Datastructures and Algorithms

Sweepline, Closest Point Pair DFS, BFS, Shortest Path Problems

Adel Gavranović — ETH Zürich — 2025

Overview

Learning Objectives

Geometric Algorithms

- Sweepline
- Geometric Divide & Conquer: Closest Point Pair

Graphs

- Graphs: DFS and BFS
- Appendix: Real World Shortest Path Problems

n.ethz.ch/~agavranovic

Material

Webpage

Mail

1. Follow-up

Follow-up from last session

Bonus Exercise I “Image Segmentation”

- Try to get your implementation to run in $\mathcal{O}(n)$ (you know how!)

[p. 17] Apparent change of member variable despite `const` keyword

- Amazingly, `const` functions can change reference types (to some degree)!
- Since it’s a reference to a variable (`count`) *outside* of the class, the function *is allowed to* change it! (I think this has something to do with the fact that references are really just pointers in disguise but don’t quote me on this)

Follow-up from last session

[p. 31] 2D cross product ($\times$)

- It's not a vector, since we *defined* it via a determinant

2. Feedback regarding **code** expert

General things regarding **code** expert

Any questions regarding **code** expert on your part?

3. Learning Objectives

Objectives

- ☐ Understand the shown sweepline-based algorithm
- ☐ Understand the shown recursive algorithm for finding the shortest pair distance
- ☐ Know when which representation for graphs is more suitable and why

4. Summary

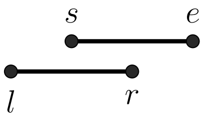
Getting on the same page

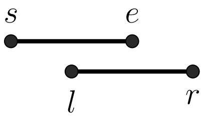
- How far did you get with graphs?
- Representations of Graphs?
- BFS, DFS?

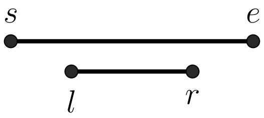
5. Geometric Algorithms

Overlaps of two intervals

Two intervals (l, r) and (s, e) overlap if





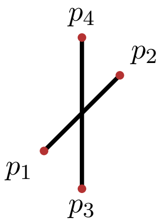


$l \leq s < r$ $\quad$ $l < e \leq r$ $\quad$ or $s \leq l < r \leq e$

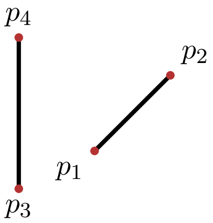
$\Rightarrow$ We can check in constant time whether two intervals intersect.

Intersection of two line segments

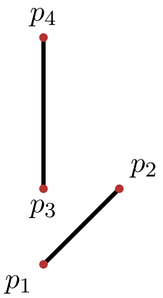
How to figure out whether two segments are intersecting without actually computing the intersection points (division!)?



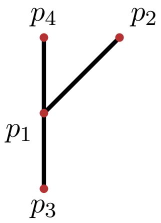
Intersection: p_1 and p_2 opposite w.r.t $\overline{p_3p_4}$ and p_3, p_4 opposite w.r.t. $\overline{p_1p_2}$



No intersection: p_1 and p_2 on the same side of $\overline{p_3p_4}$



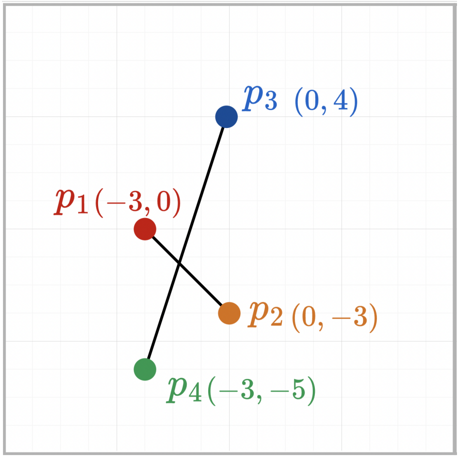
No intersection: p_3 and p_4 on the same side of $\overline{p_1p_2}$



Intersection: p_1 on $\overline{p_3p_4}$

Intersection of two line segments

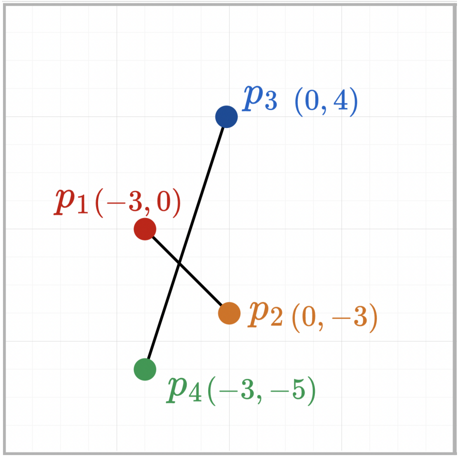
Part (a)



Intersection or no intersection?

Intersection of two line segments

Part (a)



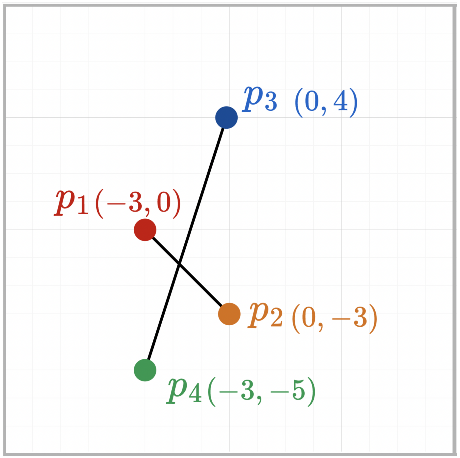
Intersection or no intersection?

