barriers), the shortest 4-path length is exactly $|x - s| + |y - t| = D_4(p, q)$.

Simple version: The D_4 distance equals the shortest 4-path when the set V includes all intensity values in the image (or when no pixel along a Manhattan-distance-optimal route has a value outside V).

Part (b): Is This Path Unique?

(b)

No, the path is generally NOT unique.
When the shortest 4-path length equals $D_4(p, q)$, the path has $|x - s|$ horizontal moves and $|y - t|$ vertical moves. These moves can be interleaved in any order, producing multiple distinct paths. The number of distinct shortest 4-paths is:

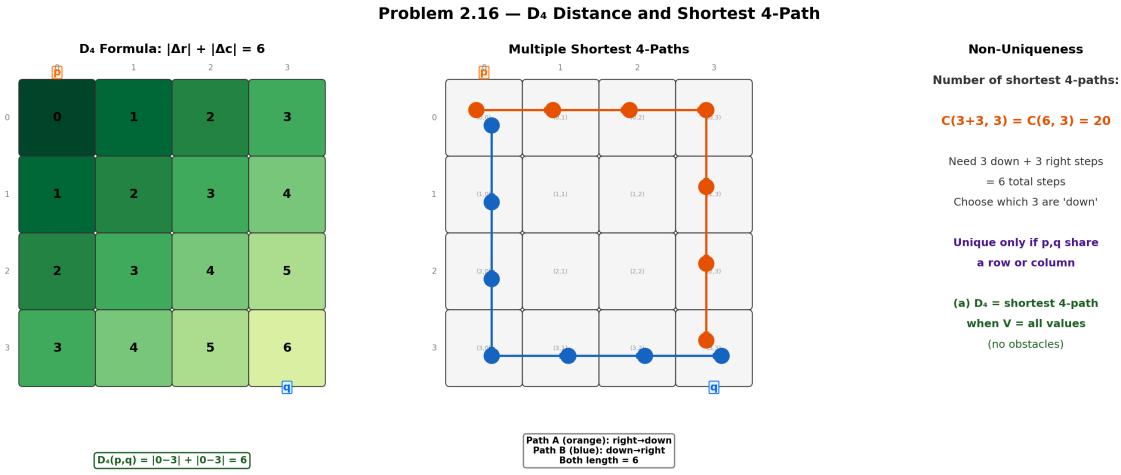
$$\binom{|x-s| + |y-t|}{|x-s|} = \frac{(|x-s| + |y-t|)!}{|x-s|! \cdot |y-t|!}$$

Example: If $p = (0, 0)$ and $q = (2, 1)$, then $D_4 = 3$. Number of paths $= \binom{3}{2} = 3$:

1. $(0, 0) \to (1, 0) \to (2, 0) \to (2, 1)$ (right, right, down)
2. $(0, 0) \to (1, 0) \to (1, 1) \to (2, 1)$ (right, down, right)
3. $(0, 0) \to (0, 1) \to (1, 1) \to (2, 1)$ (down, right, right)

The path is unique only when p and q are on the same row ($|y - t| = 0$) or same column ($|x - s| = 0$), in which case there is only one straight-line path.

Visual Proof



2.17 – D_8 Distance and 8-Path

Problem Statement

Repeat Problem 2.16 for the D_8 distance.
(a) Give the condition(s) under which the D_8 distance between two points p and q is equal to the shortest 8-path between these points.
(b) Is this path unique?

Step 1: Review D_8 Distance

The D_8 distance (chessboard distance) between two pixels $p = (x, y)$ and $q = (s, t)$ is:

$$D_8(p, q) = \max(|x - s|,\ |y - t|)$$

This is the number of moves a king in chess would need to go from p to q. Why? Because in each step along an 8-path, you can move in any of 8 directions (including diagonals). A diagonal move reduces both the horizontal and vertical distance by 1 simultaneously. So the total number of steps is the larger of the two distances — you make diagonal moves to reduce both distances, and when one distance hits 0, you continue straight in the remaining direction.

Example: If $p = (1, 3)$ and $q = (5, 6)$, then $D_8 = \max(4, 3) = 4$.

Part (a): When Does D_8 = Shortest 8-Path?

(a)

The D_8 distance between p and q equals the length of the shortest 8-path between them when the path is **unconstrained** — meaning:

1. All pixels along the path have values in V (no obstacles).
2. The path can proceed freely without detours.

This is the same condition as in Problem 2.16, but now for 8-paths. When every pixel value is in V, the shortest 8-path has length $\max(|x - s|, |y - t|) = D_8(p, q)$.

