Week 8 Tutorial
Managing spatiotemporal data

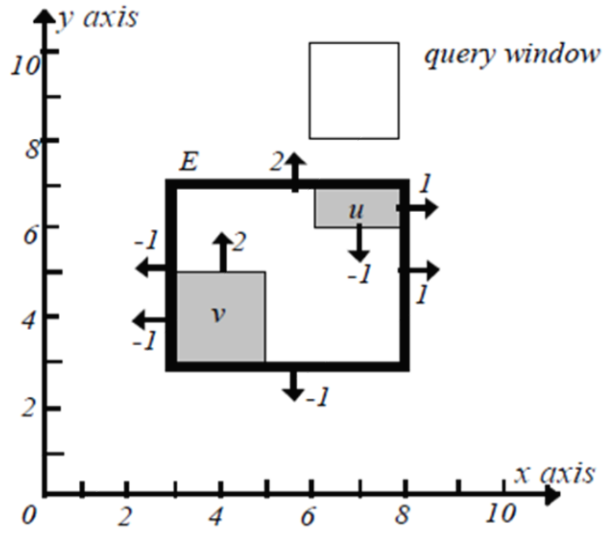
Tutor: Fengmei Jin

fengmei.jin@uq.edu.au

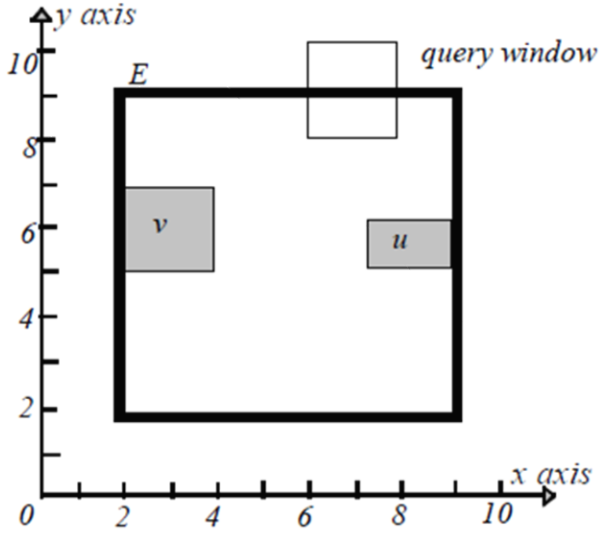
Question 1

- What is the benefit of using the TPR-Tree than the other R-Tree-based structures to index the moving objects?

- Time-Parameterized R-tree
- Store locations and MBRs as functions of time
  - L(t) = L(t0) + V(t) MBR(t) = MBR(t0) + V(t)
- VBR: V for velocity
  - An MBR grows with time, and can be estimated
  - Insertion of objects needs to minimize VBR



(a) The boundaries at current time 0



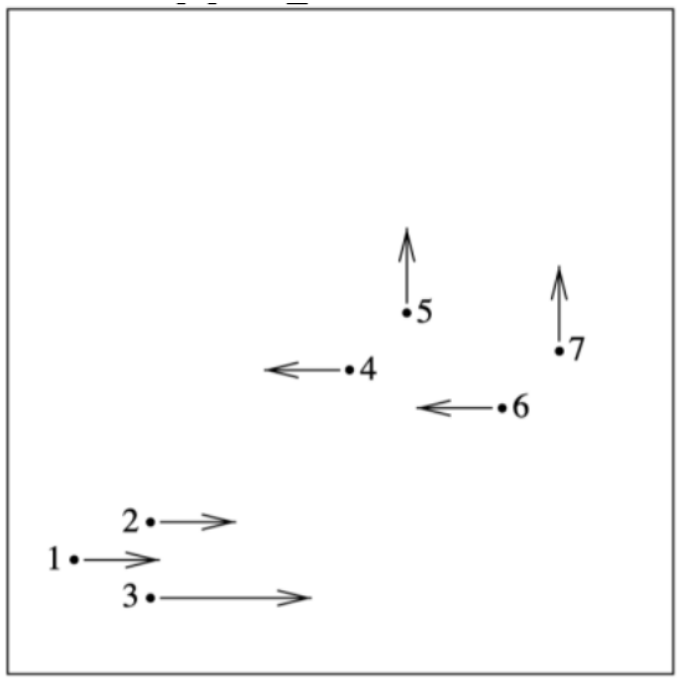
(b) The boundaries at future time 1

Question 1 – answer

- TPR-Tree is more suitable for queries with a time-interval and has smaller overlapping ratio by considering the current location and the future location.

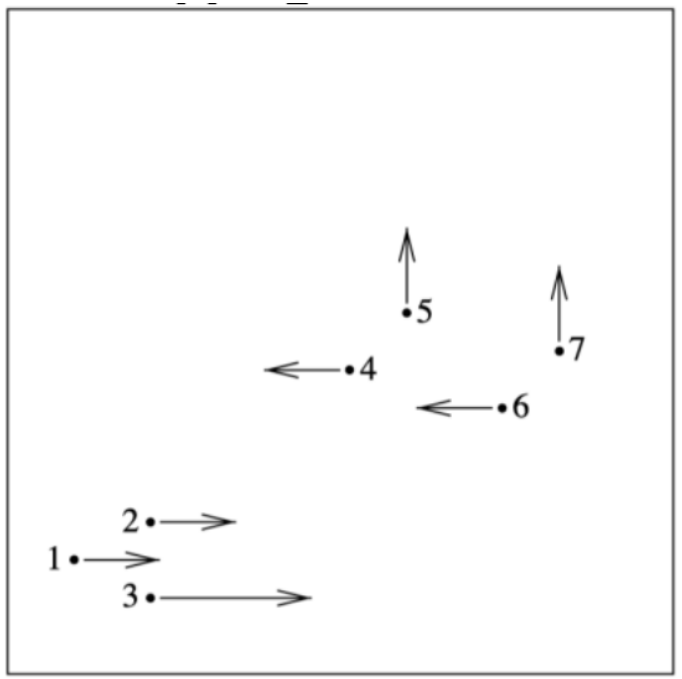
Question 1 – example

- Suppose we have seven points with their current locations at time 0 shown in the figure, and they are moving directions are labelled as the arrows (with different speeds).

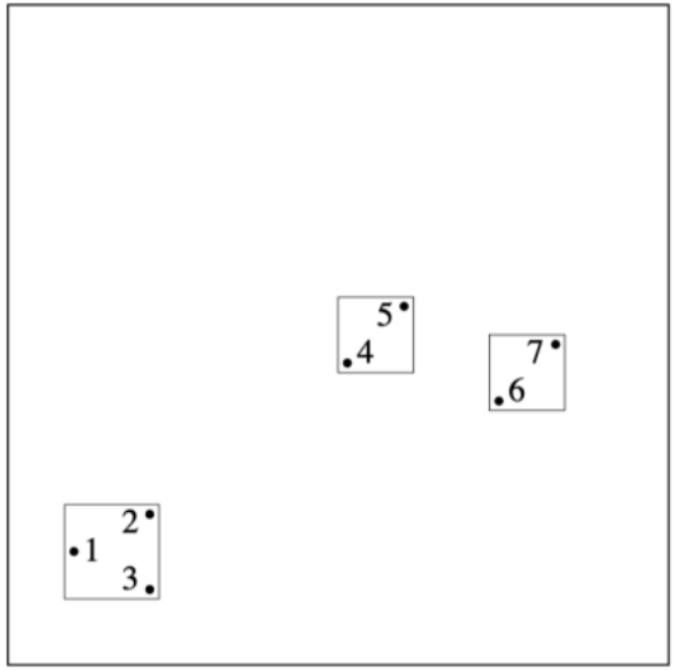


Question 1 – example

- Suppose we have seven points with their current locations at time 0 shown in the figure, and they are moving directions are labelled as the arrows (with different speeds).

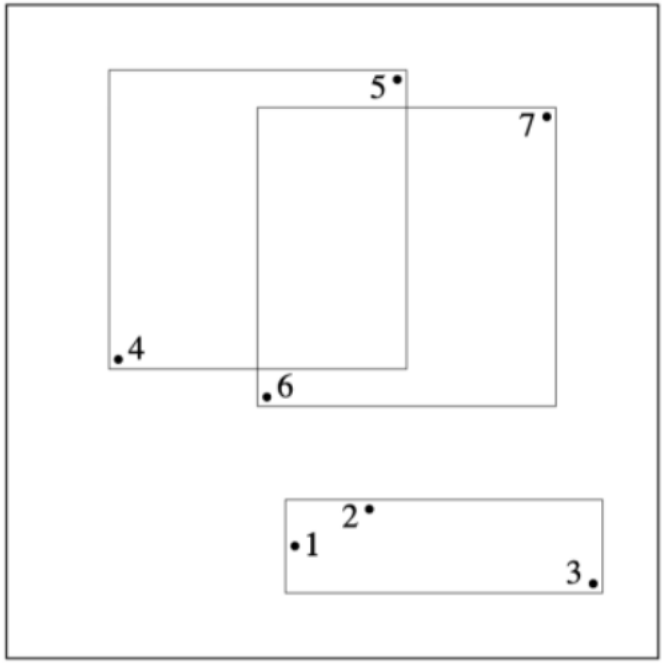


- If only considering the locations of the current timestamp, the optimal MBR should be like this:



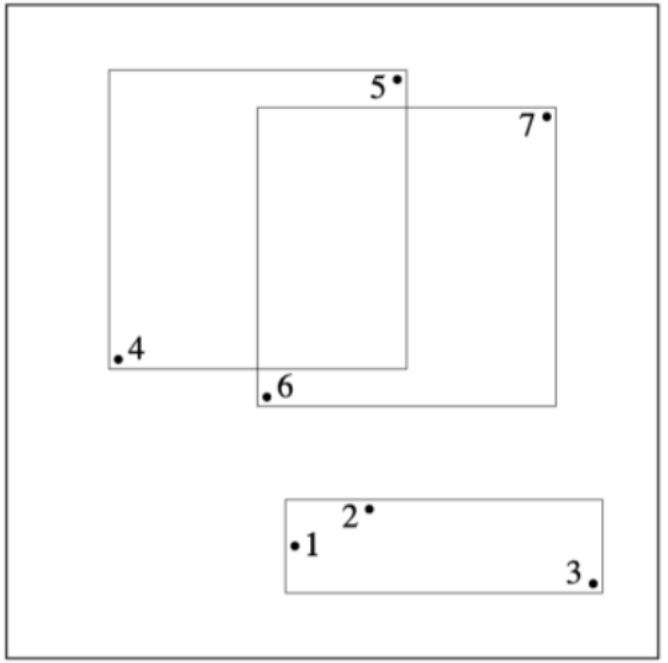
Question 1 – example

- However, if we use these three MBRs to approximate the points in the next timestamp, they will have a very large overlapping:

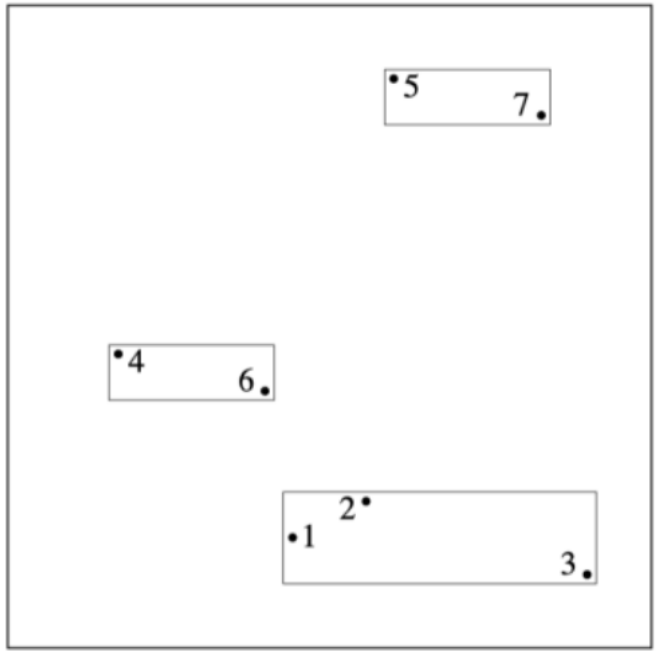


Question 1 – example

- However, if we use these three MBRs to approximate the points in the next timestamp, they will have a very large overlapping:



- Therefore, if we consider both the locations and the prediction of the future locations (with moving direction and velocity), we could have better choices, like put 5 and 7 together, put 4 and 6 together.



Dynamic programming

- Definition: Break up a problem into a series of overlapping sub-problems, and build up solutions to larger and larger sub-problems.

- Steps for solving DP problems:
  1. Define subproblems
     - the subproblems has same or similar form with the original problem, but they have smaller data size. We solve the original problem by solving all the subproblems.
  2. Write down the recurrence that relates subproblems
  3. Recognize and solve the base case.

1-dimensional DP example

- Problem: given n, find the number of different ways to write n as the sum of 1, 3, 4

- Example: for n=5, the answer is 6

  5 = 1 + 1 + 1 + 1 + 1
  = 1 + 3 + 1
  = 3 + 1 + 1
  = 4 + 1
  = 1 + 1 + 3
  = 1 + 4

1-dimensional DP example

- Define subproblems:
  - Let D_n be the number of ways to write n as the sum of 1, 3, 4.
- Find the recurrence:
  - Consider one possible solution $n = x_1 + x_2 + \cdots + x_m$
  - If $x_m = 1$, the rest of the terms must sum to $n - 1$. Thus, the number of sums that end with $x_m = 1$ is equal to D_{n-1}.
  - Take other cases into account ($x_m = 3, x_m = 4$)
- Recurrence is then: $D_n = D_{n-1} + D_{n-3} + D_{n-4}$
- Solve the base cases:
  - $D_0 = 1$
  - $D_n = 0$ for all negative n
  - Alternatively, can set: $D_0 = D_1 = D_2 = 1$, and $D_3 = 2$

Question 2

- LCSS (Longest Common Sub-Sequence) is a way to compare the similarity between two sequences.
- Definition: Given two sequences, LCSS finds the length of the longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous (if we require the contiguous, it is the longest common sub-string).
  - For example, “abc”, “abg”, “bdf”, “aeg”, “acefg” are all sub-sequences of “abcdefg”.
- Such a meter can also be used in to evaluate how similar two trajectories are. Given two points p and q, a distance threshold ϵ, we regard p and q as the same point if $dist(p, q) \leq \epsilon$.
- Please discuss the dynamic programming technique that can be used to compute LCSS.

Question 2 – LCSS

- We use two sequence to illustrate how LCSS can be computed:
  - S = AGGCAB
  - T = GXCXAYB
- Define subproblem:
  - we will look at the LCSS of a prefix of S and a prefix of T, running over all pairs of prefixes
- Find recurrence:
  - Suppose we use $LCSS[i, j]$ to denote the result of LCSS of sub-sequence $S[1,\ldots, i]$ and $T[1,\ldots, j]$ (e.g., if i=2, and j=3, we are comparing AG and GXC).
  - And suppose we already know all the results of shorter prefix pairs ($LCSS[i-1, j]$, $LCSS[i, j-1]$, $LCSS[i-1, j-1]$).
  - Solve the $LCSS[i, j]$ with those smaller and already solved problems' results

Question 2 – LCSS

S = AGGCAB
T = GXCXAYB

There are two cases:

1) When $S[i] = T[j]$, then we directly increase the already matched value of $LCSS[i-1, j-1]$ by 1:
   - $LCSS[i, j] = LCSS[i-1, j-1] + 1$
   - Top left neighbour value plus one

Question 2 – LCSS

S = AGGCAB
T = GXCXAYB

There are two cases:

1) When $S[i] = T[j]$, then we directly increase the already matched value of $LCSS[i-1, j-1]$ by 1:
   - $LCSS[i, j] = LCSS[i-1, j-1] + 1$
   - The upper-left neighbour value plus one
2) When $S[i] \neq T[j]$, then the result has to ignore one of $S[i]$ or $T[j]$, so we have:
   - $LCSS[i, j] = \max(LCSS[i-1, j], LCSS[i, j-1])$
   - The maximum value of the upper and left neighbour cells

Question 2 – LCSS

S = AGGCAB
T = GXCXAYB

- Therefore, we need only two loops to compute these LCSS. Initially, the first row and the first column are set to 0.

