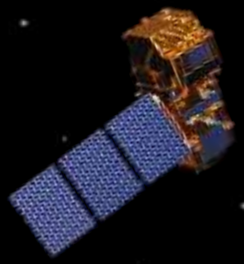
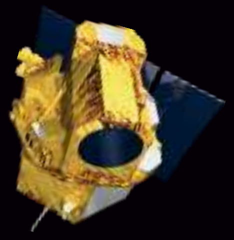


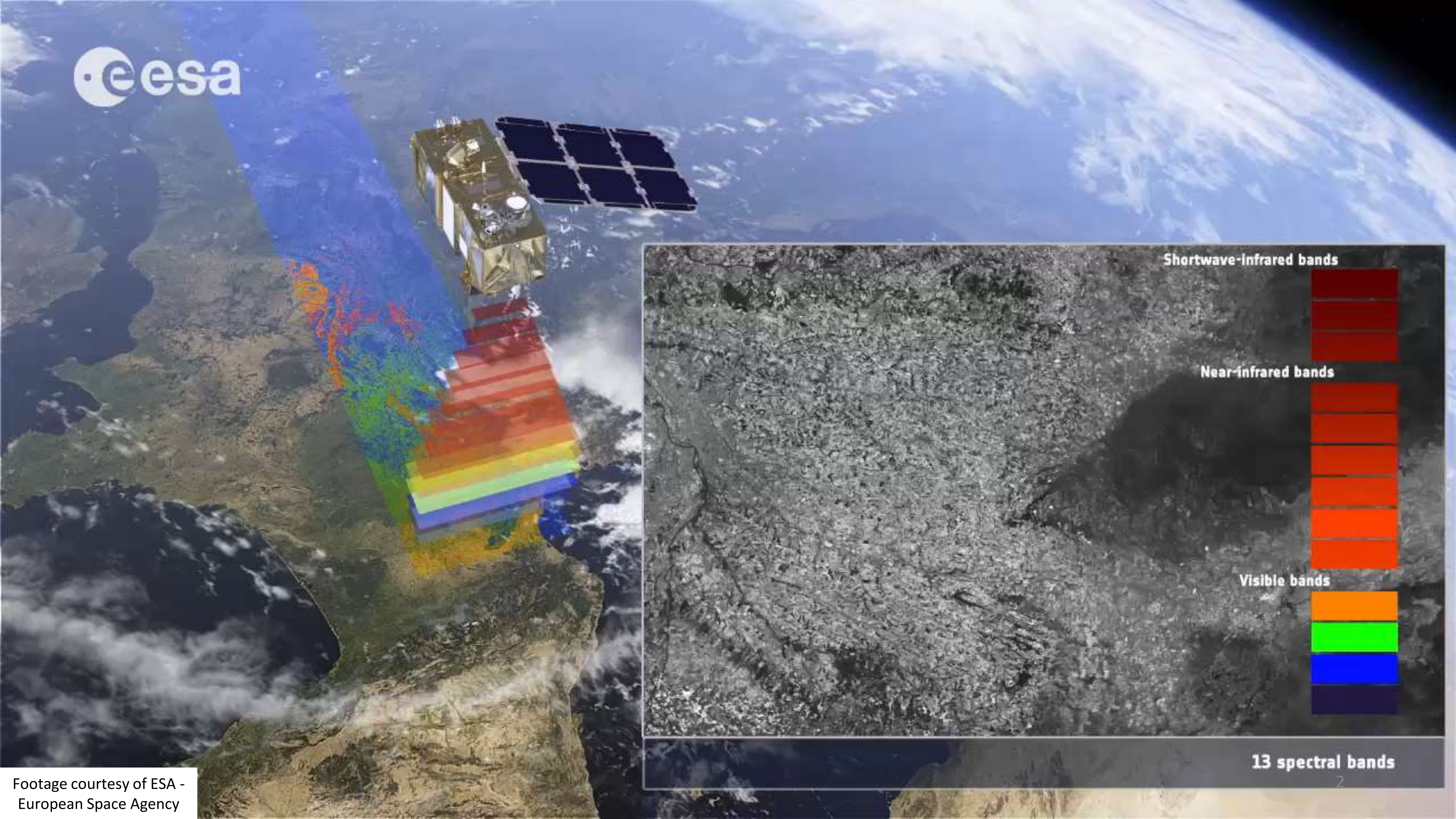
Indexing and Classifying Gigabytes of Time Series under Time Warping

C.W. Tan **G.I. Webb** **F. Petitjean**



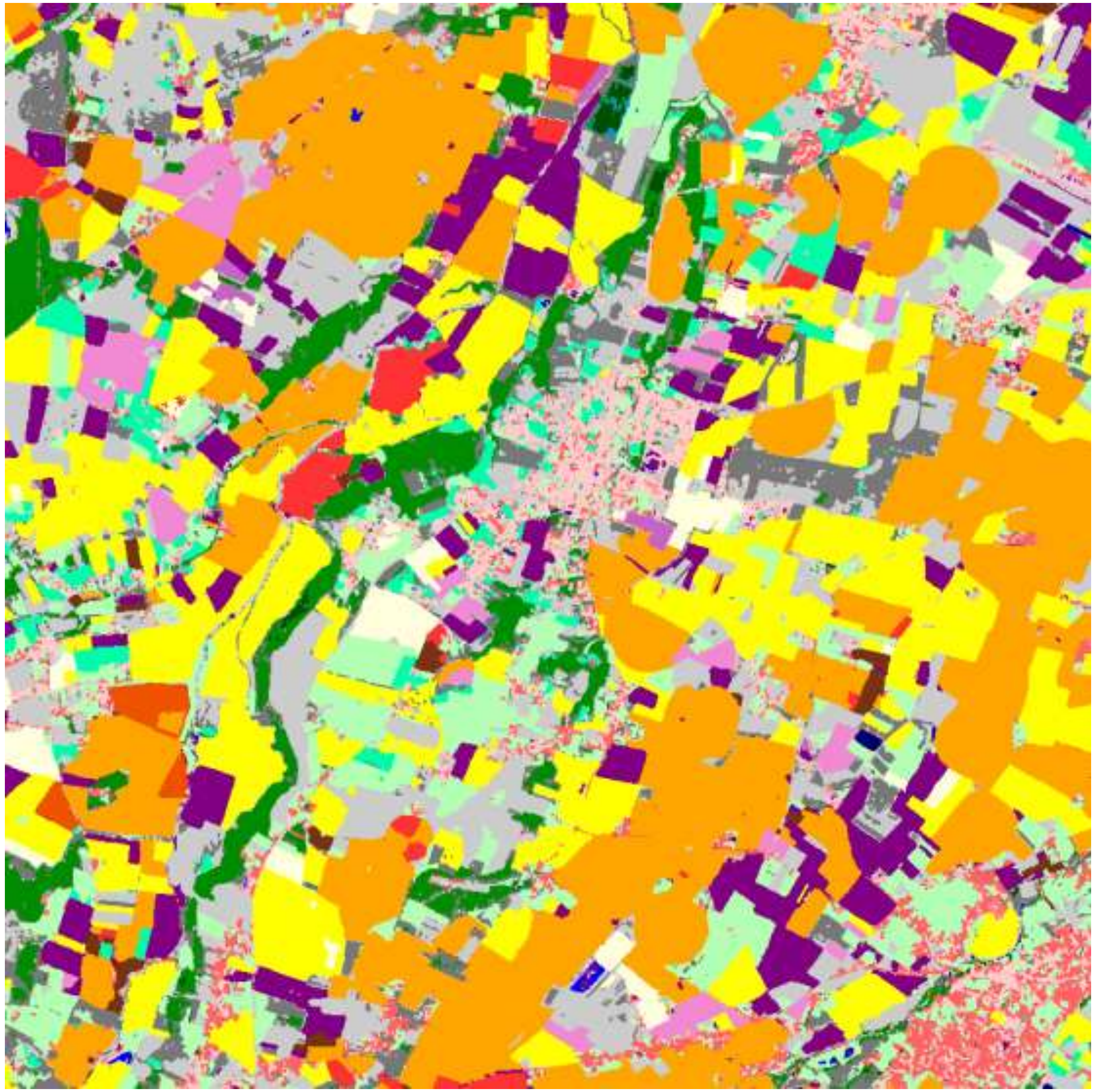
2017 SIAM International Conference on DATA MINING

27 April 2017



Footage courtesy of ESA - European Space Agency

Temporal Land-Cover Maps



LEGEND OF THE MAPS

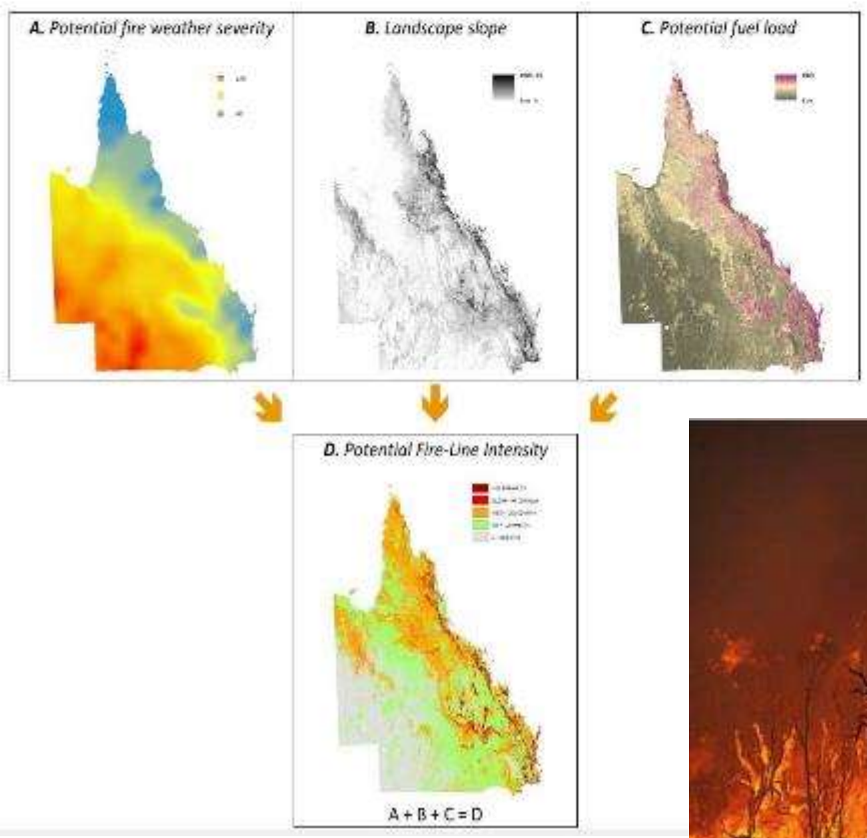
Color	Class
	corn
	corn for silage
	non-irrigated corn
	wheat
	sunflower
	sorghum
	sorghum II
	soybean
	barley
	pea
	rape
	broad-leaved tree
	conifer
	poplar tree
	eucalyptus
	water
	lake
	gravel pit
	meadow
	temporary meadow
	fallow land
	wild land
	high density housing surface
	specific urban surface
	low density housing surface
	mineral surface

What can we do with it?

- **Yield forecast**

What can we do with it?

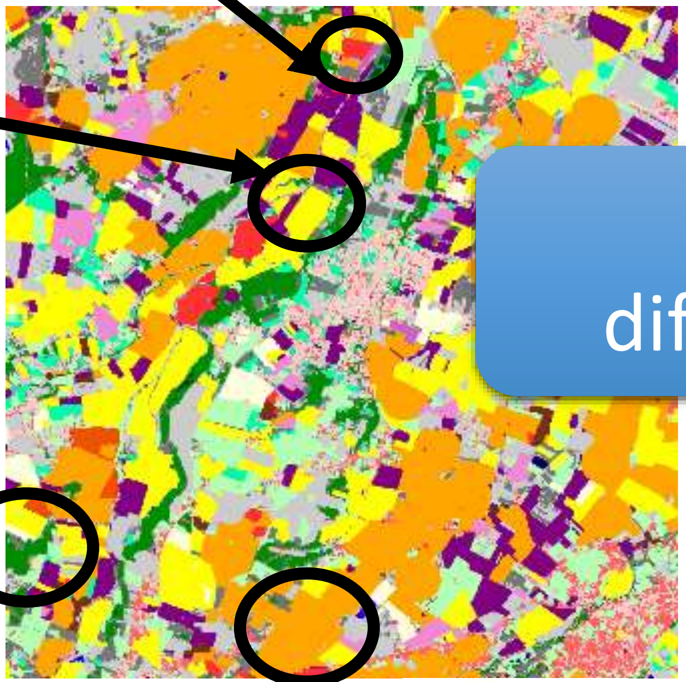
- Yield forecast
- **Fire spread model**



What can we do with it?

- **Yield forecast**
- **Fire spread model**
- **City pollution absorption models**
- and more…

One Image is not enough!



