@S S Roy,9th Sept,2025

Agglomerative (Single-link / Min-distance) Hierarchical

Problem statement (dataset chosen)

We have 6 one-dimensional data points:

Data: `12, 17, 18, 24, 50, 52`

Use **agglomerative hierarchical clustering** with **single-link (min distance)**. At each iteration:

1. Find the two clusters with the smallest inter-cluster distance (single-link = minimum pairwise distance between points of the clusters).
2. Merge them.
3. Update the proximity matrix using single-link (min of pairwise distances).
   Show the proximity matrix for each iteration and the final dendrogram.

Key reminder (single-link update)

If cluster A and cluster B are merged into cluster `A∪B`, the distance between cluster `A∪B` and any other cluster C under **single-link** is:

```r
d_single(A∪B, C) = min( d(a, c) for a in A, c in C )  = min( d(A, C), d(B, C) )
```

(So you update a cluster-to-cluster distance by taking the **minimum** of the prior distances.)

---

1) Initial proximity (pairwise absolute differences)

We list the points in ascending order and compute |xi − xj|.

```makefile
Points:  12   17   18   24   50   52
Index:   P1   P2   P3   P4   P5   P6
```

Initial symmetric proximity matrix (rows = clusters, columns = clusters). Diagonal = 0.

```java
Initial proximity matrix (absolute differences)

      12   17   18   24   50   52
12     0    5    6   12   38   40
17     5    0    1    7   33   35
18     6    1    0    6   32   34
24    12    7    6    0   26   28
```

```
50   38  33  32  26   0   2
52   40  35  34  28   2   0
```

Check a few entries (digit-by-digit):

- `|17 − 18| = 1` (smallest).
- `|50 − 52| = 2`.
- `|12 − 24| = 12`.
- `|24 − 50| = 26`.

---

2) Merge history and matrices — step by step

I will number iterations. At each step I show which clusters are merged and the updated matrix (labels show current clusters).

Step 0 — start

Clusters: `{12}, {17}, {18}, {24}, {50}, {52}`

Matrix shown above.

Smallest distance = **1** between `17` and `18`. So we merge `{17}` & `{18}`.

Step 1 — merge `{17}` and `{18}`

Merged clusters: `(17,18)` at distance **1**.

New cluster set (sorted for display):

`{12}, { (17,18) }, {24}, {50}, {52}`

Proximity matrix after the merge (single-link):

```markdown
Step 1: clusters = [12, (17,18), 24, 50, 52]

      12   (17,18)  24   50   52
12      0      5      12   38   40
(17,18) 5      0       6   32   34
24     12      6       0   26   28
```

```
50   38   32   26  0  2
52   40   34   28  2  0
```

How entries were computed (examples):

- `d( (17,18), 12 ) = min(|17−12|, |18−12|) = min(5,6) = 5`.
- `d( (17,18), 24 ) = min(|17−24|, |18−24|) = min(7,6) = 6`.
- distances to 50 and 52 use the minimum of pairwise differences (32 and 34 respectively).

Next smallest distance = **2** between `50` and `52`. Merge `{50}` & `{52}`.

Step 2 — merge `{50}` and `{52}`

Merged clusters: `(50,52)` at distance **2**.

New clusters: `{12}, {(17,18)}, {24}, {(50,52)}`

Proximity matrix:

```markdown
Step 2: clusters = [12, (17,18), 24, (50,52)]

       12   (17,18)   24   (50,52)
12      0      5      12     38
(17,18) 5      0       6     32
24     12      6       0     26
(50,52)38     32      26      0
```

Notes:

- `d((50,52), 24) = min(|50−24|,|52−24|) = min(26,28) = 26`.
- `d((50,52), (17,18)) = min(33,32,35,34) = 32` (min among 50−17, 50−18, 52−17, 52−18).

Next smallest distance = **5** between `12` and `(17,18)`. Merge `{12}` with `{17,18}`.

Step 3 — merge `{12}` and `(17,18)`

Merged clusters: `(12,17,18)` at distance **5**.

New clusters: `{ (12,17,18) }, {24}, { (50,52) }`

Proximity matrix:

```markdown
Step 3: clusters = [(12,17,18), 24, (50,52)]

            (12,17,18)   24   (50,52)
(12,17,18)      0        6      32
24              6        0      26
(50,52)        32       26       0
```

Check computation:

- `d((12,17,18), 24) = min(|12−24|,|17−24|,|18−24|) = min(12,7,6) = 6`.
- `d((12,17,18), (50,52)) = min(38,40,33,35,32,34) = 32` → the minimum is `|18 − 50| = 32`.

Next smallest distance = **6** between `(12,17,18)` and `24`. Merge them next.

Step 4 — merge `(12,17,18)` and `24`

Merged clusters: `(12,17,18,24)` at distance **6**.

Now clusters: `{ (12,17,18,24) }, { (50,52) }`

Proximity matrix:

```markdown
Step 4: clusters = [(12,17,18,24), (50,52)]

               (12,17,18,24)   (50,52)
(12,17,18,24)        0            26
(50,52)             26             0
```

Check:

- `d( (12,17,18,24), (50,52) ) = min( distances of any element in left cluster to 50 or 52 )`.

The minimum is `|24 − 50| = 26` (smaller than all the other pairwise differences).

Final merge merges these two clusters at distance **26**.

Step 5 — final merge

Merge `(12,17,18,24)` with `(50,52)` at distance **26** → final cluster `{12,17,18,24,50,52}`.

---

3) Merge summary (compact)

Merge #	Clusters merged	Distance at merge	New resulting cluster
1	`{17}` & `{18}`	1	`(17,18)`
2	`{50}` & `{52}`	2	`(50,52)`
3	`{12}` & `(17,18)`	5	`(12,17,18)`
4	`(12,17,18)` & `{24}`	6	`(12,17,18,24)`
5	`(12,17,18,24)` & `(50,52)`	26	`(12,17,18,24,50,52)`

---

4) Dendrogram (parentheses and interpretation)

Parenthetical (Newick-like) representation:

```scss
(((12,(17,18)),24),(50,52))
```

If we annotate the *heights* (merge distances):

```graphql
(((12, (17,18) @1) @5, 24) @6, (50,52) @2) @26
```

Interpreting the dendrogram left → right:

- 17 and 18 join very close (height = 1).
- 50 and 52 join very close (height = 2) in their own cluster.
- 12 joins the 17–18 pair at distance 5.
- That group joins with 24 at distance 6.
- Finally the big left cluster meets the (50,52) cluster at distance 26.

(If you draw this exactly like your screenshots, the y-axis (height) would show jumps at 1, 2, 5, 6, and 26.)

---

ghts & best-practice notes

- **Single-link (min)** tends to produce **chains**: points can be pulled into a cluster if they are close to *any* member. That is visible here: `12` merges into the small cluster `{17,18}` even though `12` is not that close to `18` (it's closer to `17`). This chaining is a property to keep in mind.
- **Choosing number of clusters:** cut the dendrogram at a horizontal line. For example:
  - cut at height < 6 → clusters would be `{12,17,18}` and `{24}` and `{50,52}` (3 clusters)
  - cut at height between 6 and 26 → clusters `{12,17,18,24}` and `{50,52}` (2 clusters)
  - cut below 2 → clusters `{17,18}`, `{50,52}`, `12`, `24` (4 clusters)
- **Other linkages:** complete-link uses max distances (compact clusters), average uses mean pairwise distance (smoother), Ward minimizes increase in total within-cluster variance (balanced clusters). If you want compact spherical clusters, complete or Ward may be more appropriate.
- **Complexity:** the naive agglomerative implementation recomputes matrices each merge → $O(n^3)$. With appropriate data structures (priority queues, nearest-neighbor tracking) you can achieve roughly $O(n^2)$ time and $O(n^2)$ memory for many linkage methods.
- **Numeric correctness:** all distances above are absolute differences computed exactly (checked step-by-step).

---

6) Appendix — quick checks & example calculations

- `d(17,18) = |17−18| = 1` → first merge.
- After merging `{17,18}`, `d((17,18), 24) = min(|17−24|,|18−24|) = min(7,6) = 6`.
- `d(50,52) = |50−52| = 2` → second merge.
- `d((12,17,18), 24) = min(|12−24|,|17−24|,|18−24|) = min(12,7,6) = 6`.
- `d((12,17,18,24), (50,52)) = min(all pairwise differences) = min(..., |24−50|=26, ...) = 26`.

@S S Roy,9th Sept,2025