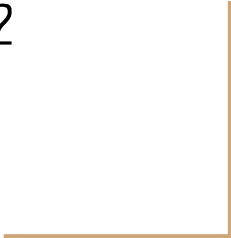


Programming, Problem Solving, and Algorithms

CPSC203, 2023 W2



Announcements

- See edX for information on Test 4.
- Reminder to book yourself a Test 4 slot next week!
- Project 1 is due soon!
- Reminder: USE THE DEBUGGER!!

Today's Plan...

1. Announcements! (5 mins)
2. Weekly Videos Review/Questions (10 mins)
3. Computing Voronoi Diagrams
 - Warning! There is a LOT we are going to try to cover today and it will rely on you having watched the pre-lecture videos...



Slides from the Assigned Videos



Everyone needs a Tim Horton

Every address in Vancouver has a nearest TH.

Partition Vancouver into regions so that points are in the same region if they have the same nearest TH.



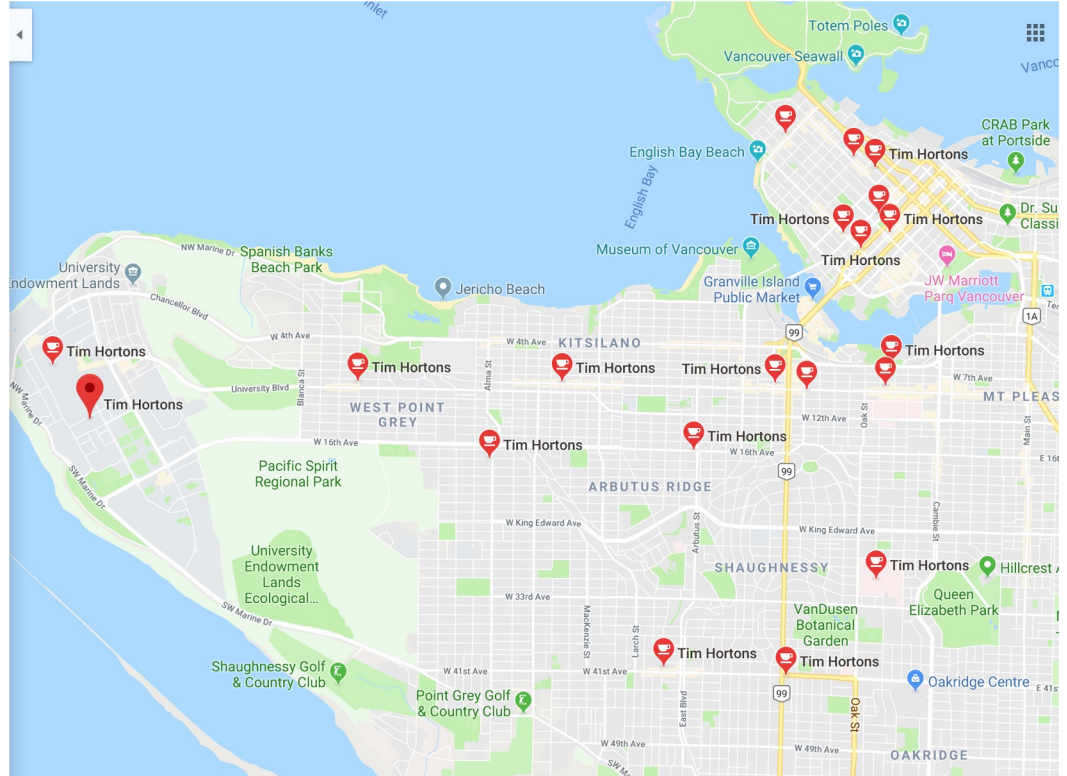


Voronoi Diagrams

Given a (finite) set of “centers” c_1, c_2, \dots, c_k , a Voronoi region, R_j consists of the set of points nearer to center c_j , than to any other center.

Together, the R_j regions compose the Voronoi Diagram of a plane.

The applications of this structure go far beyond our coffee fix!!