	ϕ	G	X	C	X	A	Y	B
ϕ	0	0	0	0	0	0	0	0
A	0							
G	0							
G	0							
C	0							
A	0							
B	0							

Question 2 – LCSS

S = AGGCAB
T = GXCXAYB

- Then we fill this table row by row with the previous two rules:

	ϕ	G	X	C	X	A	Y	B
ϕ	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	1	1
G	0	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1
C	0	1	1	2	2	2	2	2
A	0	1	1	2	2	3	3	3
B	0	1	1	2	2	3	3	4

Question 3

- Edit Distance is another widely used string similarity measurement that can be used to compare two trajectories.
- To convert from one string to another, for each character, we have four choices: let it be, insert another one, delete it, and replace it.
  - The insertion, deletion, and substitution are call edit operations, and the minimum number of edit operations it takes to transform one string to another is call the edit distance between them.
- Similarly, in the trajectory distance, we can regard two points within a distance threshold as the same one. In this way, the trajectory similarity is converted to the string similarity.
- Please use the example in Question 1 to show what is the edit distance between S and T.

Question 3 – ED

S = AGGCAB
T = GXCXAYB

- For each cell $[i,j]$ in the table, it means the minimum number of edit operations it takes to convert string $S[1,\ldots,i]$ to $T[1,\ldots,j]$.

- Define subproblem: Same as Q2

- Find recurrence:
  - The edit operations have the follow actual effects on the value computation. Suppose we are now at $[i,j]$, we have three choices to compute its edit distance, including insert, delete, and replace.

Question 3 – ED

	ϕ	G	X	C	X	A	Y	B
ϕ	0	1	2	3	4	5	6	7
A								
G								
G								
C								
A								
B								

1) We can use the edit distance between $S[1, i]$ and $T[1, j-1]$ (the left neighbour), so we only need insert the value of $T[j]$ at $S[i]$ to make $S[1, i]$ and $T[1, j]$ the same. Thus, the edit distance at $ED[i, j] = ED[i, j-1] + 1$.

- E.g., in the first row, from the empty string ϕ to G, we must insert G, so the edit distance is 1;
- From G to GX, we insert X, so the edit distance is 1+1 …
- Eventually, from ϕ to GXCXAYB, we insert 7 characters in total.

- Remark: Whenever we use the edit distance from the left neighbour, it means we do the insertion, so the edit distance plus 1.

Question 3 – ED

	ϕ	G	X	C	X	A	Y	B
ϕ	0	1	2	3	4	5	6	7
A	1							
G	2							
G	3							
C	4							
A	5							
B	6							

2) We can also use the edit distance between $S[1, i-1]$ and $T[1, j]$ (the upper neighbour), so we only need to delete the value of $S[i]$ to make $S[1, i]$ and $T[1, j]$ the same. Therefore, the edit distance at $ED[i, j] = ED[i-1, j] + 1$.

- E.g., in the first column, it means the edit distance from A to ϕ, so we delete A, and its edit distance is 1;
- Then if we convert AG to ϕ, the edit distance is 2…
- Finally, the edit distance from AGGCAB to ϕ is 6, as we delete all characters.

- Remark: Whenever we use the edit distance from the upper neighbor, it means we do the deletion, so the distance plus 1.

Question 3 – ED

	ϕ	G	X	C	X	A	Y	B
ϕ	0	1	2	3	4	5	6	7
A	1	1						
G	2		2					
G	3			3				
C	4				4			
A	5					4		
B	6						5	

3) We can also use the edit distance between $S[1, i-1]$ and $T[1, j-1]$ (the upper-left neighbour). This time we have two cases to consider:

- If $S[i] = T[j]$, we do not need any editing, so $ED[i, j] = ED[i-1, j-1]$
- If $S[i] \neq T[j]$, replace $S[i]$ with $T[j]$, so $ED[i, j] = ED[i-1, j-1] + 1$