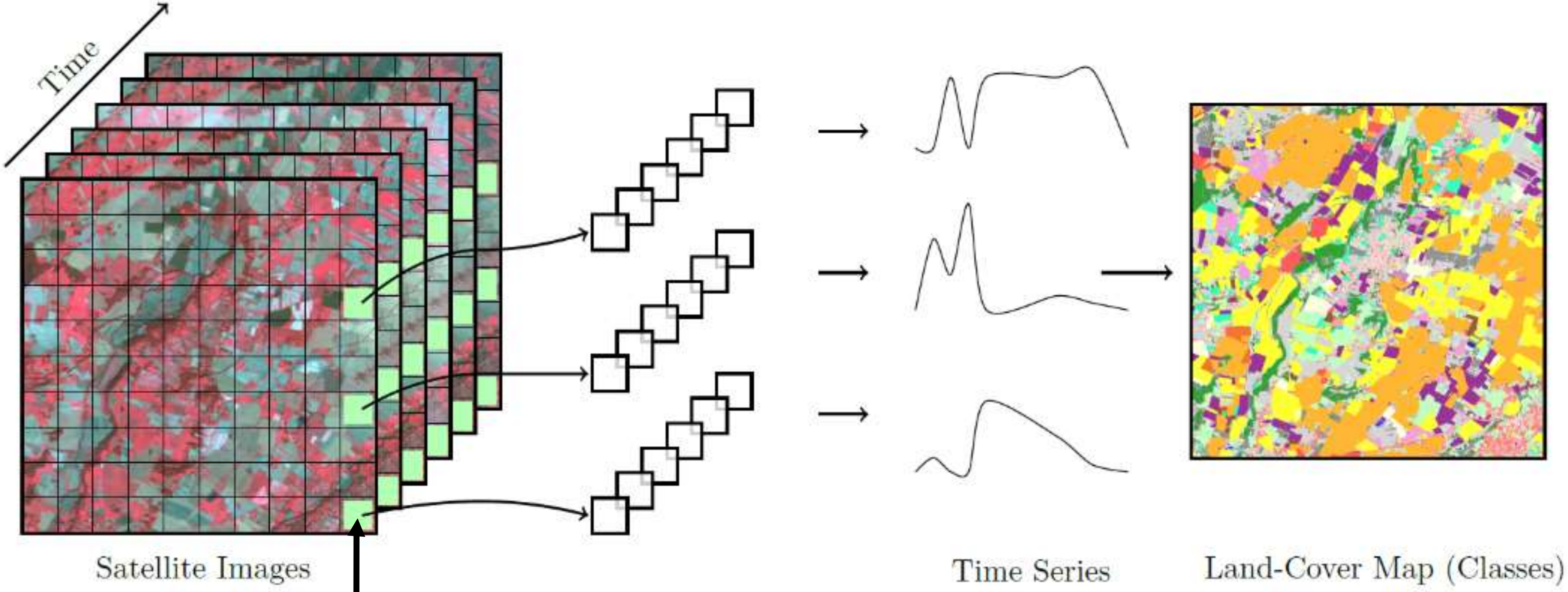
LEGEND OF THE MAPS

Color	Class
	corn
	corn for silage
	eucalyptus
	water
	lake
	gravel pit
	meadow
	temporary meadow
	fallow land
	wild land
	high density housing surface
	specific urban surface
	low density housing surface
	mineral surface

Impossible to differentiate them!

What's possible? → Temporal Evolution

Satellite Image Time Series (SITS) Analysis



Every pixel represents a geographic area (Lat, Lon) on Earth

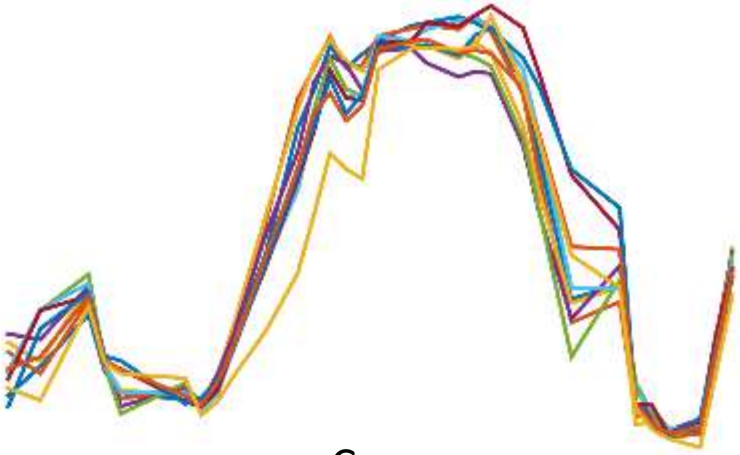
Petitjean, F., Kurtz, C., Passat, N., & Gançarski, P. (2012). Spatio-temporal reasoning for the classification of satellite image time series. Pattern Recognition Letters, 33(13), 1805-1815.

How to do this?

- Time series classification
- State-of-the-art, Nearest Neighbor coupled with Dynamic Time Warping (NN-DTW) [1]
  - Many phenomena of interest – vegetation cycles, have periodic behavior which can be modulated by weather artifacts. [2]
  - Too short for the Bag-of-word-type approaches to perform best
    - Length of 46 – 52
    - Less features in the series
    - BOSS-VS [3] achieved around 40% error rate, NN-DTW achieved 16%

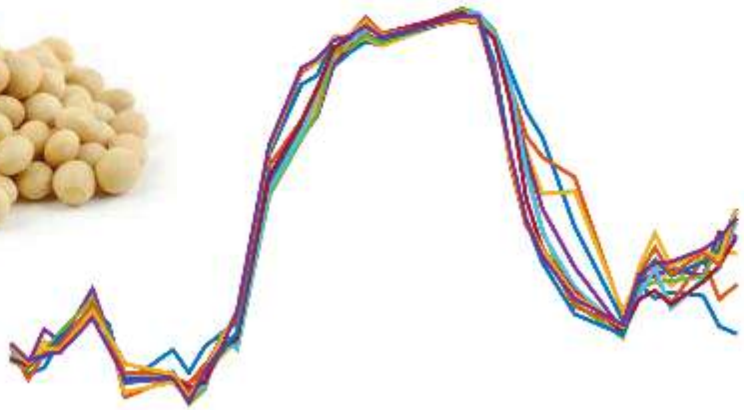
[1] Bagnall, A., & Lines, J. (2014). An experimental evaluation of nearest neighbour time series classification. technical report# CMP-C14-01. *Department of Computing Sciences*, *University of East Anglia*, Tech. Rep.
[2] Petitjean, F., Inglada, J., & Gançarski, P. (2012). Satellite image time series analysis under time warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8), 3081-3095.
[3] Schäfer, P. (2016). Scalable time series classification. *Data Mining and Knowledge Discovery*, *30*(5), 1273-1298.

Example series for different crops



Corn

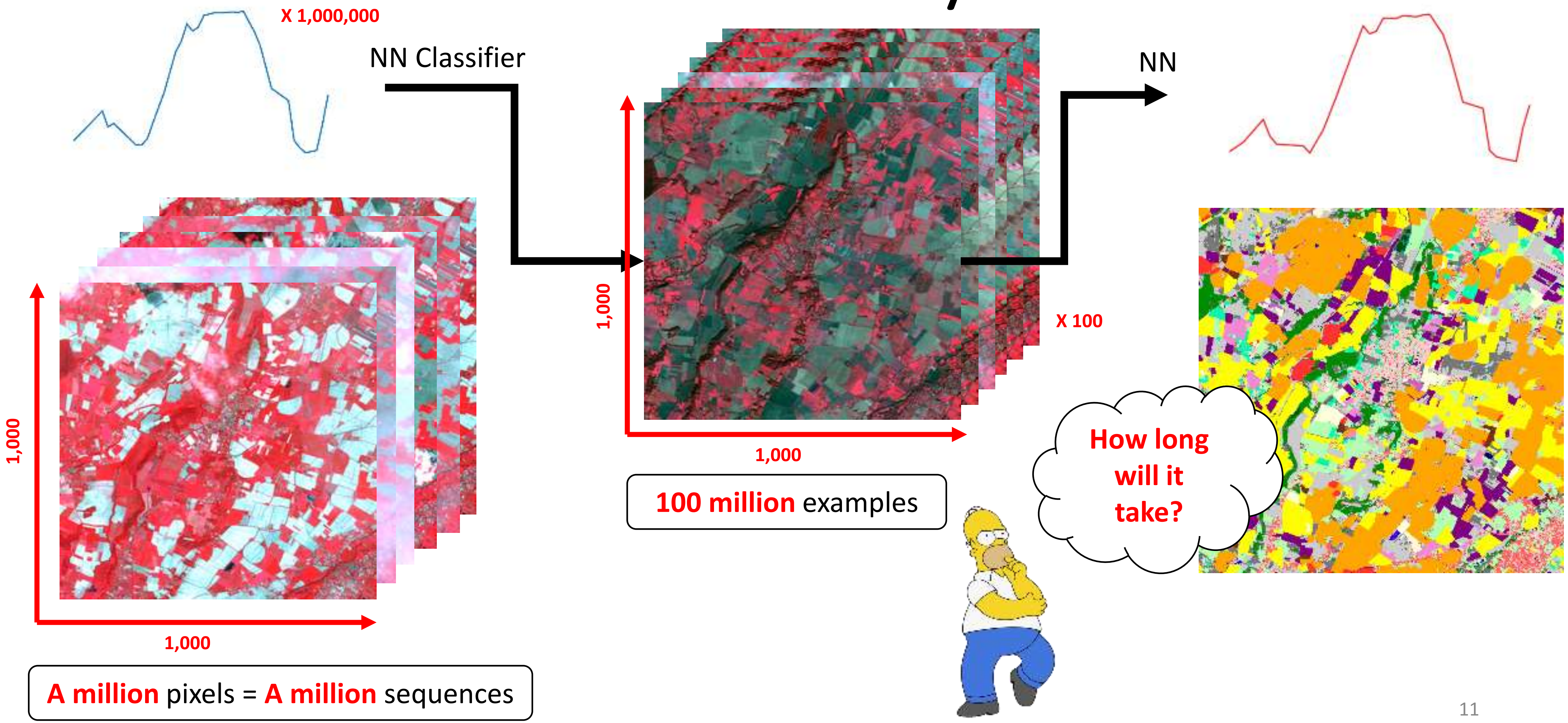
Wheat

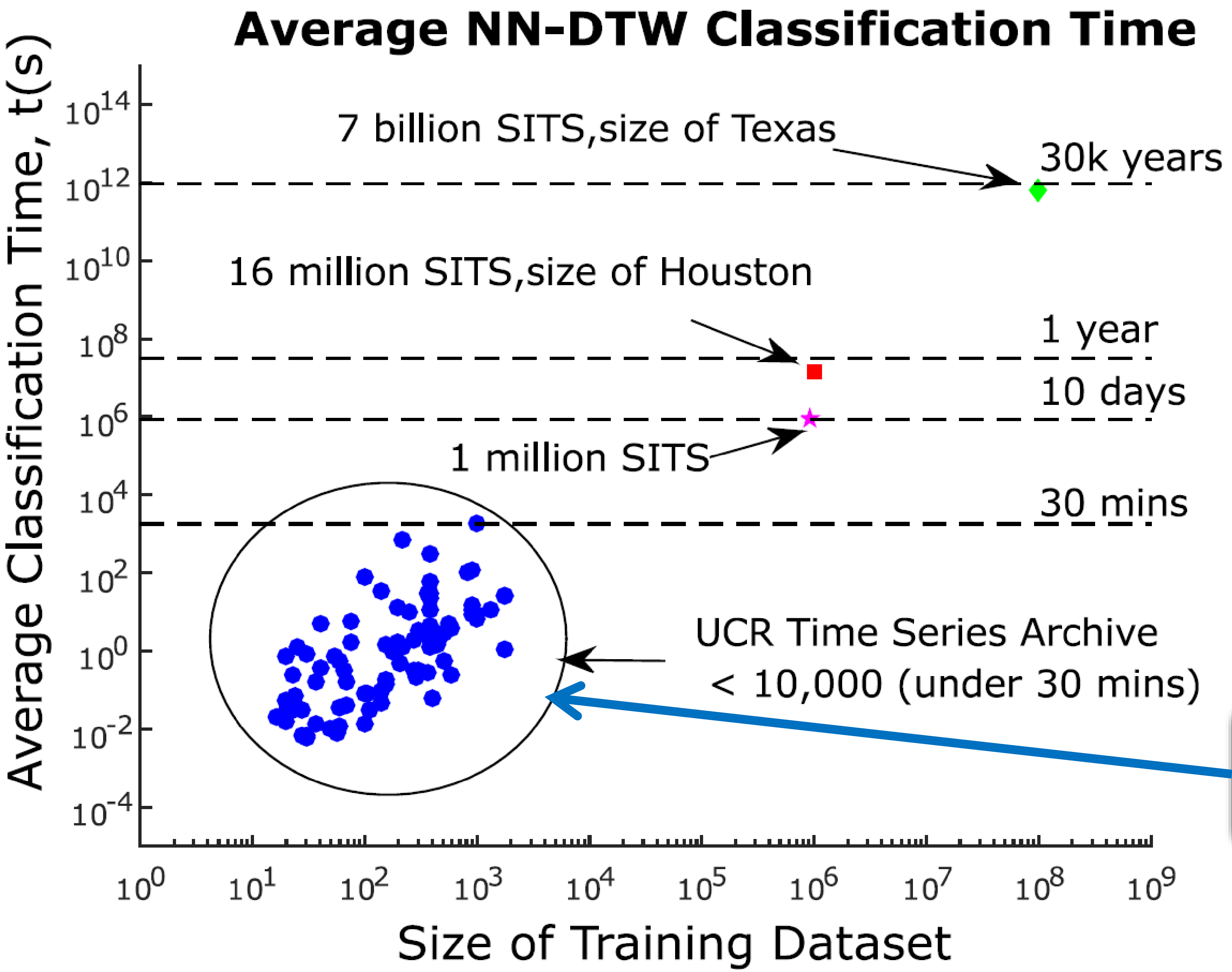


Soybean

Broad-Leaved Tree

Traditionally

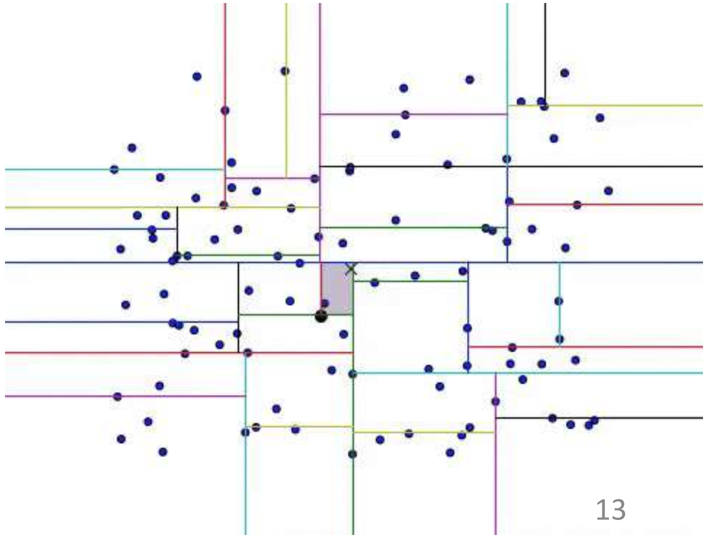




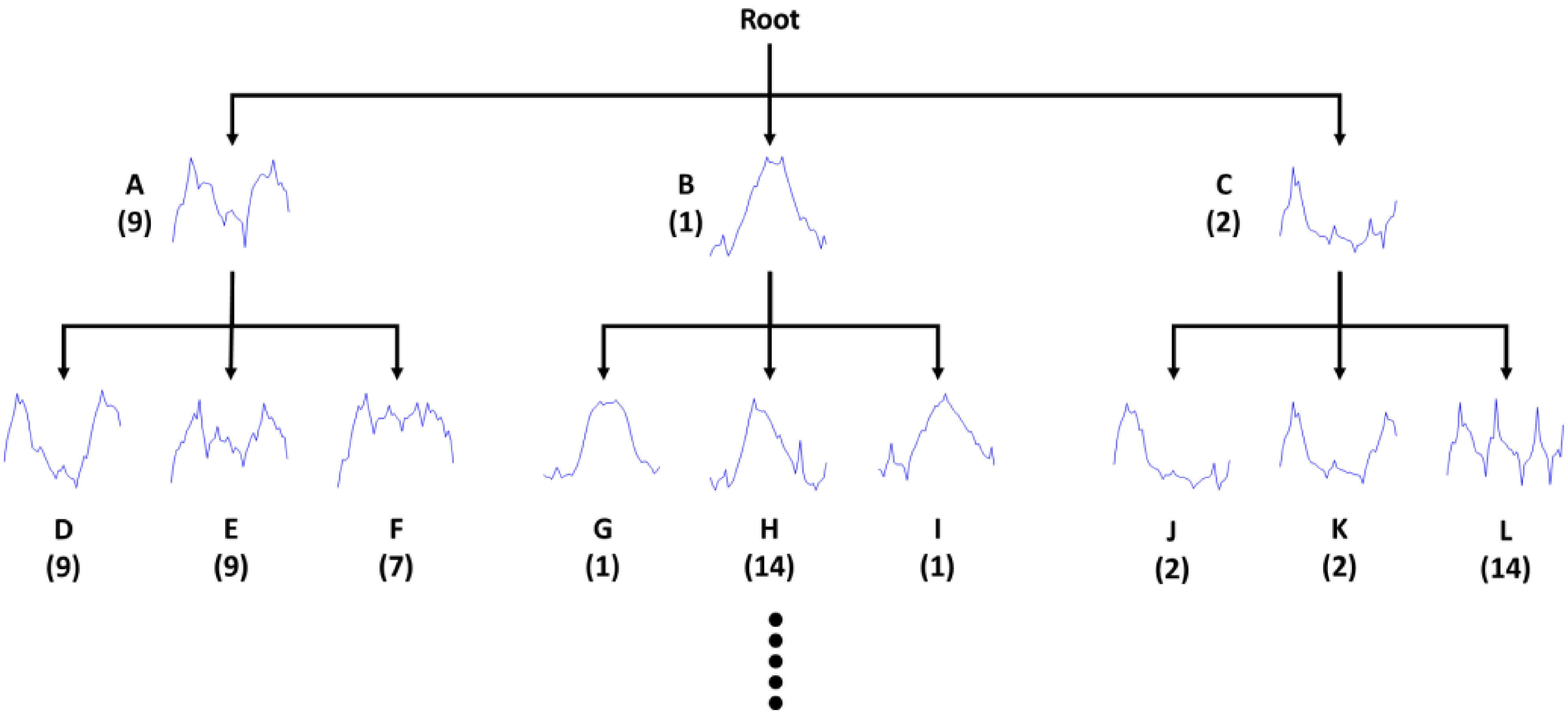
Most research in time series classification

Problem Statement

- **Anytime Time Series Classification**
  - Classify a query at any given time with high accuracy
  - Without constraints on computational resources at training time
- In Nearest Neighbor classification
  - Find the nearest neighbor much faster than full linear scan
  - Traditional techniques
    - Build an indexing structure in Euclidean Space
    - k-d tree, R tree, LSH …
    - Does not work with DTW

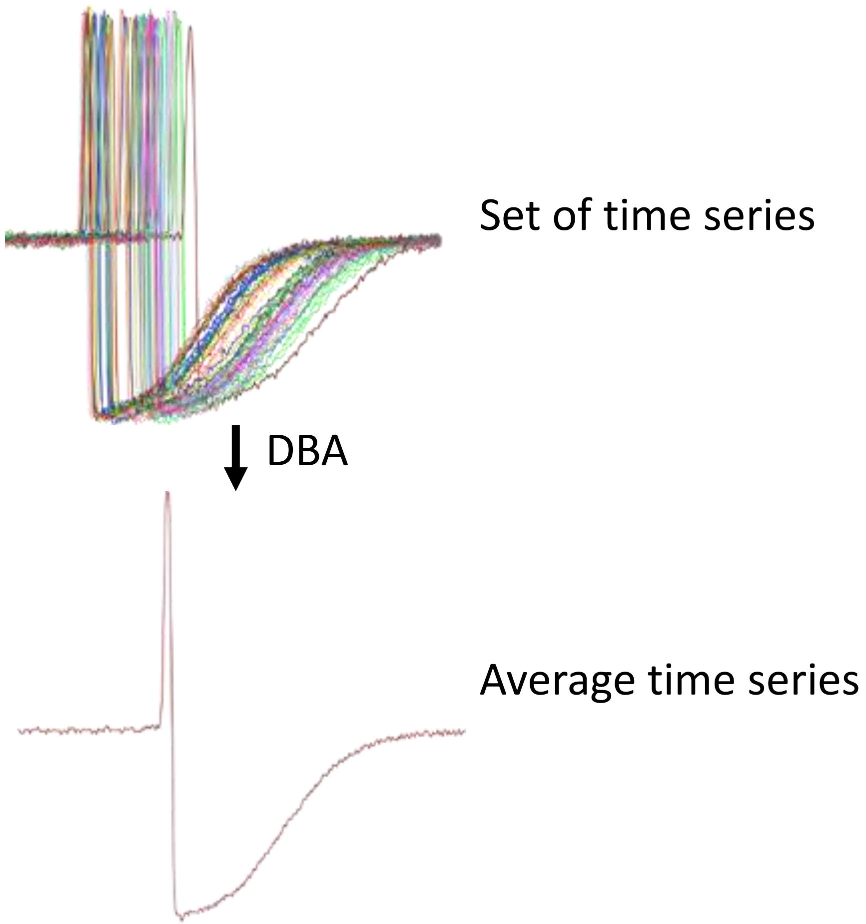


Indexing with Hierarchical Clusters



Time Series Indexing

- Hierarchical K-means indexing structure
  - Uses a priority search to speedup the process [1]
- Leverage off a recent work on DTW averaging
  - DTW Barycenter Averaging (DBA) [2, 3]
  - [2] shows that K-means and DBA allows faster and more accurate classification



[1] Muja, M., & Lowe, D. G. (2014). Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(11), 2227-2240.
[2] Petitjean, F., Forestier, G., Webb, G. I., Nicholson, A. E., Chen, Y., & Keogh, E. (2014, December). Dynamic time warping averaging of time series allows faster and more accurate classification. In *Data Mining (ICDM), 2014 IEEE International Conference on* (pp. 470-479). IEEE.
[3] Petitjean, F., Ketterlin, A., & Gançarski, P. (2011). A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, *44*(3), 678-693.

Time Series Indexing

- At testing time

Traverse to first leaf

SearchTree(*T*, *Q*, *K*)
    PQ, *Res* = empty priority queues
    Traverse(*T, Q, PQ, Res*)
    while (within contract **and** *PQ* not empty) **do**
        nextBranch = *PQ*.*pop*()
        Traverse(*nextBranch*, *Q*, *PQ, Res*)
    end while

    return *Res*.*pop*(*k*)

Unexplored branches to here

Traverse(*T*, *Q*, *PQ*, *Res*)
    if (*T* is leaf) **then**
        Res.addAll(T.data) with distances to *Q*
    else
        C = *T.child* nearest to *Q*
        PQ.addAll(*T.child* except *C*) with
            distances to *Q*
        Traverse(*C*, *Q*, *PQ*, *Res*)
    end if

Time Series Indexing

- At testing time

Traverse to first leaf

Unexplored branches to here

SearchTree(*T*, *Q*, *K*)
    PQ, *Res* = empty priority queues
    Traverse(*T*, *Q*, *PQ*, *Res*)
    while (not stop **and** *PQ* not empty) **do**
        nextBranch = *PQ.pop*()
        Traverse(*nextBranch*, *Q*, *PQ*, *Res*)
    end while

    return *Res.pop*(*k*)

Traverse(*T*, *Q*, *PQ*, *Res*)
    if (*T* is leaf) **then**
        Res.addAll(T.data) with distances to *Q*
    else
        C = *T.child* nearest to *Q*
        PQ.addAll(*T.child* except *C*) with
            distances to *Q*
        Traverse(*C*, *Q*, *PQ*, *Res*)
    end if

These are a NN search with DTW *O*(L^2) time

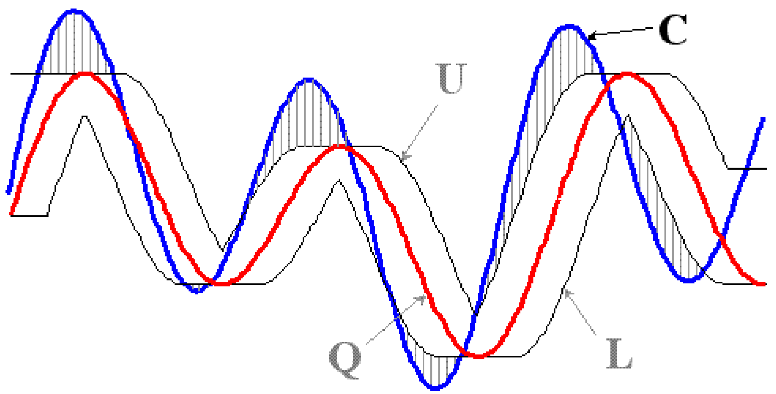
Apply DTW lower bounds, **LB Keogh** to minimize DTW computations and have **2 PQ**

Lower Bound Keogh (LB Keogh)

1. Computes **Upper** (**U**) and **Lower** (**L**) envelope for query *Q*
2. Computes the distance of the projection of a candidate sequence *C* onto the envelope

Only need to compute the envelopes for *Q* once!!

$$LB\_Keogh(Q,C)=\sum_{i=1}^{n}\begin{cases}(q_i-U_i)^2 & if\ q_i>U_i\\(q_i-L_i)^2 & if\ q_i<L_i\\0 & otherwise\end{cases}$$



http://www.cs.ucr.edu/~eamonn/LB_Keogh.htm

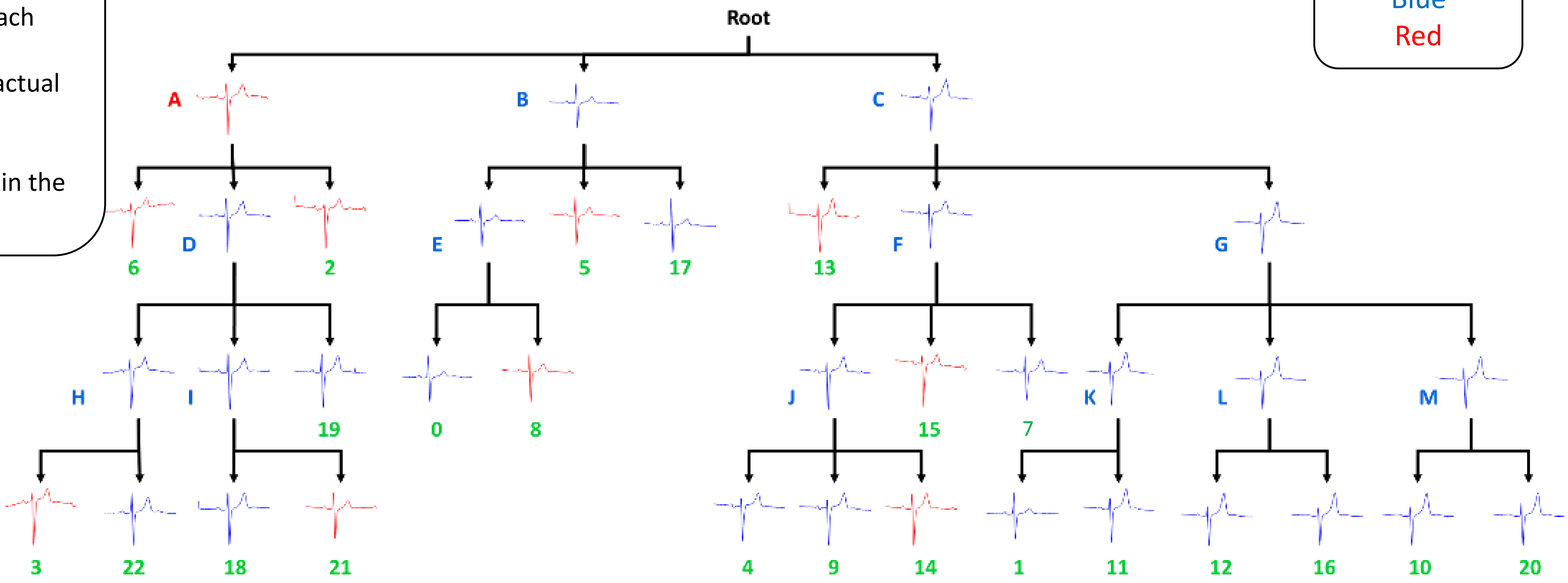
[1] Keogh, E. (2002, August). Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases* (pp. 406-417). VLDB Endowment.

Simple example

Time Series Indexing Example

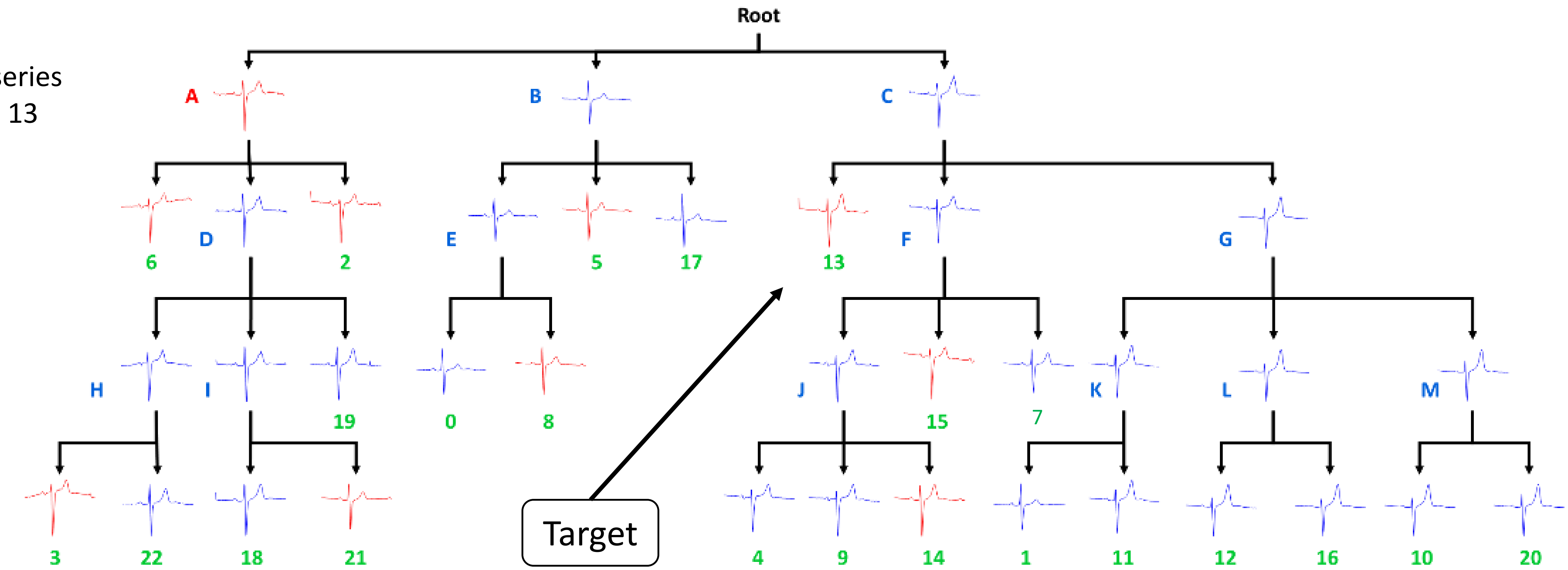
- **Alphabets** are Centroids of each cluster
- **Numbers** are actual time series in training set
- 23 time series in the training set

Classes:
Blue
Red

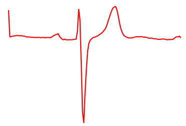


Time Series Indexing Example

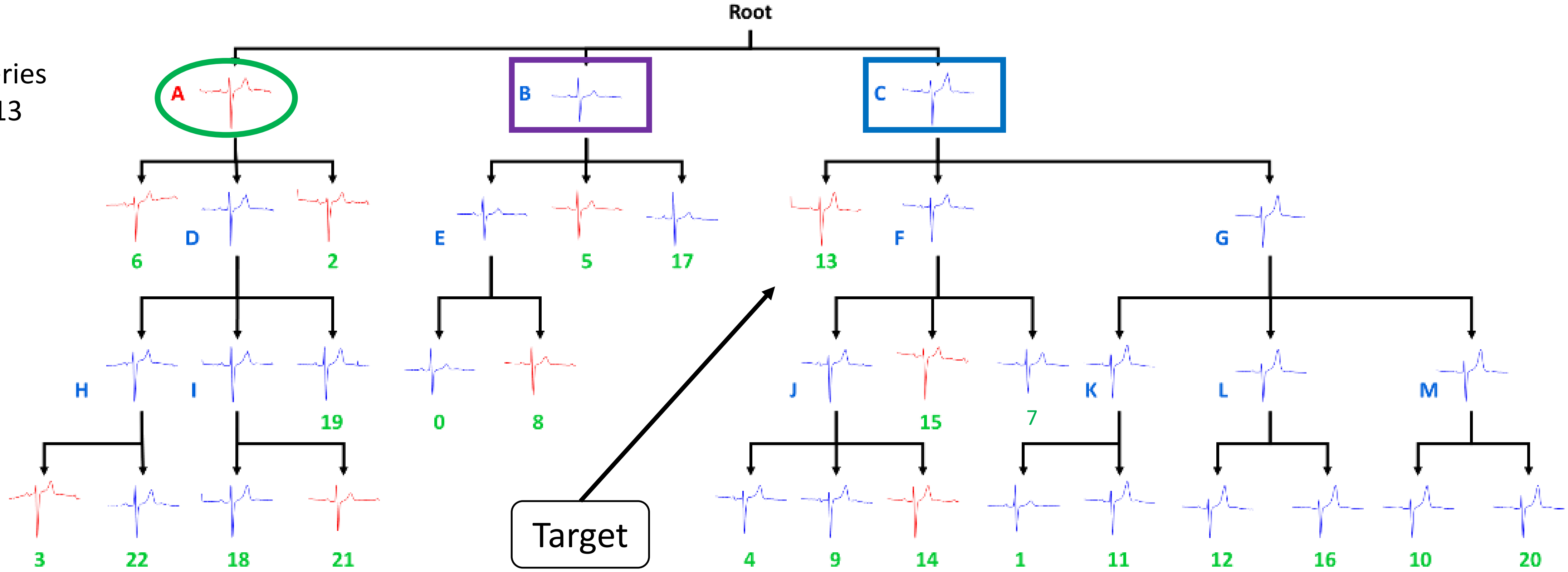
Query time series
Actual NN: 13



Time Series Indexing Example



Query time series
Actual NN: 13



LB Distance to
A: 0.895
B: 6.157
C: 0.814

DTW Distance to
A: 4.893
B: Skip (16.920)
C: 5.231

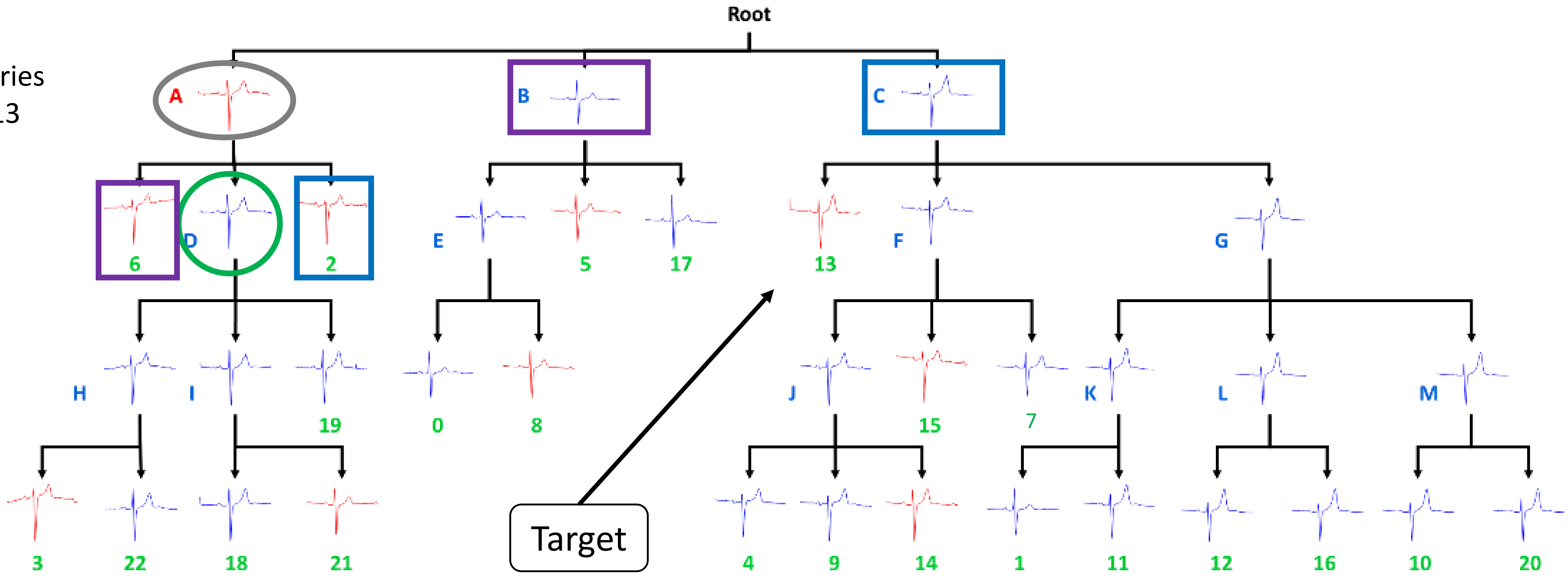
LB Priority Queue : {B}
Priority Queue Distance to Query : {6.2}
DTW Priority Queue : {C}
Priority Queue Distance to Query : {5.2}

Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
6: 20.253
D: 0.573
2: 0.781

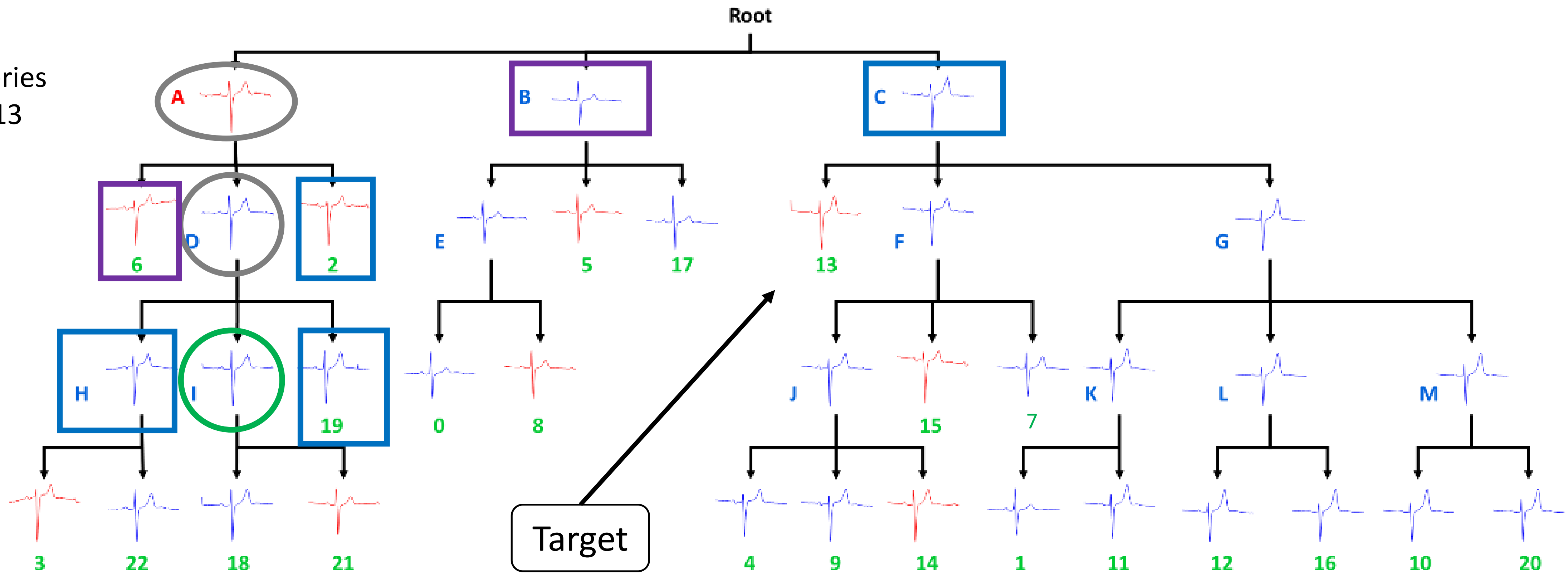
DTW Distance to
6: Skip (40.592)
D: 6.668
2: 10.194



LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {C, 2}
Priority Queue Distance to Query : {5.2, 10.2}

Time Series Indexing Example

Query time series
Actual NN: 13



LB Distance to
H: 1.252
I: 0.726
19: 1.321

DTW Distance to
H: 11.387
I: 4.839
19: 9.335

LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {C, 19, H, 2}
Priority Queue Distance to Query : {5.2, 9.3, 11.4, 10.2}

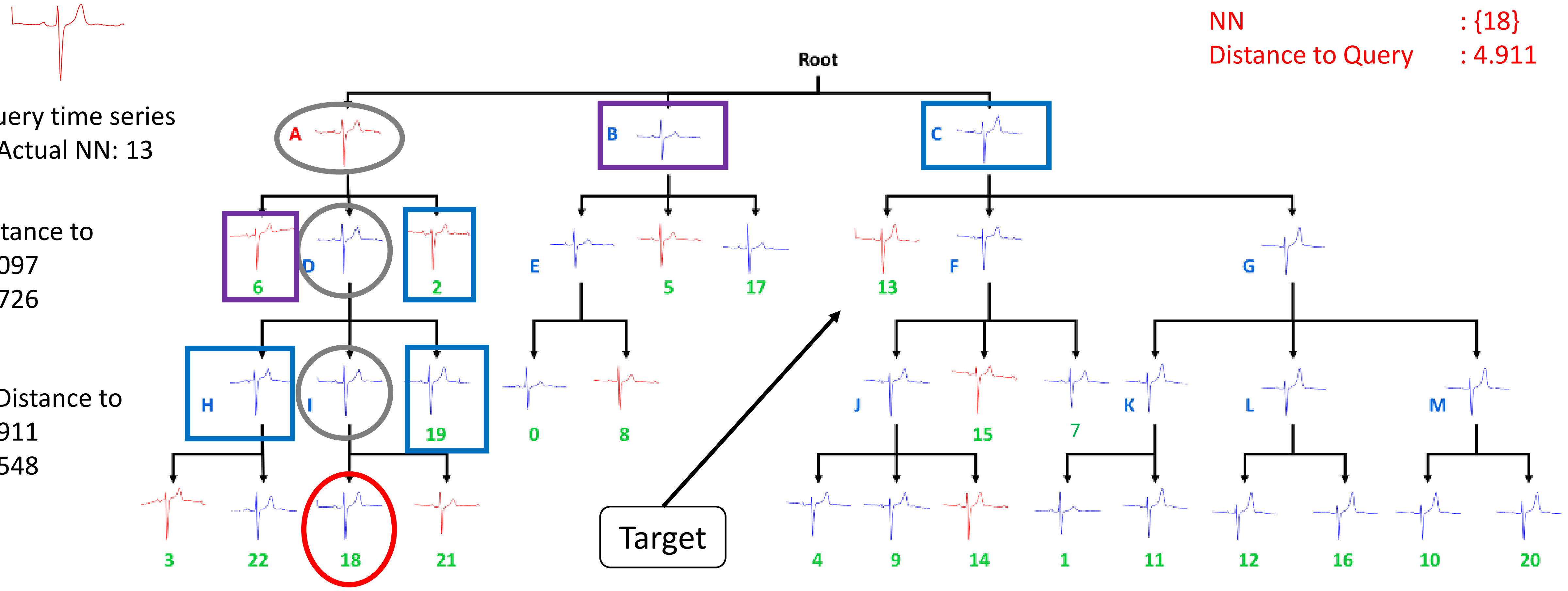
Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
18: 1.097
21: 1.726

DTW Distance to
18: 4.911
21: 9.548

NN : {18}
Distance to Query : 4.911

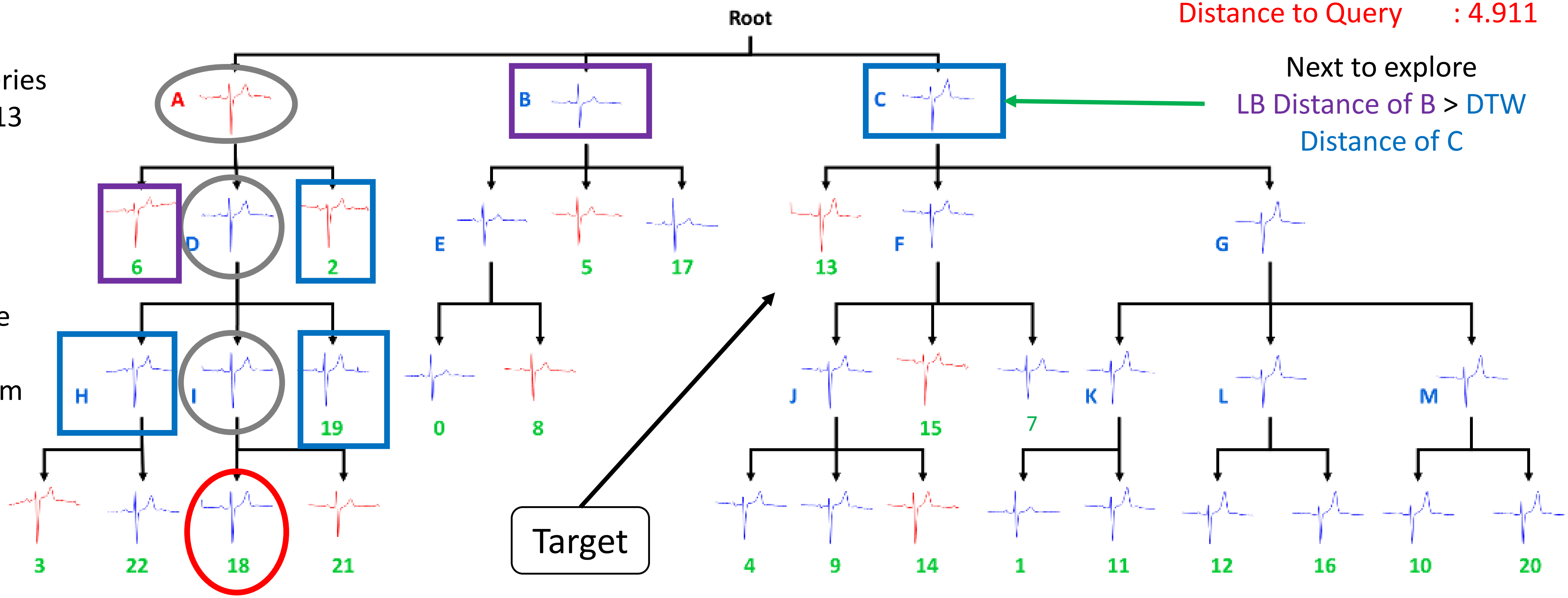


LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {C, 19, H, 2}
Priority Queue Distance to Query : {5.2, 9.3, 11.4, 10.2}

Time Series Indexing Example

Query time series
Actual NN: 13

- Current NN is 18, Class 1
- Not actual NN
- Next to explore is Node C
- Dequeue C from DTW Priority Queue

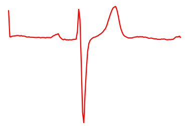


NN : {18}
Distance to Query : 4.911

Next to explore
LB Distance of B > DTW Distance of C

LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {C, 19, H, 2}
Priority Queue Distance to Query : {5.2, 9.3, 11.4, 10.2}

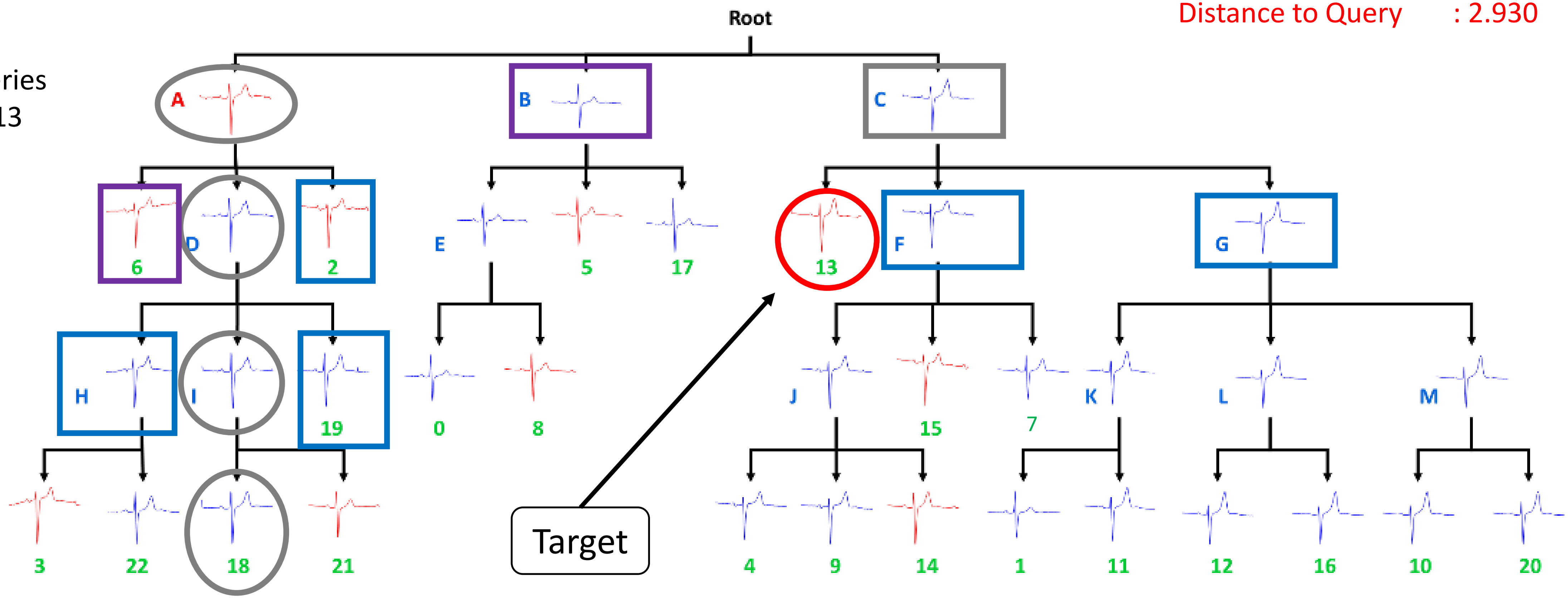
Time Series Indexing Example



LB Distance to
13: 0.672
F: 0.497
G: 2.585

DTW Distance to
13: 2.930
F: 4.249
G: 11.446

NN : {13}
Distance to Query : 2.930



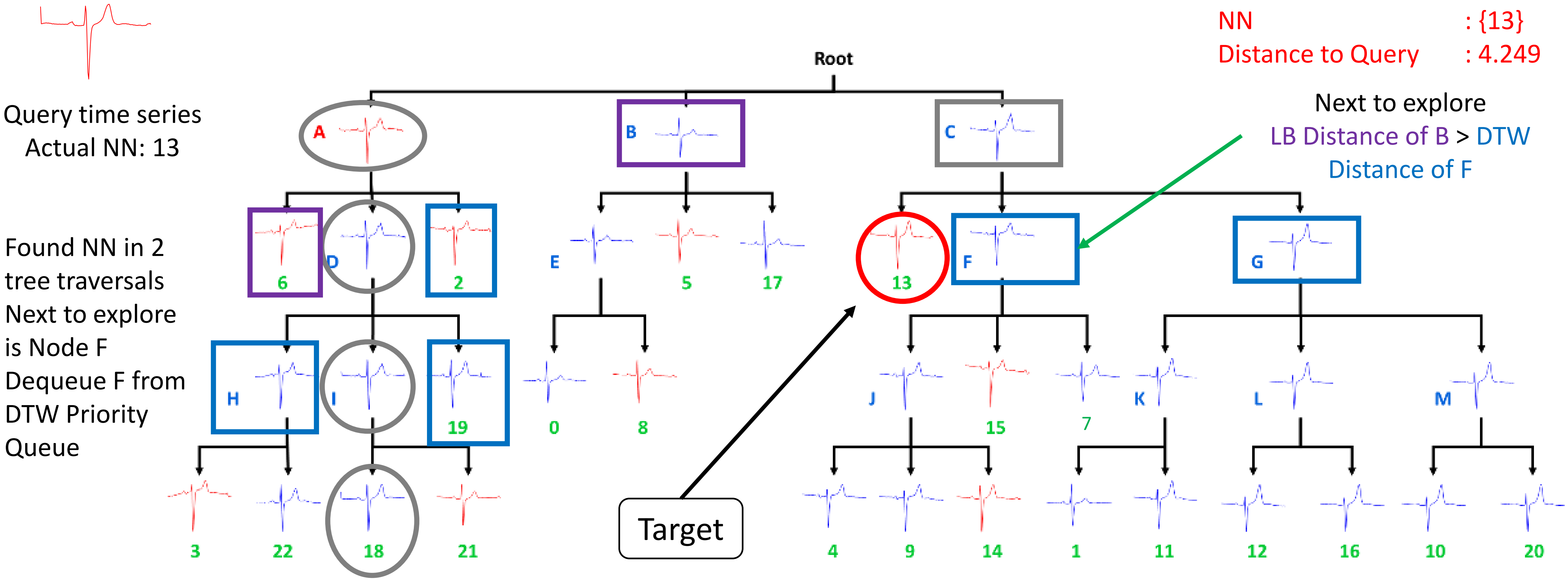
LB Priority Queue : {B, 6}

Priority Queue Distance to Query : {6.2, 20.3}

DTW Priority Queue : {F, 19, H, 2, G}

Priority Queue Distance to Query : {4.2, 9.3, 11.4, 10.2, 11.4}

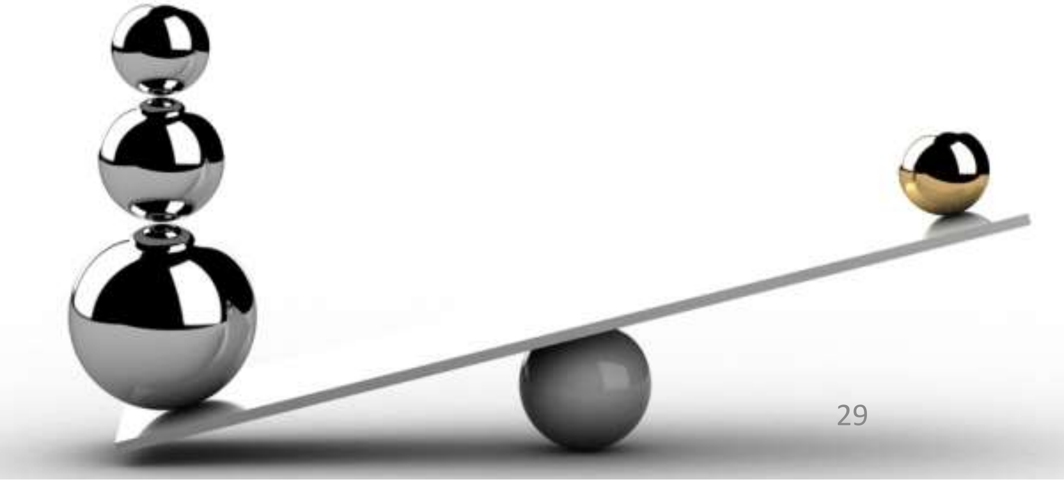
Time Series Indexing Example



- Found NN in 2 tree traversals
- Next to explore is Node F
- Dequeue F from DTW Priority Queue

LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {F, 19, H, 2, G}
Priority Queue Distance to Query : {4.2, 9.3, 11.4, 10.2, 11.4}

Comparison with state of the art



Experiments

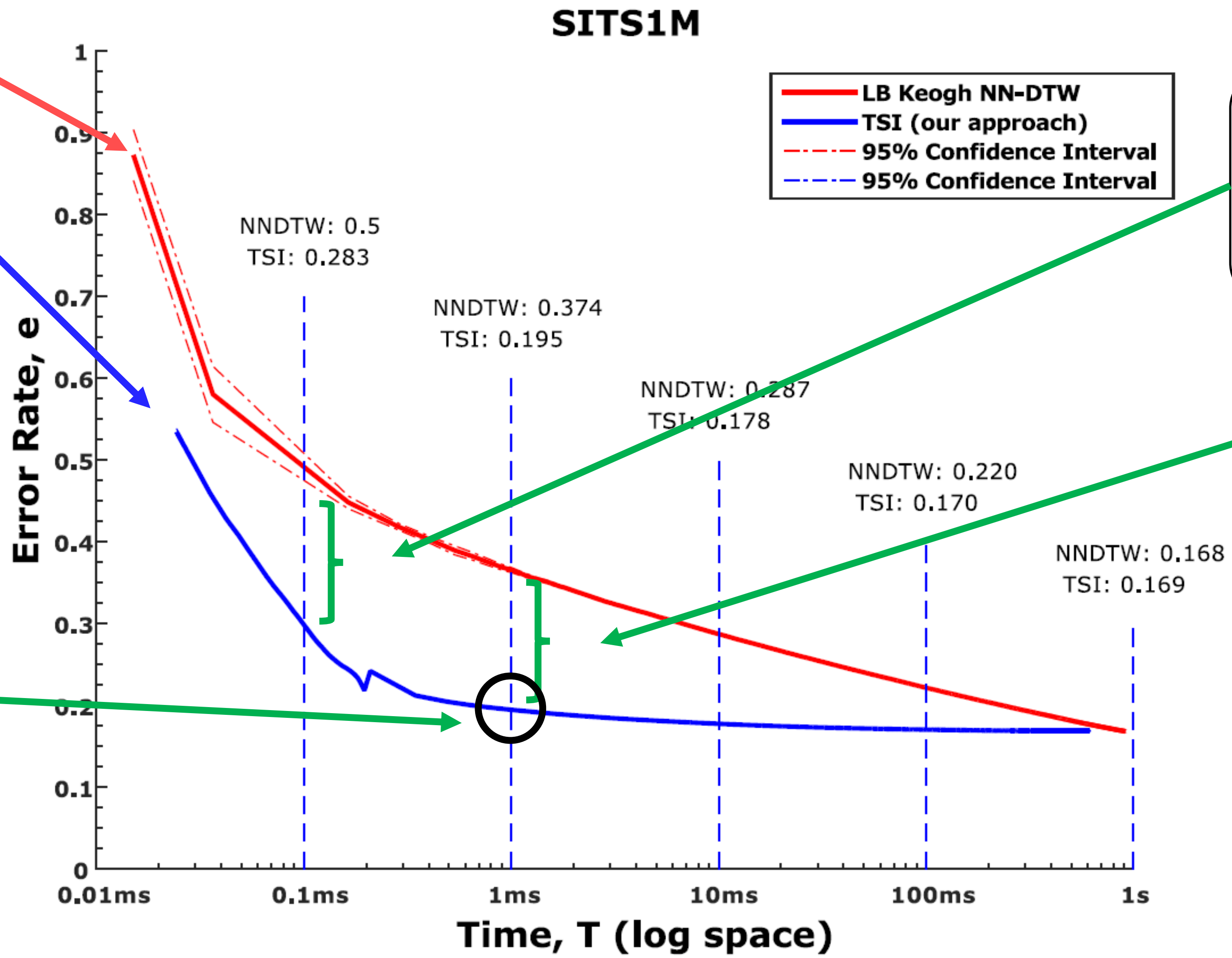
- Compared with NN-DTW with LB_Keogh
  - at **x %** of the time of the full NN-DTW
  - 1%, 10%, 20%, 30%, 40%, 50%
- Satellite Dataset
  - Train 1M series
  - Length 46
  - Number of classes: 24
- 84 UCR Repository [1]

[1] Chen, Yanping, et al. "The ucr time series classification archive." URL www.cs.ucr.edu/~ eamonn/time_series_data (2015).

Results on the satellite data

State of the art – random sampling

Our approach

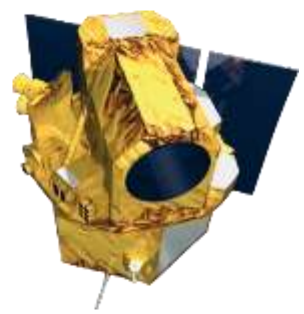


If given only 0.1ms to classify a pixel, we do better by **22%**

At 1ms to classify a pixel, we do better by **18%**

Almost same accuracy as full search but 1,000x faster!

- **Classifying Houston would take 4 hours instead of 1 year!**



Performance on UCR repository

- Look at how well we perform if we are given **x %** of the time of the full NN-DTW.

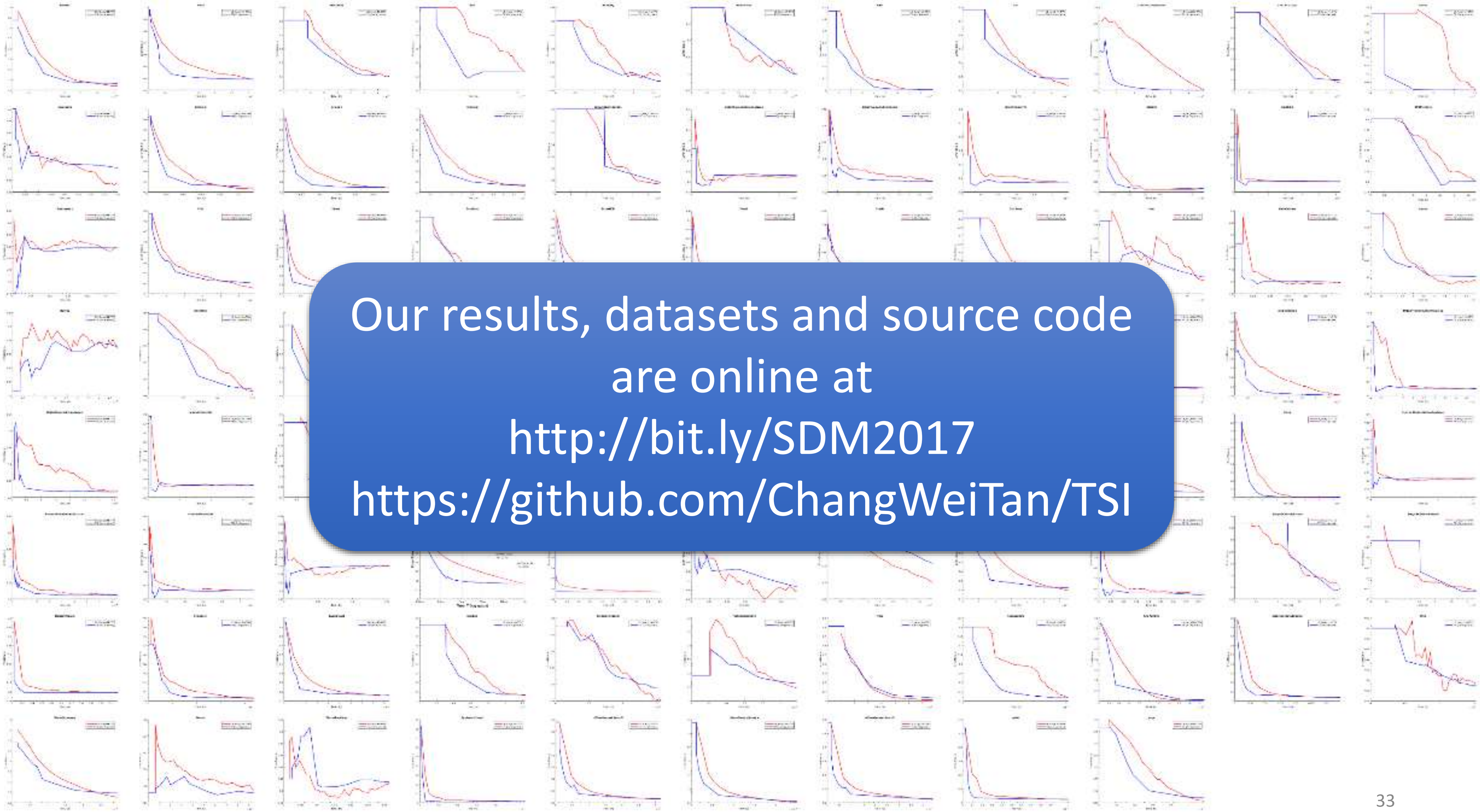
LB_Keogh NN-DTW vs TSI					
	Average Ranks		Wilcoxon Test Statistics		
Intervals	NN-DTW	TSI	R^+	R^-	z
1%	1.529	**1.471**	2034.5	1620.5	-0.907
10%	1.841	**1.159**	3449	206	**-7.105**
20%	1.871	**1.129**	3451	204	**-7.114**
30%	1.806	**1.194**	3219.5	435.5	**-6.099**
40%	1.741	**1.259**	2903	752	**-4.713**
50%	1.671	**1.329**	2616	1039	**-3.455**
Average	1.743	**1.257**			

There isn't enough time to see 1 data point for most of the dataset

Statistically significant

TSI performs better even on smaller datasets with average training size < 500

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research, 7(Jan), 1-30.



Our results, datasets and source code are online at

http://bit.ly/SDM2017

https://github.com/ChangWeiTan/TSI

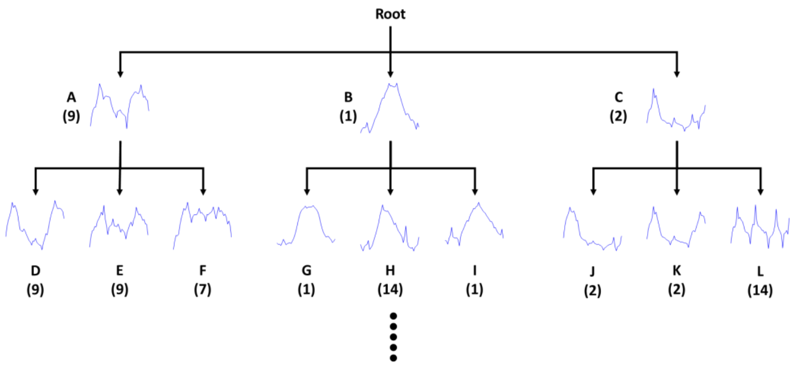
Future Work

- Pruning the whole branch
  - Atomic Wedgie [1]
  - If everything in that branch is of the same class
- Optimizing the branching factor, *K*
  - Vary *K* and keep the *K* value that gives the best trade-off between query time and error rate.
- Speeding up search for the best warping window on large dataset
  - Current method via Cross Validation

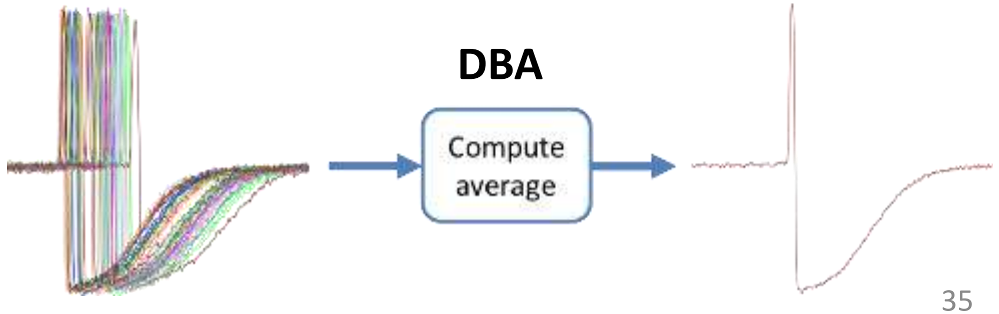
[1] Wei, L., Keogh, E., Van Herle, H., & Mafra-Neto, A. (2005, November). Atomic wedgie: efficient query filtering for streaming time series. In Data Mining, Fifth IEEE International Conference on (pp. 8-pp). IEEE.