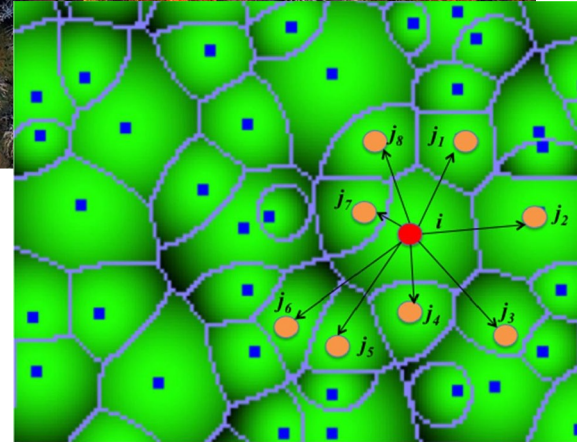


Voronoi Diagrams

Given a (finite) set of “centers” c_1, c_2, \dots, c_k , a Voronoi region, R_j consists of the set of points nearer to center c_j , than to any other center.

Together, the R_j regions compose the Voronoi Diagram of a plane.

The applications of this structure go far beyond our coffee fix!!



More examples...

Too good not to share:

https://www.khanacademy.org/partner-content/pixar/pattern/dino/v/patterns2_new



Yet more examples...

Robotics -- Path planning in the presence of obstacles

Zoology -- Model and analyze the territories of animals

Astronomy -- Identify clusters of stars and clusters of galaxies

Biology, Ecology, Forestry -- Model and analyze plant competition

Cartography -- Piece together satellite photographs into large "mosaic" maps

Geography -- Analyzing patterns of urban settlements

Marketing -- Model market of US metropolitan areas;

Metallurgy -- Modelling "grain growth" in metal films

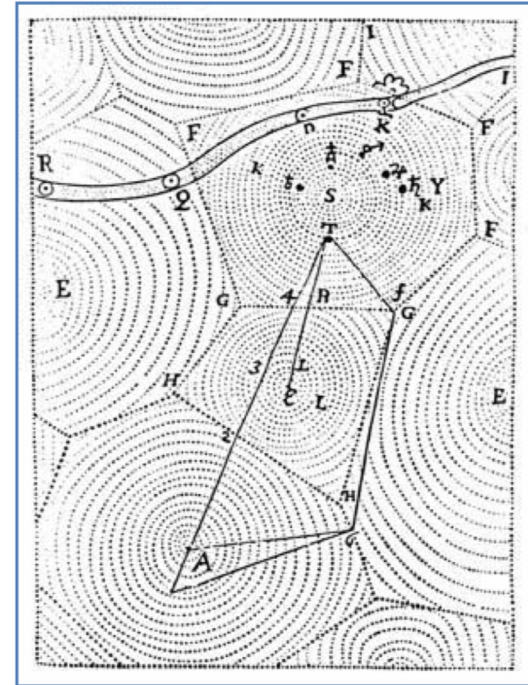
Meteorology -- Estimate regional rainfall averages, given data at discrete rain gauges

Physiology -- Analysis of capillary distribution in cross-sections of muscle tissue to compute oxygen transport ("Capillary domains")

Anthropology and Archeology -- Identify regions under the influence of different neolithic clans, chiefdoms, ceremonial centers, or hill forts.

Crystallography and Chemistry -- Study chemical properties of metallic sodium); Modelling alloy structures as sphere packings ("Domain of an atom")

Geology -- Estimation of ore reserves in a deposit using info obtained from bore holes; modelling crack patterns in basalt due to contraction on cooling



Computing Voronoi Diagrams

Choice of algorithm depends on the domain of your data. If real valued domain (like Cartesian Coords) then we use geometry. Exploit the fact that the Voronoi edges are perpendicular bisectors of the edge between 2 centers.

Our computation will always be on planar regions represented by images.

Data for the problem:

1)

2)

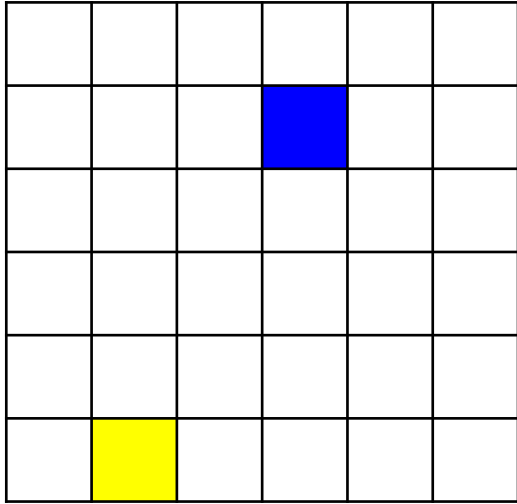
Data for the solution:

1)

2)