Question 3 – ED

	ϕ	G	X	C	X	A	Y	B
ϕ	0	1	2	3	4	5	6	7
A	1	1						
G	2		2					
G	3			3				
C	4				4			
A	5					4		
B	6						5	

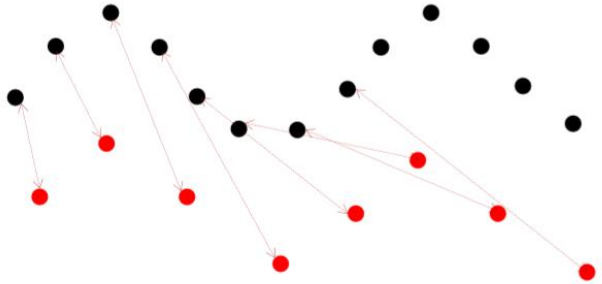
- Conclusion: Now for each $ED[i,j]$, we have the following three choices, and we select the minimum of these three values as the edit distance:
  - From the left neighbour, $ED[i,j] = ED[i,j-1] + 1$
  - From the upper neighbour, $ED[i,j] = ED[i-1,j] + 1$
  - From the upper-left neighbour:
    - $ED[i,j] = ED[i-1,j-1]$, if $S[i] = T[j]$
    - $ED[i,j] = ED[i-1,j-1] + 1$, if $S[i] \neq T[j]$
- In summary, $ED[i,j] = \min(ED[i,j-1] + 1, ED[i-1,j] + 1, ED[i-1,j-1] + 0/1)$
  - only three computations and taking the minimum is enough.

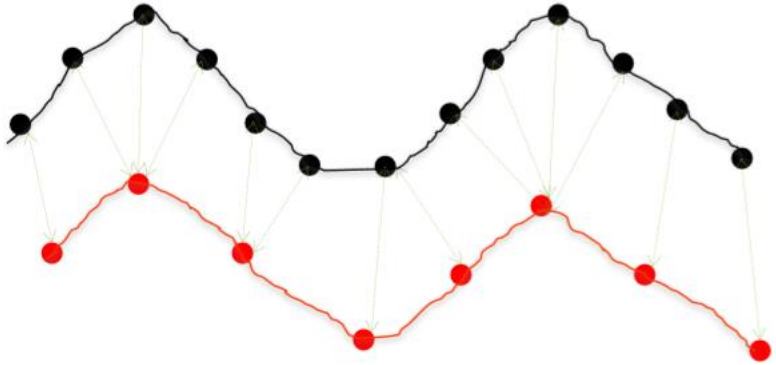
Question 3 – ED

- The edit distance between AGGCAB and GXCXAYB is 4.
- The actual edit steps are:
  - delete the first A;
  - replace the second G with X;
  - insert X between C and A;
  - insert Y between A and B.

	ϕ	G	X	C	X	A	Y	B
ϕ	0	1	2	3	4	5	6	7
A	1	1	2	3	4	4	5	6
G	2	1	2	3	4	5	5	6
G	3	2	2	3	4	5	6	6
C	4	3	3	2	3	4	5	6
A	5	4	4	3	3	3	4	5
B	6	5	5	4	4	4	4	4

Question 4





- How to compute the DTW with the dynamic programming?

- Unlike LCSS and ED, the DTW is not about how many same characters or how many operations, but the actual distance between two trajectories.
- **Goal:** minimize the aggregate distance between matched points
- **1-to-many mapping:** one point in one sequence can be mapped to multiple points in another sequence

Question 4 – DTW

- Still, using the points are not handy to illustrate the example, try to use two numbers here:
  - $S = \{1,2,3,5,5,5,6\}$
  - $T = \{1,1,2,2,3,5\}$
- Suppose the distance between two numbers are just their difference.
  - When working in the 2D point, we can simply replace the distance with Euclidean distance or Network distance.

Question 4 – DTW

- Like the dynamic programming in LCSS and ED, for each cell $[i, j]$, we also have three sources:
  - the left neighbour $DTW[i-1, j]$
  - the upper neighbour $DTW[i, j-1]$
  - the upper-left neighbour $DTW[i-1, j-1]$
- All we need to do is take the minimum of them and add the distance between $S[i]$ and $T[j]$.

	0	1	1	2	2	3	5
0	0	INF	INF	INF	INF	INF	INF
1	INF	0	0	1	2	4	8
2	INF	1	1	0	0	1	4
3	INF	3	3	1	1	0	2
5	INF	7	7	4	4	2	0
5	INF	11	11	7	7	4	0
5	INF	15	15	10	10	6	0
6	INF	20	20	14	14	9	1