The optimal strategy (when unconstrained):

1. First, make $\min(|x - s|, |y - t|)$ diagonal moves. After this, one of the two distances reaches 0.
2. Then, make the remaining $\big||x-s|-|y-t|\big|$ moves in a straight line (horizontal or vertical) to reach q.
3. Total steps: $\min(|x - s|, |y - t|) + \big||x - s| - |y - t|\big| = \max(|x - s|, |y - t|) = D_8(p, q)$.

If pixels outside V block this direct route, the shortest 8-path may be longer than D_8 or may not exist at all.

Part (b): Is This Path Unique?

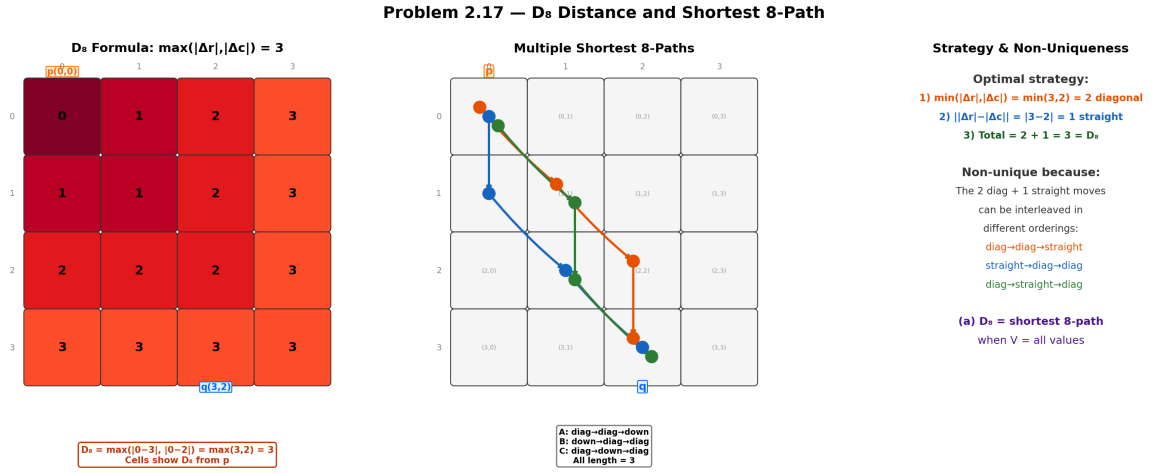
(b)

No, the path is generally NOT unique.
Consider the case where both $|x - s| > 0$ and $|y - t| > 0$. During the diagonal phase, you must make $\min(|x - s|, |y - t|)$ diagonal moves. If $|x - s| \neq |y - t|$, you also need straight-line moves, and these can be interleaved with the diagonal moves in different orders, producing multiple distinct paths.
Even when $|x-s| = |y-t|$ (all moves are diagonal), if the number of diagonal steps > 1, there may be multiple paths with the same length that take slightly different routes.
Example: $p = (0, 0)$, $q = (2, 1)$. $D_8 = \max(2, 1) = 2$. Two possible shortest 8-paths:

1. $(0, 0) \to (1, 1) \to (2, 1)$: one diagonal + one vertical. Length 2.
2. $(0, 0) \to (1, 0) \to (2, 1)$: one vertical + one diagonal. Length 2.

Both have length $2 = D_8$, demonstrating non-uniqueness.
The path is unique only when p and q are on the same row, same column, or same diagonal AND the distance is 0 or 1. In general, there are multiple shortest 8-paths.

Visual Proof



Summary of All Answers

Problem	Key Answer
2.1	Smallest dot ≈ 0.0304 mm (30.4 μm) at 0.2 m distance
2.2	Brightness adaptation: transition from photopic (cone) vision to scotopic (rod) vision; rods bleached, rhodopsin regenerates in 20–30 min
2.5	Camera resolves ≈ 5.12 line pairs/mm on the scene
2.10	2-hour HDTV movie $\approx 1.1664 \times 10^{13}$ bits (≈ 1.458 TB)
2.11	S_1, S_2: (a) NOT 4-adjacent; (b) 8-adjacent; (c) m-adjacent
2.12	Replace each diagonal step with two 4-connected steps (H + V)
2.13	Same as 2.12; m-adjacency guarantees inserted pixels $\notin V$
2.15(a)	$V = \{0, 1\}$: 4-path DNE (q isolated); 8-path length=4; m-path length=5
2.15(b)	$V = \{1, 2\}$: 4-path length=6; 8-path length=4; m-path length=6
2.16	D_4 = shortest 4-path when V = all values; path **not unique**
2.17	D_8 = shortest 8-path when V = all values; path **not unique**