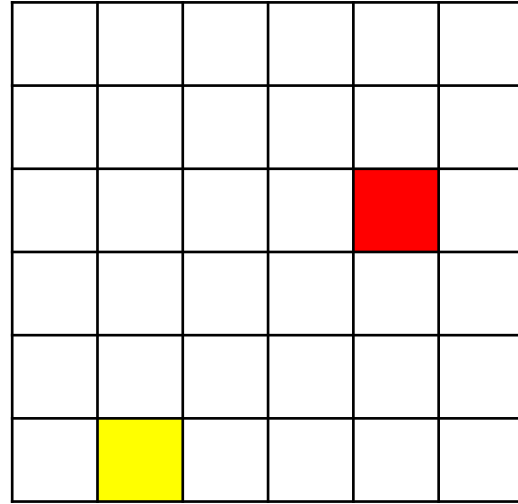
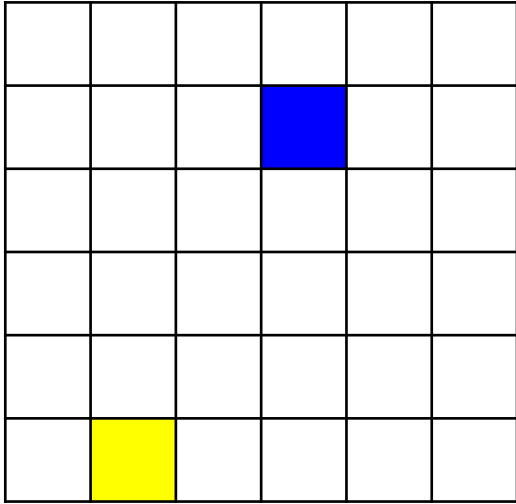
Distance between pixels

Find the distance between two pixel locations:



Distance between pixels

Given a point and two centers, determine which is the nearest center...



Pointillism



[A Sunday on La Grande Jatte, Georges Seurat](#)

The Idea...



- 1) Select a subset of points from the original image.
- 2) Use those points, with their colors, as centers in a new image of the same size.
- 3) Build the voronoi diagram in the new image, using the ctr colors from the original image.

The quality of the new image

depends on _____.

Planning

Point:

Color:

Center:

Centers:

Image:

Planning

Data flow:

1)

2)

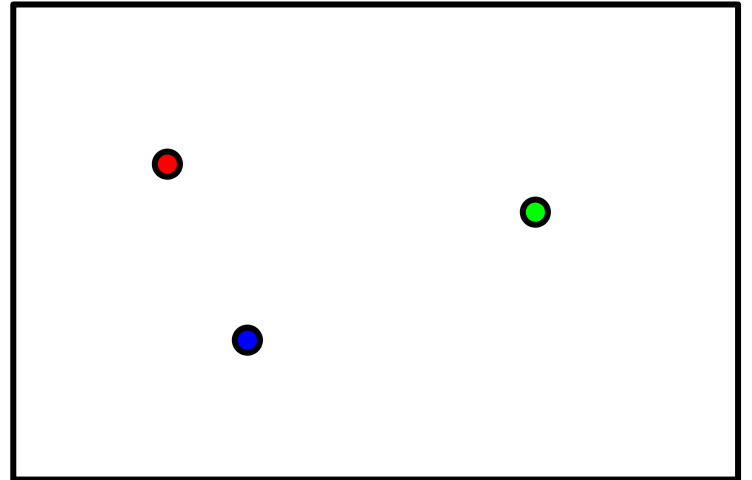
3)

4)

Demo and Analysis

How much work is done? Let n denote the size of the image, $n = \text{width} * \text{height}$

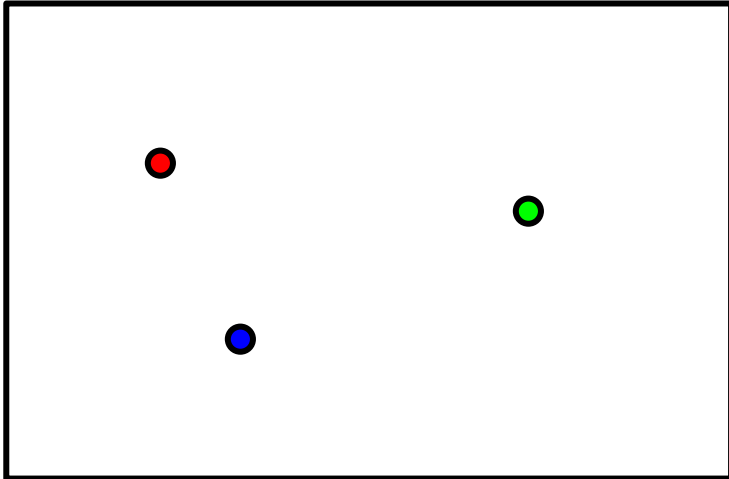
- 1) Read image:
- 2) Choose centers:
- 3) Build new image:
- 4) Write out new image:



Can we do better?