Intersection

p_1, p_2 are opposite w.r.t $\overline{p_4 p_3}$,
and p_3, p_4 are opposite w.r.t. $\overline{p_1 p_2}$.

Intersection of two line segments

Part (a)

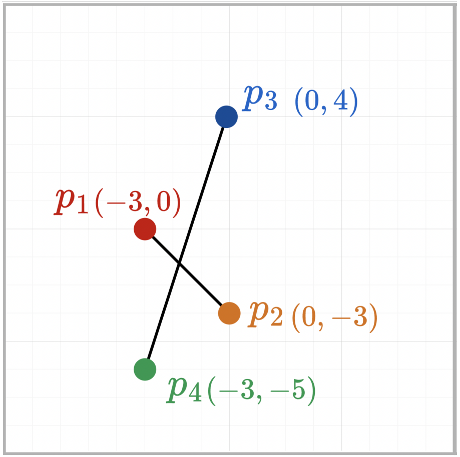


$(p_3 - p_4) \times (p_1 - p_4) =$
$= ((0,4) - (-3,-5)) \times ((-3,0) - (-3,-5)) = (3,9) \times (0,5) = \det\begin{bmatrix} 3 & 0 \\ 9 & 5 \end{bmatrix}$
$= (3)(5) - (0)(9) = \mathbf{15} > \mathbf{0}.$

$(p_3 - p_4) \times (p_2 - p_4) =$
$= ((0,4) - (-3,-5)) \times ((0,-3) - (-3,-5))$
$= (3,9) \times (3,2) = \det\begin{bmatrix} 3 & 3 \\ 9 & 2 \end{bmatrix}$
$= (3)(2) - (3)(9) = -\mathbf{21} < \mathbf{0}.$

Intersection of two line segments

Part (a)

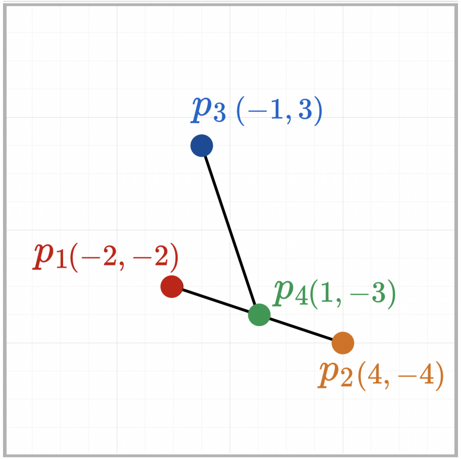


and $(p_2 - p_1) \times (p_3 - p_1) =$
$= ((0,-3) - (-3,0)) \times ((0,4) - (-3,0))$
$= (3,-3) \times (3,4) = \det \begin{bmatrix} 3 & 3 \\ -3 & 4 \end{bmatrix}$
$= (3)(4) - (3)(-3) = \mathbf{21} > \mathbf{0}.$

$(p_2 - p_1) \times (p_4 - p_1) =$
$= ((0,-3) - (-3,0)) \times ((-3,-5) - (-3,0))$
$= (3,-3) \times (0,-5) = \det \begin{bmatrix} 3 & 0 \\ -3 & -5 \end{bmatrix}$
$= (3)(-5) - (0)(-3) = -\mathbf{15} < \mathbf{0}.$

Intersection of two line segments

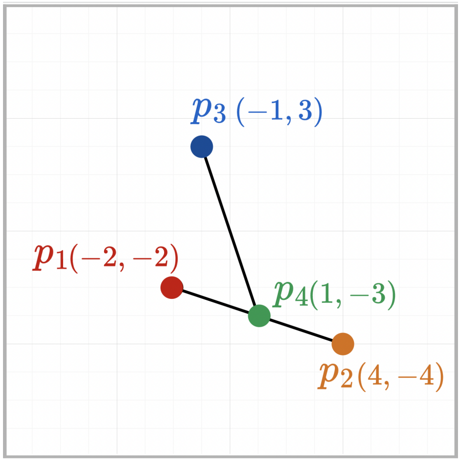
Part (b)



Intersection or no intersection?

Intersection of two line segments

Part (b)



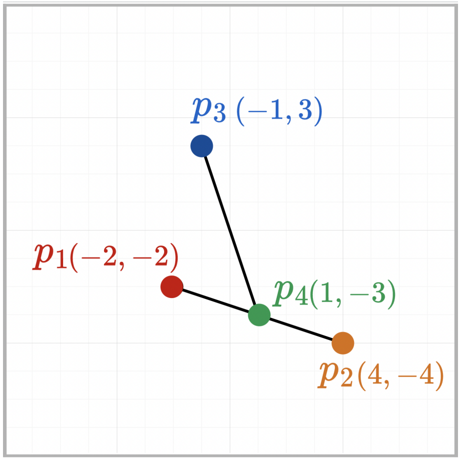
Intersection or no intersection?

Intersection

p_4 is on $\overline{p_1 p_2}$ for two reasons.

Intersection of two line segments

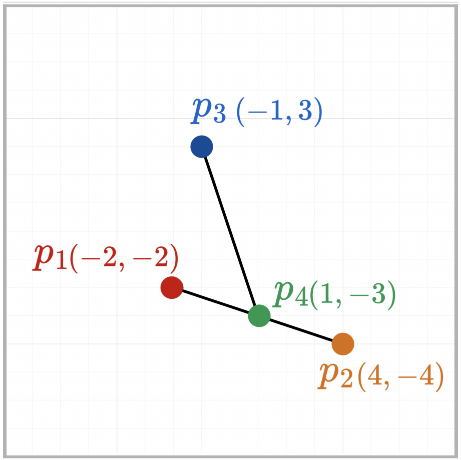
Part (b)



First,
$(p_2 - p_1) \times (p_4 - p_1) =$
$= ((4,-4) - (-2,-2)) \times ((1,-3) - (-2,-2))$
$= (6,-2) \times (3,-1) = \det\begin{bmatrix} 6 & 3 \\ -2 & -1 \end{bmatrix}$
$= (6)(-1) - (3)(-2) = \mathbf{0}.$

Intersection of two line segments

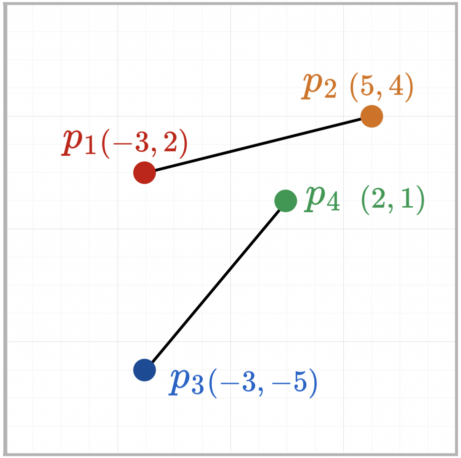
Part (b)



But this only shows that p_4 is in the line created by $\overline{p_1 p_2}$.
To conclude that p_4 is in $\overline{p_1 p_2}$, note that
$-2 = p_1[0] \leq 1 = p_4[0] \leq 4 = p_2[0]$
and
$-4 = p_2[1] \leq -3 = p_4[1] \leq -2 = p_1[1]$.

Intersection of two line segments

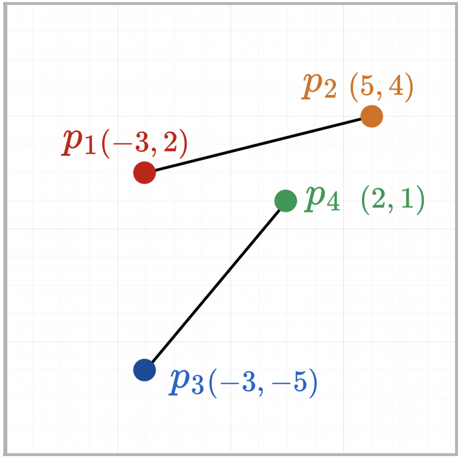
Part (c)



Intersection or no intersection?

Intersection of two line segments

Part (c)



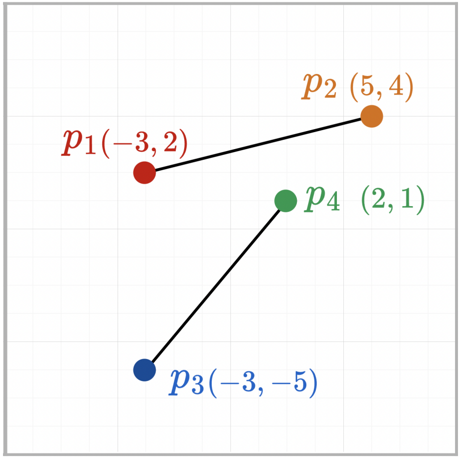
Intersection or no intersection?

No Intersection

p_3 and p_4 are on the same side of $\overline{p_1p_2}$.

Intersection of two line segments

Part (c)

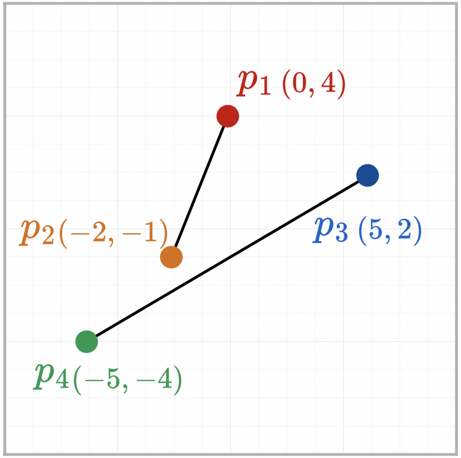


$$
\begin{aligned}
&(p_2 - p_1) \times (p_3 - p_1) = \\
&= ((5,4)-(-3,2))\times((-3,-5)-(-3,2)) \\
&= (8,2) \times (0,-7) = \det \begin{bmatrix} 8 & 0 \\ 2 & -7 \end{bmatrix} \\
&= (8)(-7) - (0)(2) = -\mathbf{56} < \mathbf{0}.
\end{aligned}
$$

$$
\begin{aligned}
&(p_2 - p_1) \times (p_4 - p_1) = \\
&= ((5,4) - (-3,2)) \times ((2,1) - (-3,2)) \\
&= (8,2) \times (5,-1) = \det \begin{bmatrix} 8 & 5 \\ 2 & -1 \end{bmatrix} \\
&= (8)(-1) - (5)(2) = -\mathbf{18} < \mathbf{0}.
\end{aligned}
$$

Intersection of two line segments

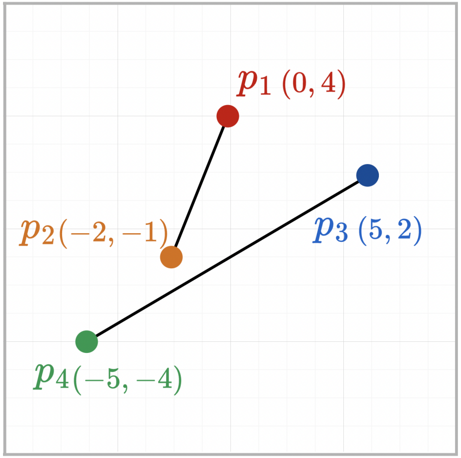
Part (d)



Intersection or no intersection?

Intersection of two line segments

Part (d)



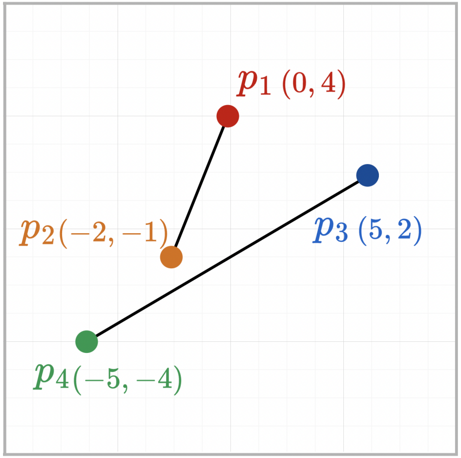
Intersection or no intersection?

No Intersection

p_1 and p_2 are on the same side of $\overline{p_4 p_3}$.

Intersection of two line segments

Part (d)



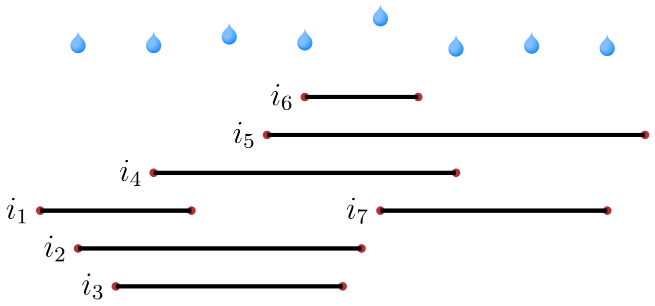
$(p_3 - p_4) \times (p_1 - p_4) =$
$= ((5,2)-(-5,-4)) \times ((0,4)-(-5,-4))$
$= (10,6) \times (5,8) = \det \begin{bmatrix} 10 & 5 \\ 6 & 8 \end{bmatrix}$
$= (10)(8) - (5)(6) = \mathbf{60} > \mathbf{0}.$

$(p_3 - p_4) \times (p_2 - p_4) =$
$= ((5,2) - (-5,-4)) \times ((-2,-1) - (-5,-4))$
$= (10,6) \times (3,3) = \det \begin{bmatrix} 10 & 3 \\ 6 & 3 \end{bmatrix}$
$= (10)(3) - (6)(3) = \mathbf{12} > \mathbf{0}.$

5. Geometric Algorithms

5.1. Sweepline

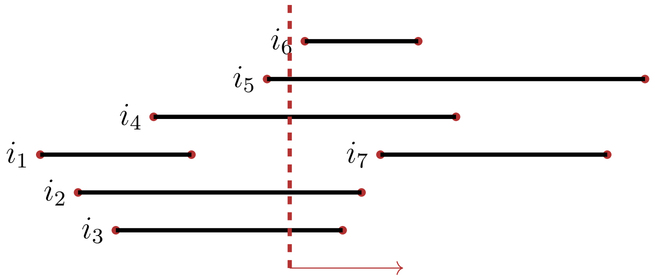
Preparation: Horizontal Line Segments



Questions:

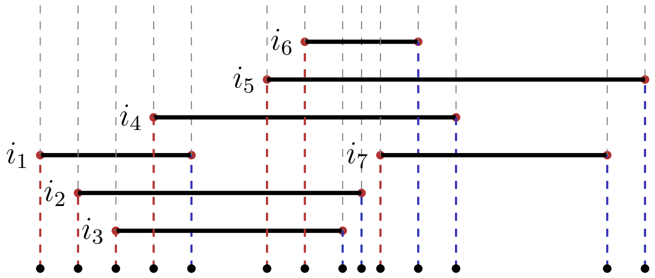
- What is the maximum number of overlapping segments?
- Which line segments (don't) get wet?
- Which line segments are neighbours?

Preparation: Horizontal Line Segments



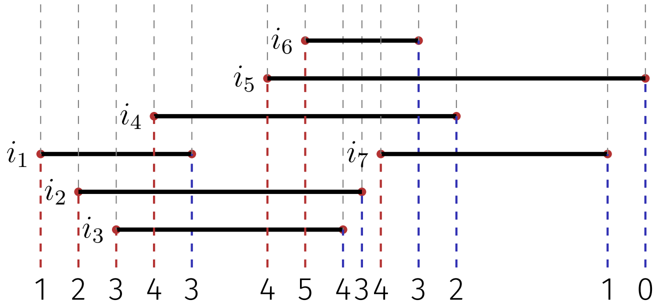
Idea of a sweep line: vertical line, moving in x-direction, observes the geometric objects.

Preparation: Horizontal Line Segments



Event list: list of points where the state observed by the sweepline changes.

Preparation: Overlapping Line Segments

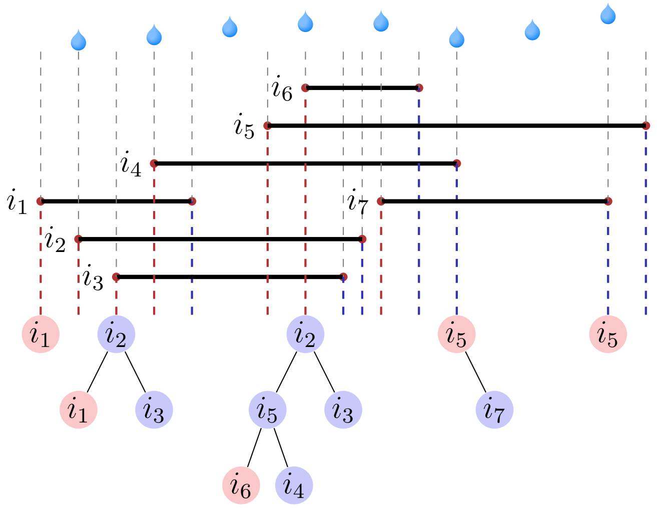


Q: What is the maximum number of overlapping segments?

Sweep line controls a counter that is incremented (decremented) at the left (right) end point of a line segment.

A: maximum counter value

Preparation: Top-Most Line Segments

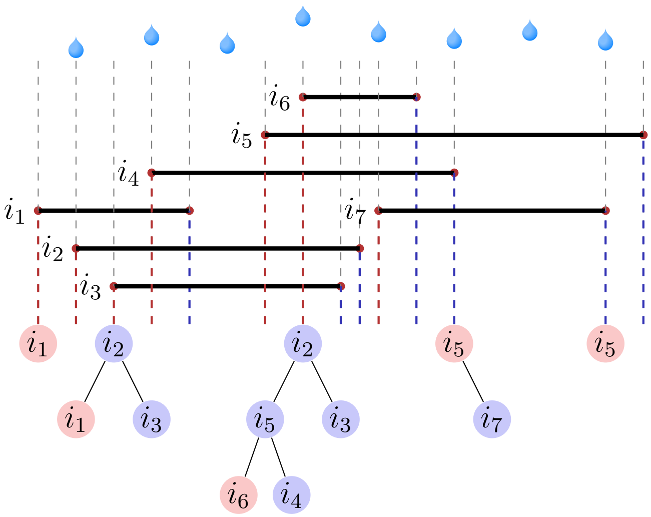


Q: Which line segments get wet?

Sweep line controls a **binary search tree** that comprises the line segments according to their vertical ordering.

A: Line segments on the very left of the tree.

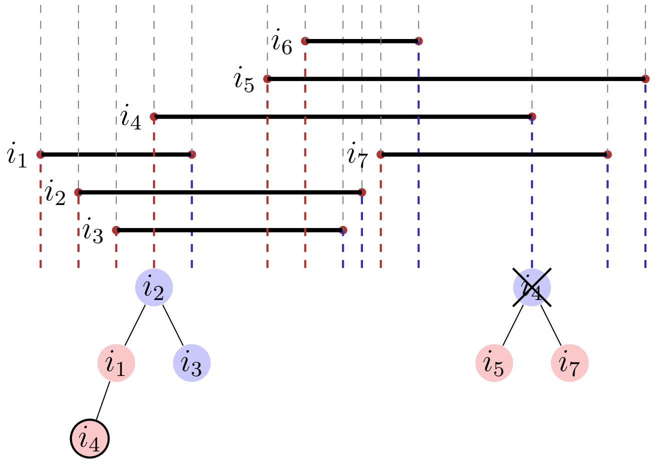
Preparation: Top-Most Line Segments



Q: Why don't we use Max-Heap (instead of BST)?

A: The deletion of an arbitrary element (not the maximum) from a heap is not easy.

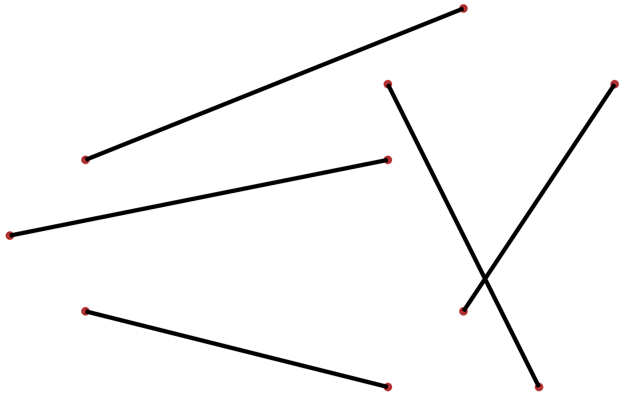
Preparation: Neighboring Line Segments



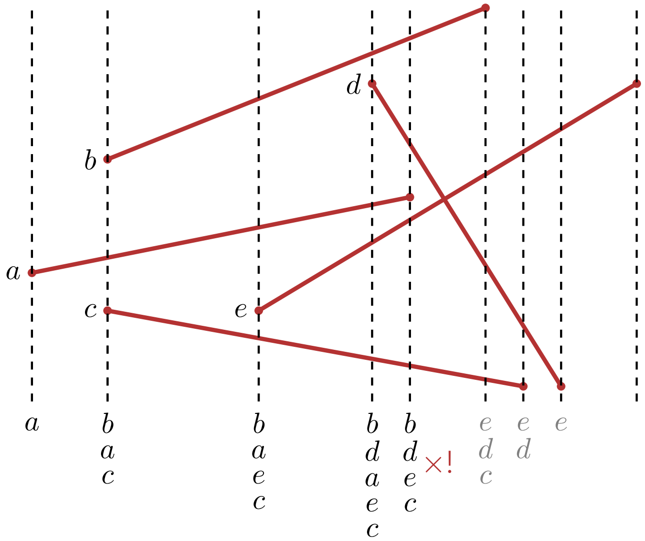
Q: Which line segments are neighbours?

A: Line segments that lie next to each other (symmetric predecessor/successor) at the beginning of a line segment or when another line segment ends.