Take home message

1. The first algorithm (**TSI**) to index DTW-induced space
   - Hierarchical **K-means tree**
   - DTW Barycenter Averaging (**DBA**)
2. Twice the accuracy than NN-DTW on large (**1M**) remote sensing data if given 1ms to classify a query
3. Perform better even on smaller datasets
   - If we just have 50% of the full search time.



+



MONASH University
Information Technology

chang.tan@monash.edu
https://github.com/ChangWeiTan/TSI
http://bit.ly/SDM2017

Thank you!

Questions and Answers

This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-16-1-4023. This work was supported by the Australian Research Council under awards DE170100037 and DP140100087, and by the 2016 IBM Faculty Award (F. Petitjean).

Australian Government
Australian Research Council

AIR FORCE RESEARCH LABORATORY

IBM
Faculty Award

Additional Information

Lower Bound for DTW

```
LinearScan(Q)
    bestSoFar = infinity
    for each sequence S in database


        dtwDist = DTW(Q, S)
        if (dtwDist < bestSoFar) then
            bestSoFar = dtwDist
            nn = S
        end if

    end for
    return nn
```

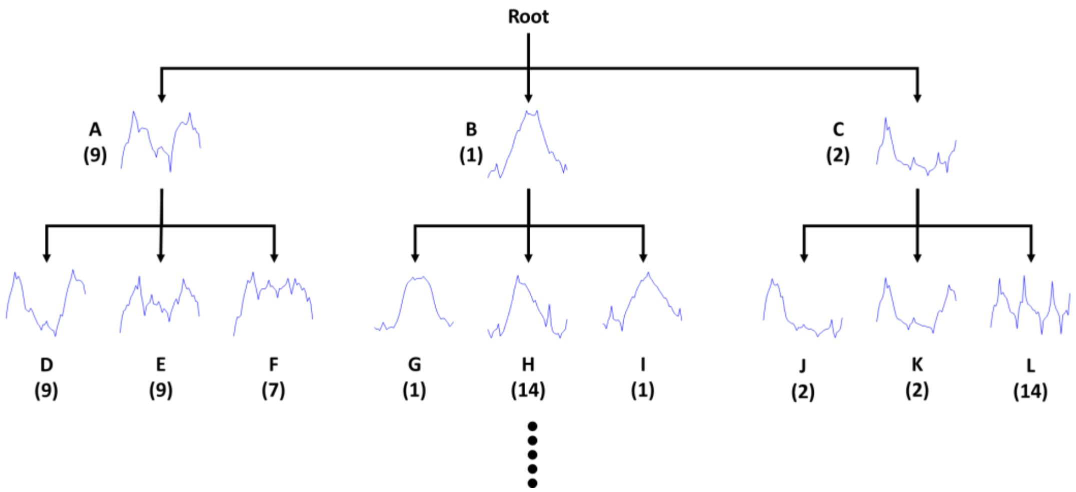
```
LowerBoundScan(Q)
    bestSoFar = infinity
    for each sequence S in database
        lbDist = LowerBound(Q, S)
        if (lbDist < bestSoFar) then
            dtwDist = DTW(Q, S)
            if (dtwDist < bestSoFar) then
                bestSoFar = dtwDist
                nn = S
            end if
        end if
    end for
    return nn
```

Cheap test before computing the actual DTW distance

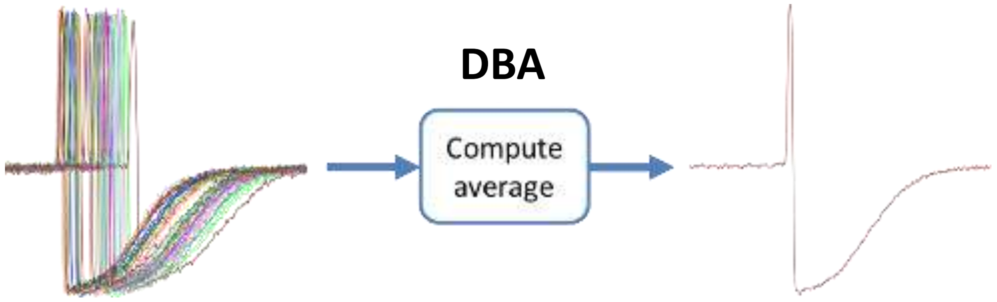
Time Series Indexing

- At training time

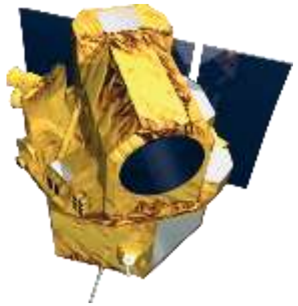
BuildTree(*data*, *K*)
  if ($|data| \le K$) **then**
    create leaf node with all the data
  else
    (C,P) = Kmeans($data,K$)
    for each *cluster* C_i **do**
      create node N_i = **BuildTree**(C_i , K)
      assign center P_i to N_i
    end for
  end if

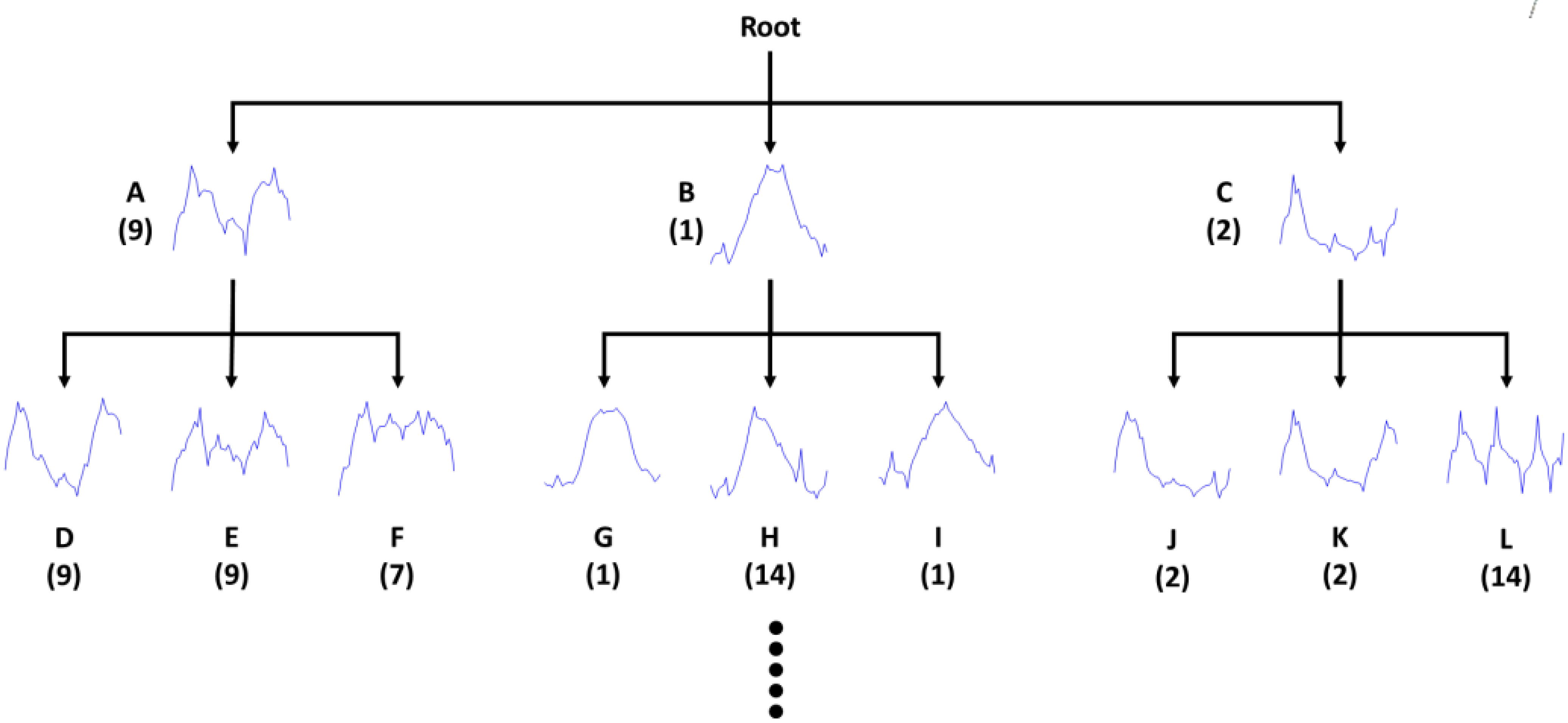


Replace arithmetic mean with **DBA**



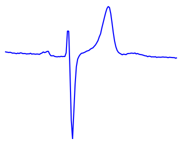
Example on SITS Dataset





Example 2

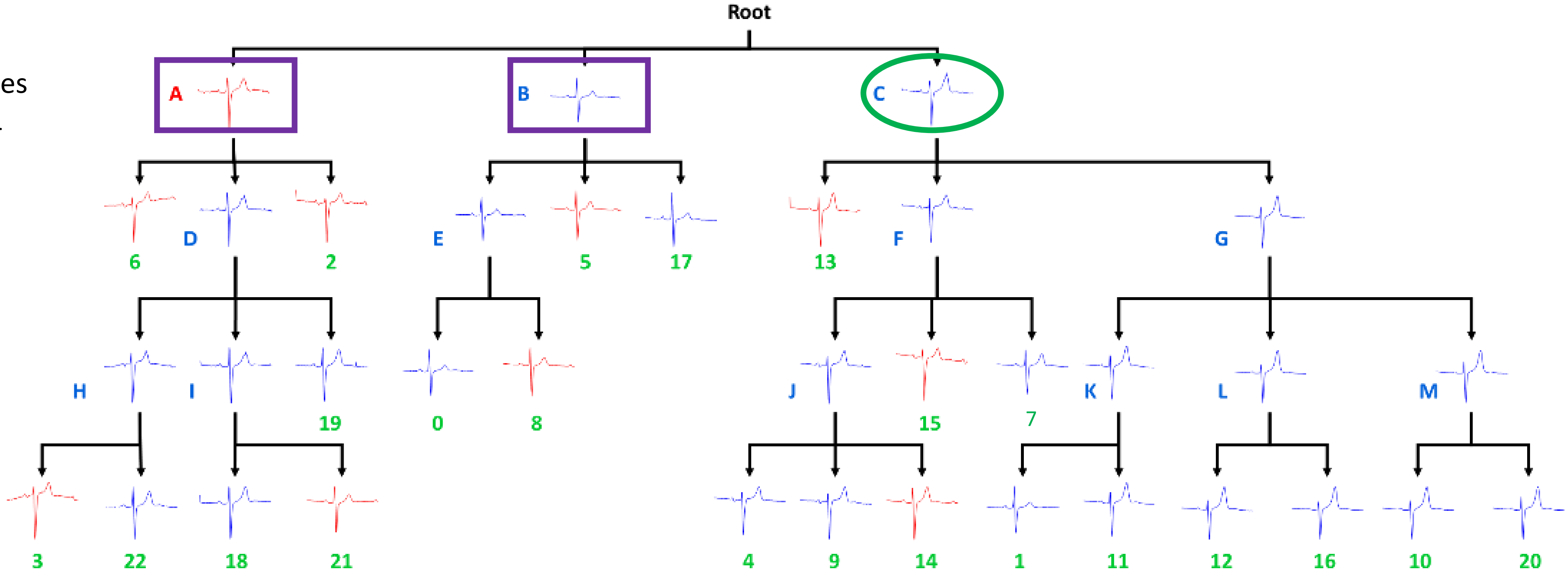
Time Series Indexing Example



Query time series
Actual NN: 11

LB Distance to
A: 2.990
B: 10.900
C: 0.302

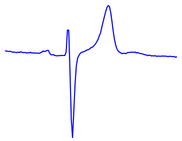
DTW Distance to
A: Skip (2.917)
B: Skip (5.348)
C: 1.316



Target

LB Priority Queue	: {A, B}
Priority Queue Distance to Query	: {3.0, 10.9}
DTW Priority Queue	: {}
Priority Queue Distance to Query	: {}

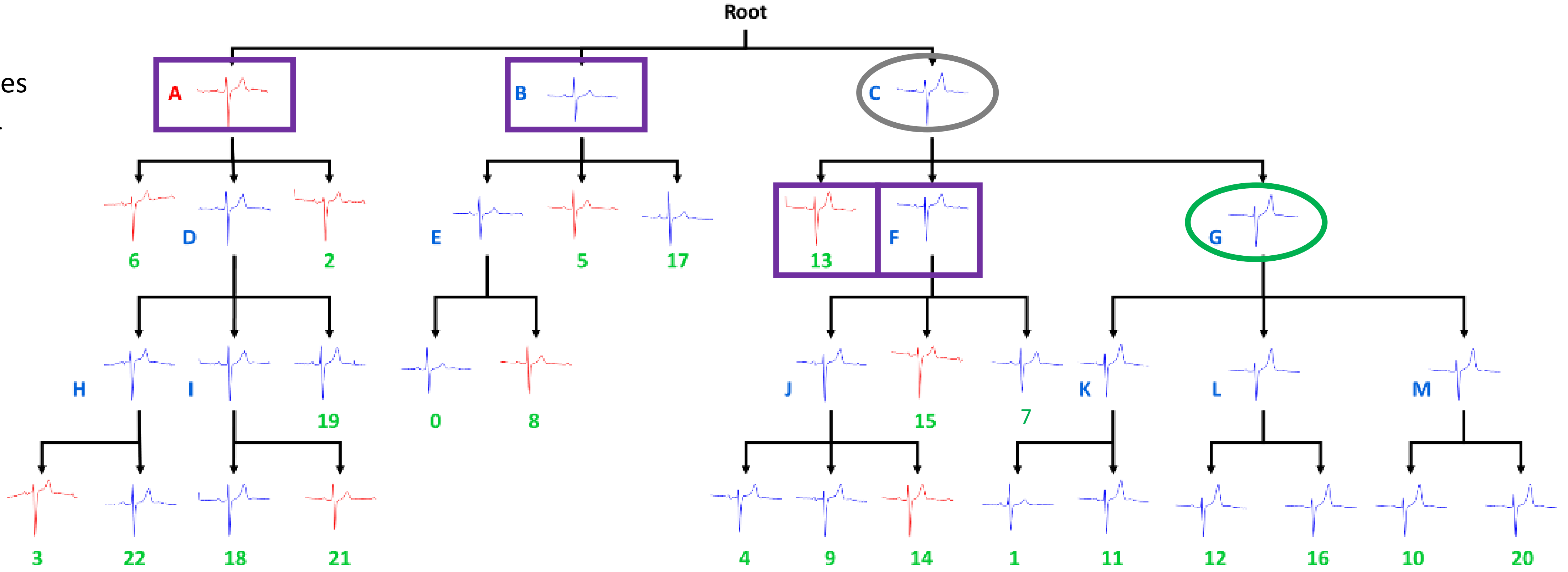
Time Series Indexing Example



Query time series
Actual NN: 11

LB Distance to
13: 4.087
F: 1.876
G: 0.047

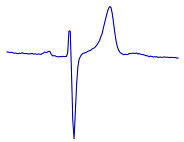
DTW Distance to
13: Skip (2.536)
F: Skip (2.592)
G: 0.9998



Target

LB Priority Queue : {F, A, 13, B}
Priority Queue Distance to Query : {1.9, 3.0, 4.1, 10.9}
DTW Priority Queue : {}
Priority Queue Distance to Query : {}

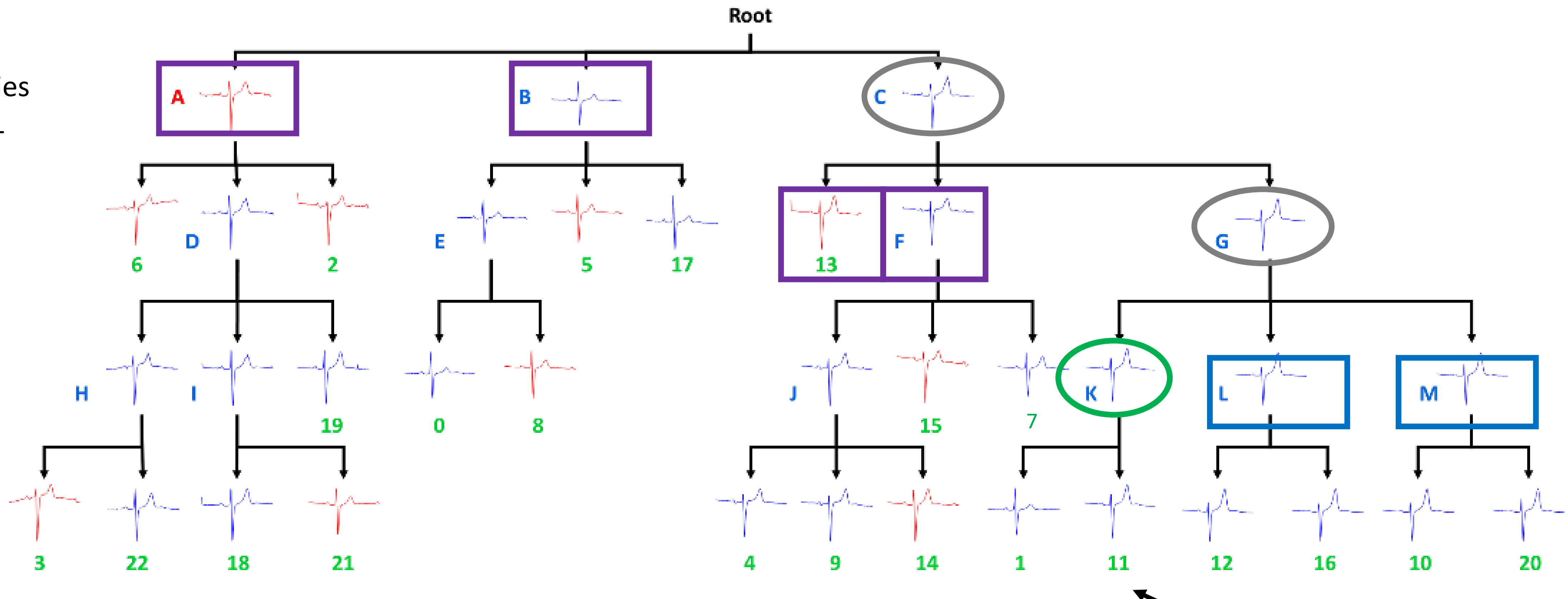
Time Series Indexing Example

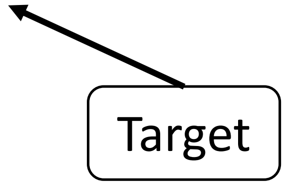


Query time series
Actual NN: 11

LB Distance to
K: 0.059
L: 0.225
M: 0.226

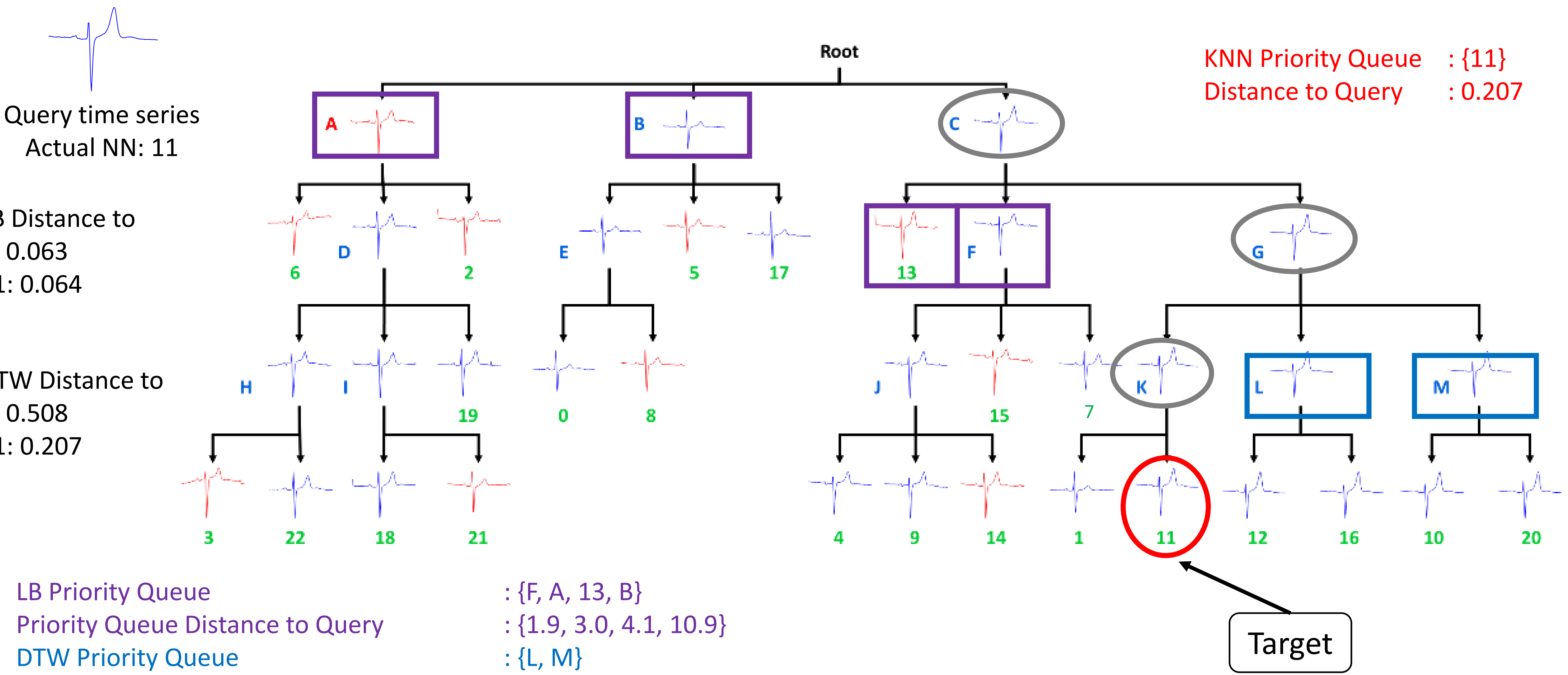
DTW Distance to
K: 0.281
L: 2.913
M: 3.791





LB Priority Queue : {F, A, 13, B}
Priority Queue Distance to Query : {1.9, 3.0, 4.1, 10.9}
DTW Priority Queue : {L, M}
Priority Queue Distance to Query : {2.9, 3.8}

Time Series Indexing Example



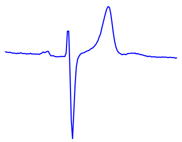
KNN Priority Queue : {11}
Distance to Query : 0.207

LB Distance to
1: 0.063
11: 0.064

DTW Distance to
1: 0.508
11: 0.207

LB Priority Queue : {F, A, 13, B}
Priority Queue Distance to Query : {1.9, 3.0, 4.1, 10.9}
DTW Priority Queue : {L, M}
Priority Queue Distance to Query : {2.9, 3.8}

Time Series Indexing Example

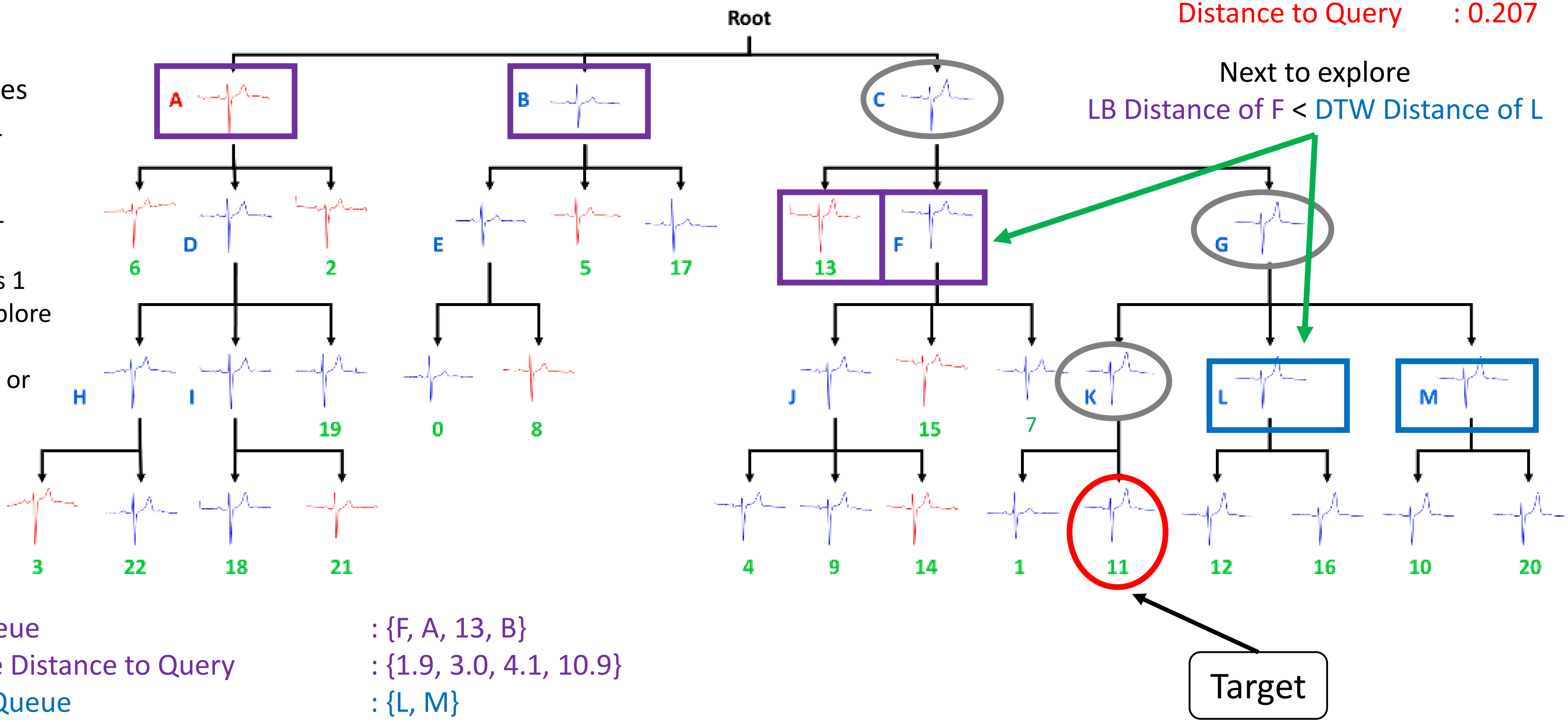


Query time series
Actual NN: 11

KNN Priority Queue : {11}
Distance to Query : 0.207

Next to explore
LB Distance of F < DTW Distance of L

- Found NN in 1 tree traversal
- Assign to class 1
- Next to be explore is node **L** or **F**
- Can stop here or until contract exhausted



LB Priority Queue : {F, A, 13, B}
Priority Queue Distance to Query : {1.9, 3.0, 4.1, 10.9}
DTW Priority Queue : {L, M}
Priority Queue Distance to Query : {2.9, 3.8}