The running time of the original algorithm: _____

What would be better? _____



Orchestrate a fill from each center, growing out at the same rate.

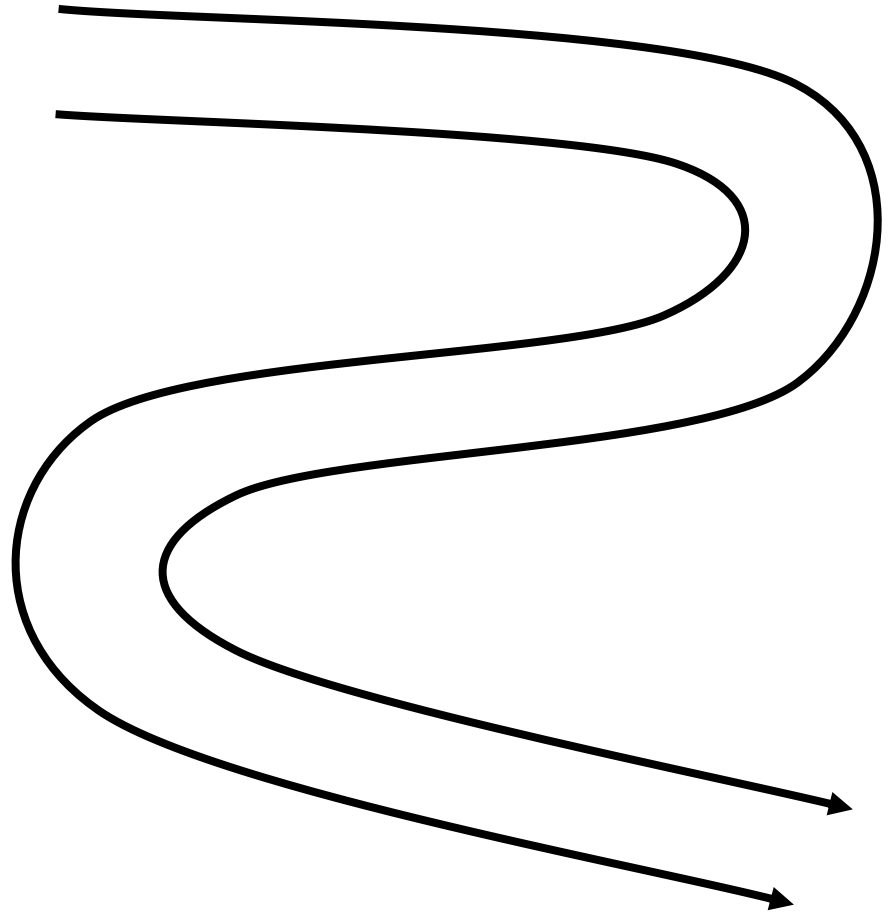
Each pixel is processed exactly once, not once per center as before.

This means we can have lots of centers!

Putting it together

00	10	20	30	40	50	60	70	80	90
01	11	21	31	41	51	61	71	81	91
02	12	22	32	42	52	62	72	82	92
03	13	23	33	43	53	63	73	83	93
04	14	24	34	44	54	64	74	84	94
05	15	25	35	45	55	65	75	85	95