Cutting many line segments



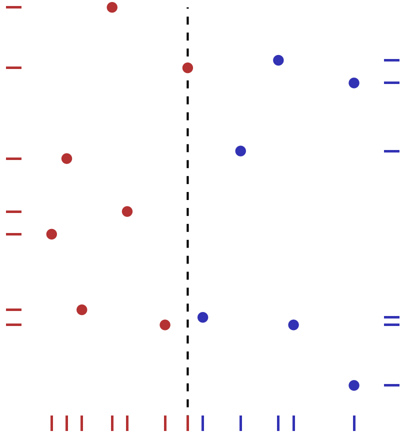
Intersection of line segments



5.2. Geometric Divide & Conquer: Closest Point Pair

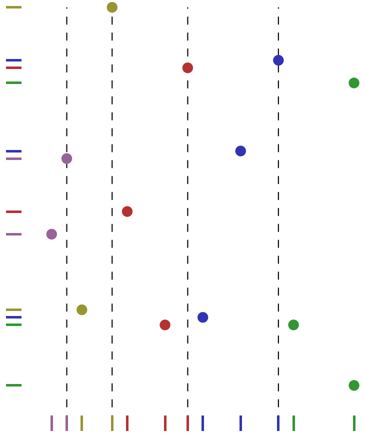
Divide And Conquer: Closest Point Pair

- Set of points P, starting with $P \leftarrow Q$
- Arrays X and Y, containing the elements of P , sorted by x- and y-coordinate, respectively.
- Partition point set into two (approximately) equally sized sets P_L and P_R, separated by a vertical line through a point of P.
- Split arrays X and Y accordingly in X_L, X_R. Y_L and Y_R.



Divide And **Conquer**: Closest Point Pair

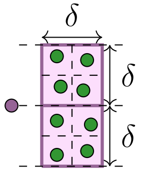
- Recursive call with P_L, X_L, Y_L and P_R, X_R, Y_R. Yields minimal distances δ_L, δ_R.
- (If only $k \leq 2$ points: compute the minimal distance directly)
- After recursive call $\delta = \min(\delta_L, \delta_R)$. Combine (next slides) and return best result.



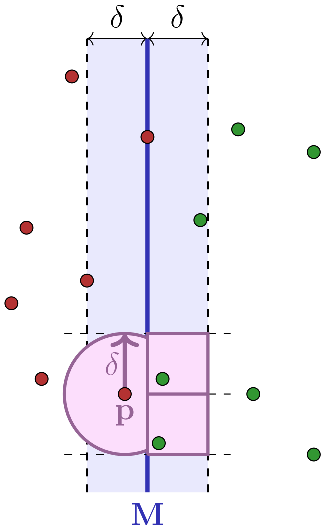
Minimum Distance across middle line: Observations

Which points are relevant for point p?
$\Rightarrow$ the ones in a circle around p with radius δ
Observation 1: The relevant points are contained in two $(\delta \times \delta)$-rectangles.

How many points are in these rectangles?
Observation 2: At most 8.



At most one point per $(\delta/2 \times \delta/2)$-rectangle, otherwise they have distance $\sqrt{2} \cdot \frac{\delta}{2} < \delta$.



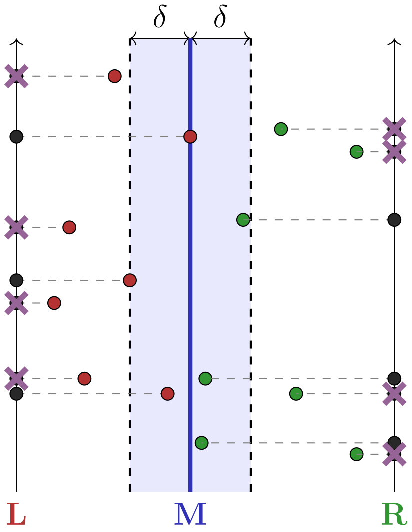
Minimum Distance across middle line: Algorithm

- sort L and R according to y-coordinates
- filter L and R according to band around M
- for every remaining point in L, compute distance to all points in R in the strip with y-distance $\leq \delta$
  $\Rightarrow$ at most 8 points

Running time:

- Sorting: $\Theta(n \log n)$
- Filtering: $\Theta(n)$
- compute the distances: $\Theta(n)$

$\Rightarrow \Theta(n \log n)$ per recursion step



Implementation

- Goal: recursion equation (runtime) $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(n)$.
- Non-trivial: only arrays Y and Y'
- Idea: merge reversed: run through Y that is presorted by y-coordinate. For each element follow the selection criterion of the x-coordinate and append the element either to Y_L or Y_R. Same procedure for Y'. Runtime $\mathcal{O}(|Y|)$.

Overall runtime: $\mathcal{O}(n \log n)$.

Questions

- How does the algorithm compare to a brute-force approach?
  - Divide and conquer reduces the problem size at each step, resulting in a time complexity of $\mathcal{O}(n \log n)$, while a brute-force approach has a time complexity of $\mathcal{O}(n^2)$.
- Why do we avoid sorting at each step of the recursion?
  - Sorting is $\mathcal{O}(n \log n)$ and the time complexity of merging should be linear.

6.1. Graphs: DFS and BFS

Quiz: Runtimes of simple Operations

Operation	Matrix	List
$(v, u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
find all edges $e \in E$	$\Theta(n^2)$	$\Theta(n + m)$
Insert edge	$\Theta(1)$	$\Theta(1)$
Delete edge	$\Theta(1)$	$\Theta(\deg^+ v)$

Quiz #1

Question

Which graph representation, adjacency matrix or adjacency list, is more suitable for representing a graph with a high number of edges compared to vertices?

Answer

For very dense graphs, when the number of edges is close to n^2, an adjacency matrix is more suitable; the space complexity of an adjacency matrix is $\Theta(n^2)$, which is independent of the number of edges.

Quiz #2

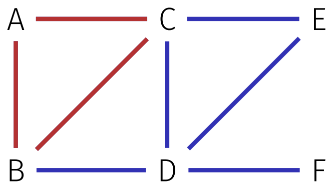
Question

When would it be more appropriate to use an adjacency matrix representation rather than an adjacency list representation? Provide another example scenario.

Answer

For example, in a scenario where you need to frequently check the presence of an edge or update edges between vertices, an adjacency matrix would be more suitable due to its $\Theta(1)$ edge lookup, insertion, and deletion time complexity.

Quiz #3



We want to count the number of triangles (cycles with 3 nodes and edges) in a graph G.

In what time can we do this with an adjacency matrix? How about an adjacency list?

Quiz #3 Solution

Adjacency matrix: $\Theta(n^2 + m \cdot n)$

Naively: $\Theta(n^3)$: check for each of the $\binom{n}{3}$ combinations of 3 nodes whether the corresponding 3 edges are there.

Efficient: for every edge and every additional node, check whether the two additional edges are there.

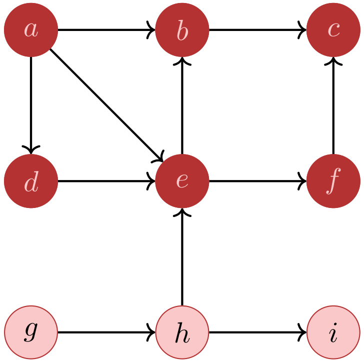
Adjacency list: $\Theta(n \cdot m)$ with $\Theta(n)$ additional memory or $\Theta(n^2 \cdot m)$

Naively: $\Theta(n^2 \cdot m)$: for every edge $e = \{u, v\}$ and every potential third node w, we go through the two lists $A[u]$ and $A[v]$ to see whether w is a neighbor of both.

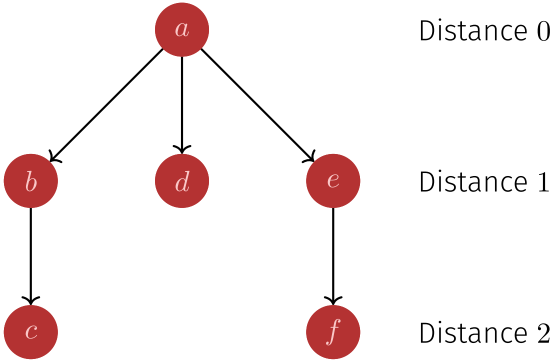
Efficient: go through $A[u]$, store the neighbors in a bitmap of length n, then for each neighbor v construct the bitmap of v and compare. So we are effectively comparing $\Theta(m)$ bitmaps of length n.

Breadth-First-Search BFS

BFS starting from a:

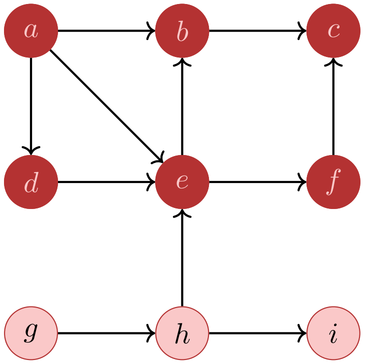


BFS-Tree: Distances and Parents

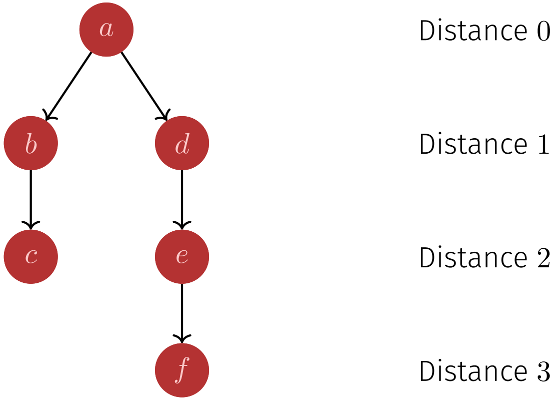


Depth-First-Search DFS

DFS starting from a:



DFS-Tree: Distances and Parents



Detect Cycles

Cycle Detection

How can you detect cycles in a graph? Explain the process for undirected and directed graphs.

Detect Cycles

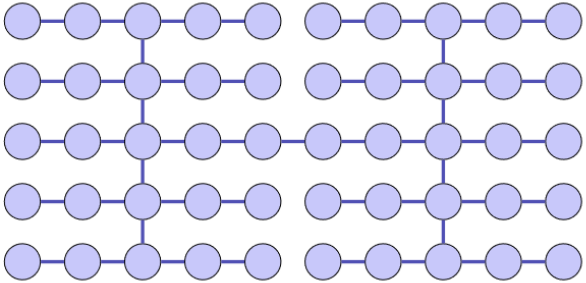
DFS Cycle Detection

- Start DFS traversal from an arbitrary node
- undirected: If a visited node is encountered again (excluding the immediate parent), a cycle exists.
- directed: If an edge to a grey node is found, a directed cycle exists.

Exam Question Example

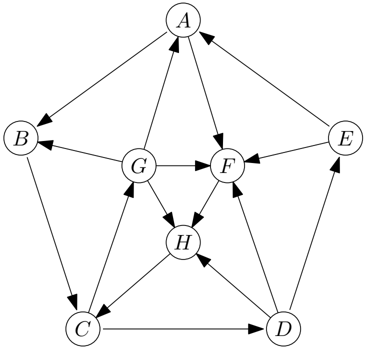
Was ist die maximale Rekursionstiefe der (rekursiv implementierten) Funktion DFS angewendet auf folgenden Graphen. Der erste Aufruf wird mitgezählt.

What is the maximum recursion depth of the (recursively implemented) function DFS in the following graph. The first call is counted.



Answer: 14

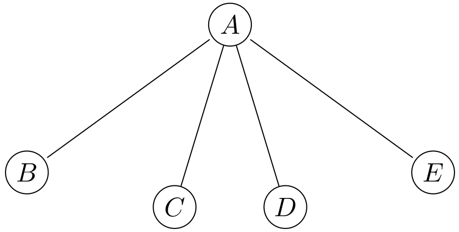
Depth-first-search and Breadth-first-search



Starting at A
DFS: A, B, C, D, E, F, H, G
BFS: A, B, F, C, H, D, G, E
There is no starting vertex where the DFS ordering equals the BFS ordering.

Depth-first-search and Breadth-first-search

Star: DFS ordering equals BFS ordering



Starting at A
DFS: A, B, C, D, E
BFS: A, B, C, D, E

Starting at C
DFS: C, A, B, D, E
BFS: C, A, B, D, E

Quiz (from an old exam): BFS/DFS

The following graph is visited with a breadth-first search and a depth-first search algorithm starting at node A. If there are several possibilities for a visiting order of the neighbours, the alphabetical order is chosen. Provide both visiting orders.

$$\begin{tikzcd}
A \arrow[r] & C \arrow[r] \arrow[d] & E \arrow[dl] \\
B \arrow[u] \arrow[ur] & D \arrow[l] \arrow[r] & F
\end{tikzcd}$$

Breadth First Search: ?

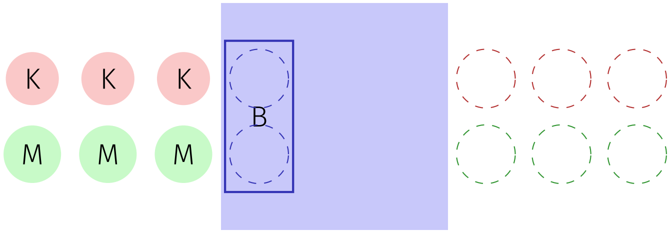
Depth First Search: ?

6.2. Appendix: Real World Shortest Path Problems

Modeling

River Crossing (Missionaries and Cannibals)

Problem: Three cannibals and three missionaries are standing at a river bank. The available boat can carry two people. At no time may at any place (banks or boat) be more cannibals than missionaries. How can the missionaries and cannibals cross the river as fast as possible? [1]



[1] There are slight variations of this problem. It is equivalent to the jealous husbands problem.

Problem as Graph

Enumerate permitted configurations as nodes and connect them with an edge, when a crossing is allowed. The problem then becomes a shortest path problem.

Example

	left	right
missionaries	3	0
cannibals	3	0
boat	x	

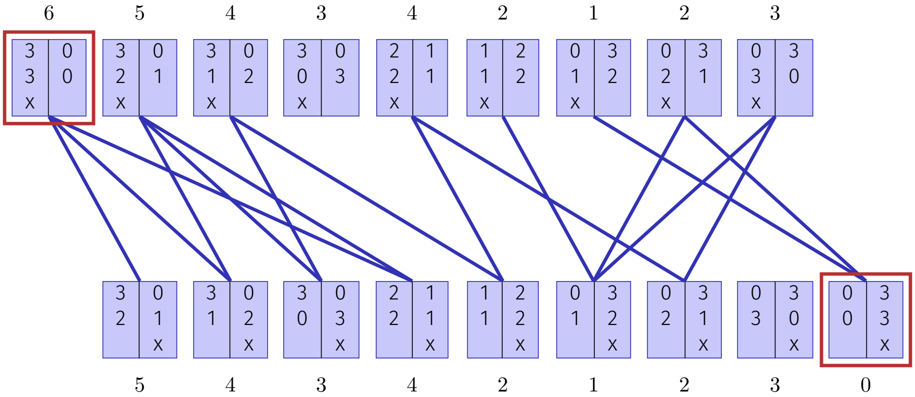
6 People on the left bank

Possible crossing

	left	right
missionaries	2	1
cannibals	2	1
boat		x

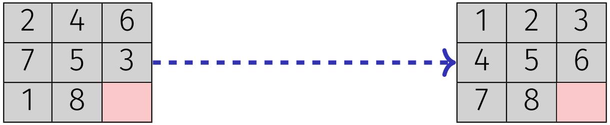
4 People on the left bank

The whole problem as a graph

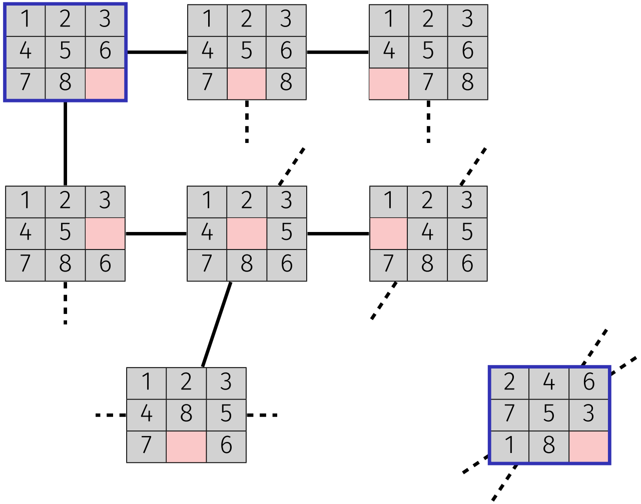


Real-World Example: Mystic Square

Fastest solution for



Problem as Graph



7. Outro

General Questions?

See you next time!

Have a nice week!