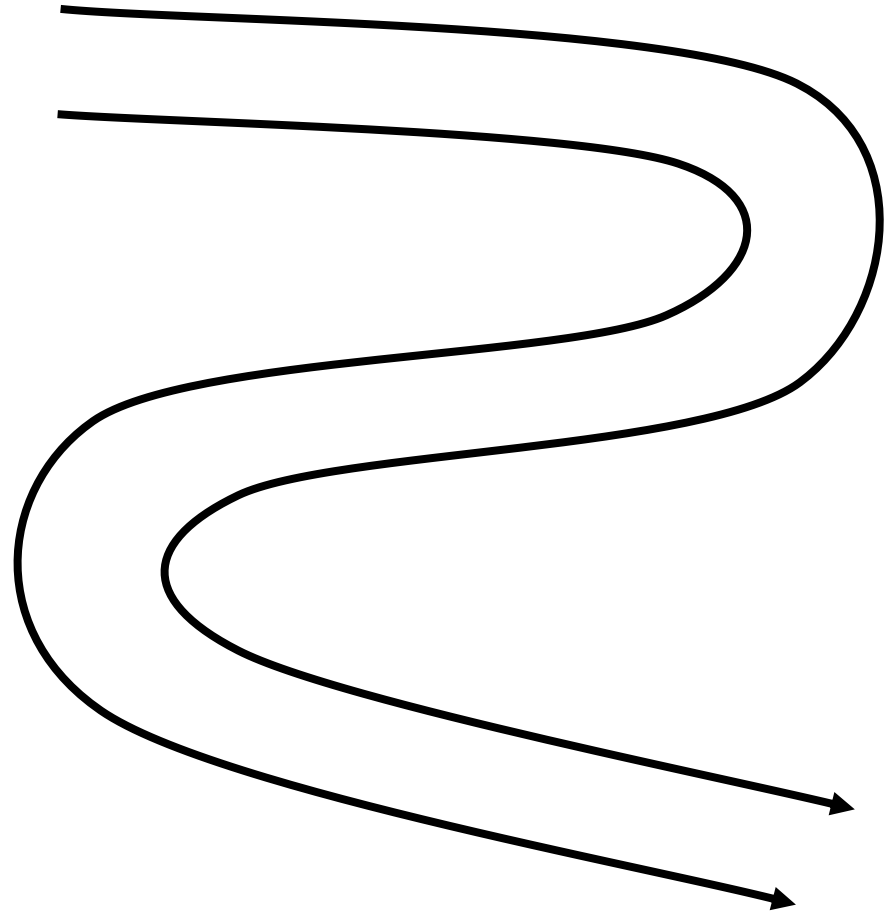
- 1) enqueue the center to start
- 2) while the queue is not empty:
 - a) $v = \text{dequeue}$
 - b) for each valid neighbor w , of v :
 - i) color w
 - ii) enqueue w



Putting it together

00	10	20	30	40	50	60	70	80	90
01	11	21	31	41	51	61	71	81	91
02	12	22	32	42	52	62	72	82	92
03	13	23	33	43	53	63	73	83	93
04	14	24	34	44	54	64	74	84	94
05	15	25	35	45	55	65	75	85	95

- 1) enqueue the center to start
- 2) while the queue is not empty:
 - a) $v = \text{dequeue}$
 - b) for each valid neighbor w , of v :
 - i) color w
 - ii) enqueue w



Designing the solution

- 1) enqueue the centers to start
- 2) while the queue is not empty:
 - a) $v = \text{dequeue}$
 - b) for each valid neighbor w , of v :
 - i) color w
 - ii) enqueue w

1. What info should we put on the queue?

1. We'll use `deque` as our queue (Python). What `deque` functions will we use?

1. Do `deques` have a way to check for empty?

1. What are the "neighbors" of pixel (x,y) ?

1. What would be an *invalid* neighbor?

-
-

Demo and Analysis NEW

How much work is done?

- 1) Read image: $w * h$
- 2) Choose centers: $c = density * w * h$
- 3) Build new image:
- 4) Write out new image: $w * h$





Looking ahead...



Graphs: A new model for representing images

00	10	20	30	40	50	60	70	80	90
01	11	21	31	41	51	61	71	81	91
02	12	22	32	42	52	62	72	82	92
03	13	23	33	43	53	63	73	83	93
04	14	24	34	44	54	64	74	84	94
05	15	25	35	45	55	65	75	85	95

A *Graph* is a collection of *vertices*, and *edges* between them. They're used as a general model for many problems.

In our images every _____ is a vertex, and every _____ is an edge. How many edges are there in the graph representing the image on the left?

Our fast algorithm for Voronoi Art mirrors a classic algorithm on graphs called Breadth First Search.

Breadth First Search

Breadth-first search (BFS) is an [algorithm](#) for traversing or searching [tree](#) or [graph](#) data structures. It starts at the [tree root](#) (or some arbitrary node of a graph, sometimes referred to as a 'search key'^[1]), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. (--Wikipedia)

Simplified description: