DEPARTMENT OF COMPUTER SCIENCE

HSPC 2024: Solution Sketches

Problem	Title
P1	Koopa Rescue Mission
P2	Mario’s Magic Matrix
P3	Mario’s Perilous Path
P4	Princess Peach’s Garden Party
P5	Maximizing Harmony
P6	Magical Platform Planning for the Star Festival
P7	Chompy Chain Clusters
P8	Colorful Chaos in the Mushroom Kingdom!
P9	Bowser’s Diabolical Parentheses

$P1$: Koopa Rescue Mission

Problem: Given a string S, and strings S1, S2:

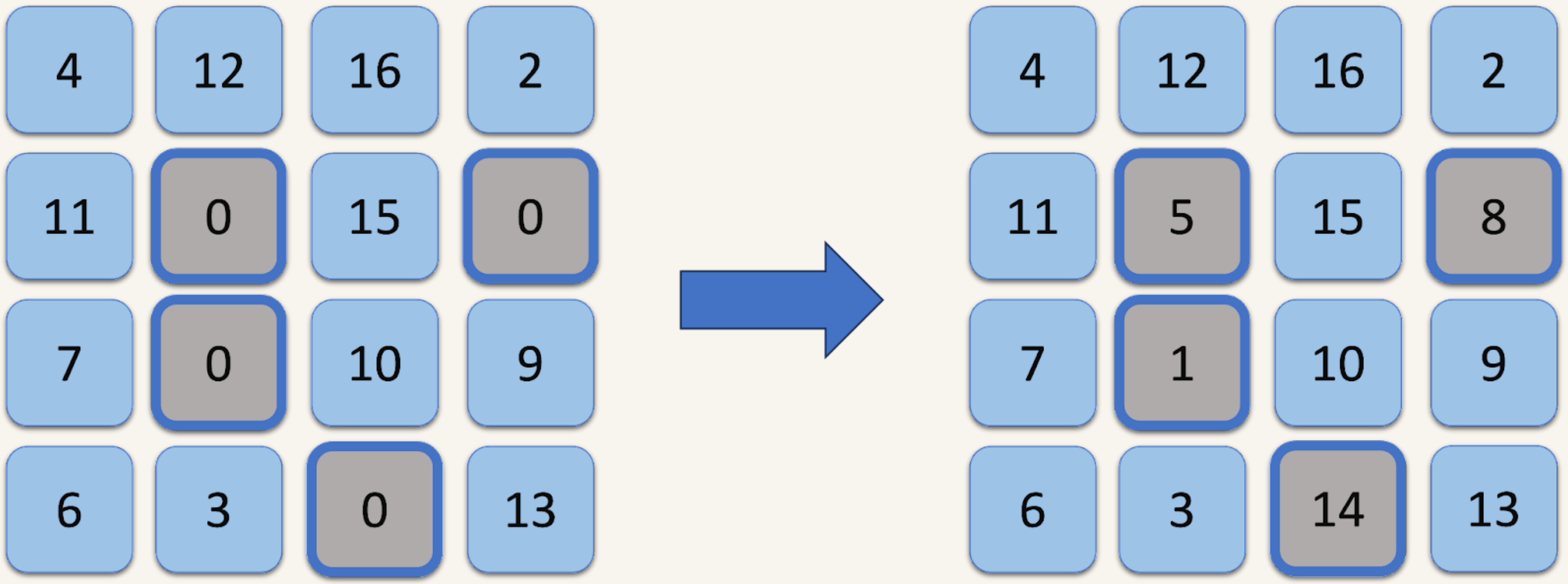
- If S2 is present in S, compute the number of times a string S1 appears in S, and

Solution: First check if S2 is present using S.contains(S2). If it does, repeatedly call S.indexOf(S1, offset) to count all occurrences of S1 in S.

```
private static int lookForKoopa(String str) {
  int result = -1;
  if (str.contains("Peach") ) {
    //count the Koopas
    result = 0;
    int searchPos = 0;
    int resultPos;
    while ((resultPos=str.indexOf("Koopa",searchPos)) != -1) {
      result++;
      searchPos=resultPos+5;
    }
  }
  return result;
}
```

$P2$: Mario's Magic Matrix

Problem: Given a 4 x 4 grid containing the numbers [1, 16] (each once). 4 cells are erased (along with 4 numbers in [1, 16]). Find the best way to place the numbers to maximize the sum of row products:

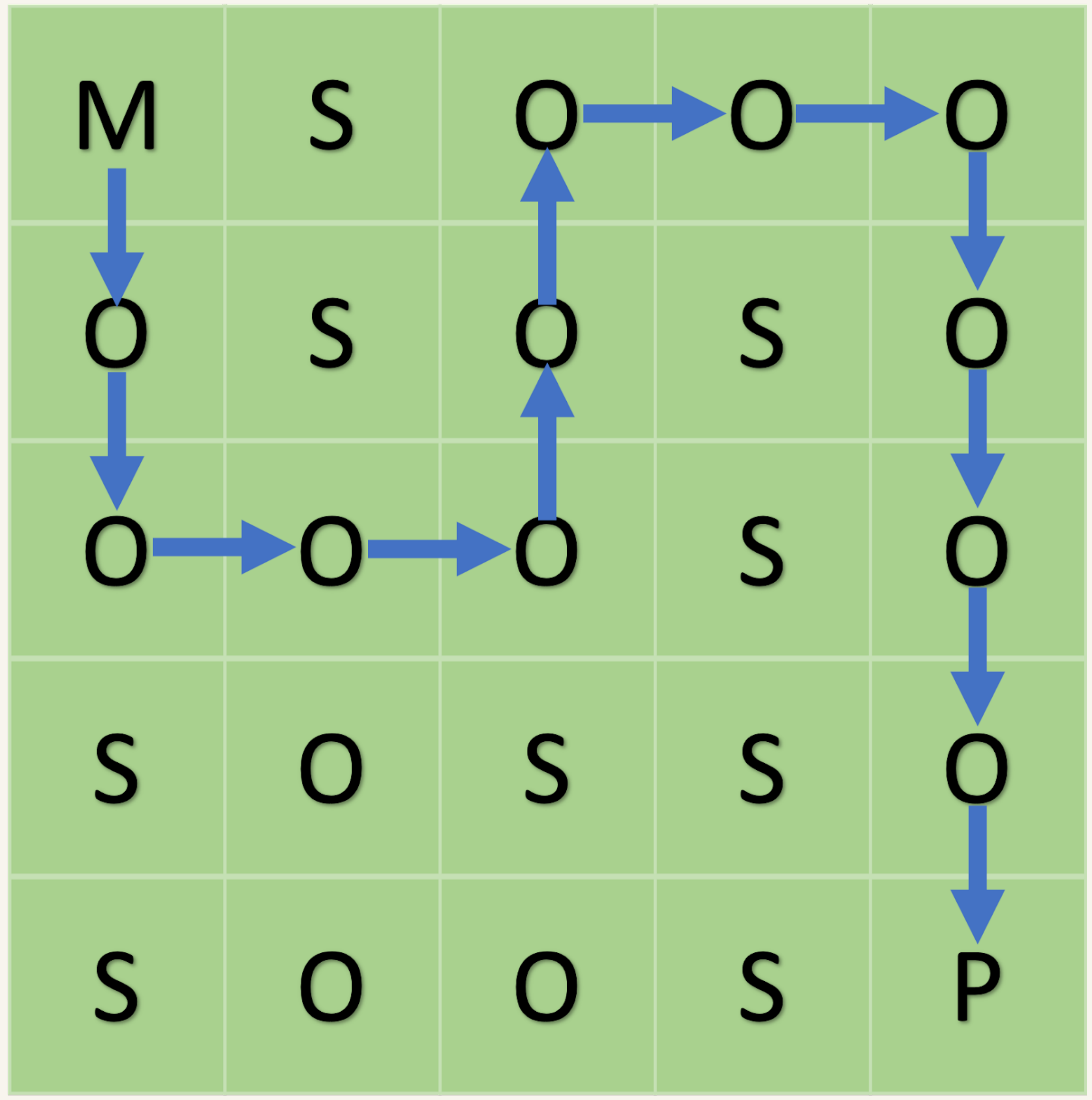


Solution: Try all 4! ways of placing the missing numbers, compute the score for each placement and take the best one.

P3 : Mario’s Perilous Path

- 'M': Mario's starting position
- 'P': Princess Peach's location
- 'S': Spikes obstacles Mario cannot pass through
- 'O': Unobstructed path

Mario can only move up/down/left/right on the grid.
Cannot move through spikes or step outside grid.
Problem: find min dist path from Mario to Peach.



Solution: Build a grid graph. For any cell marked “O”, connect it to adjacent cells marked “O”, but not if they are marked “S”. Run a breadth-first search (BFS) from the cell corresponding to M, and store the distance to each reachable cell from M.

$P4$: Princess Peach's Garden Party

Problem: Given an 8 vertex graph with M edges, determine:

- If there is a 4-set of vertices where all 4 vertices are pairwise connected (i.e., there is a 4-clique on these vertices)
- If there is a 3-set of vertices where there are no edges between any of the 3.

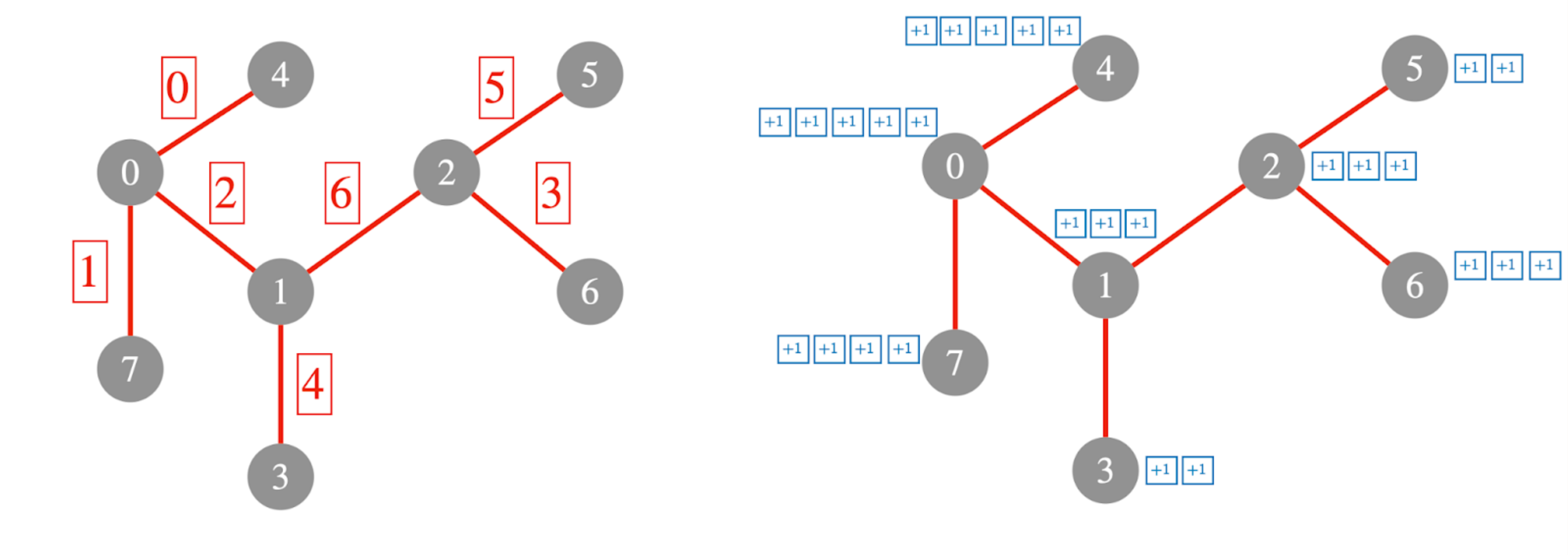
Solution:

- Check all sets of 4 vertices, and determine if they are fully connected. Otherwise return null for this part.
- Check all sets of 3 vertices, and determine if no edges exist among the 3. Otherwise, return null.

P5 : Maximizing Harmony

Problem:

Merge the edges in the tree in order of increasing weight, and determine the maximum number of merges "visible" to a vertex in the tree.



Solution:

Sort the edges by weight. Initially each vertex is in its own cluster.

Store the vertices in a Disjoint Set Union (DSU) data structure (i.e., Union-Find).

Merge each (u,v) edge by:

(1) finding the clusters corresponding to u, v: c(u) and c(v)

(2) merge c(u) into c(v)

(3) set harmony(c(v)) = max(harmony(c(u)), harmony(c(v)) + 1

Return the max harmony we see over the course of all merges.

P5 : Maximizing Harmony

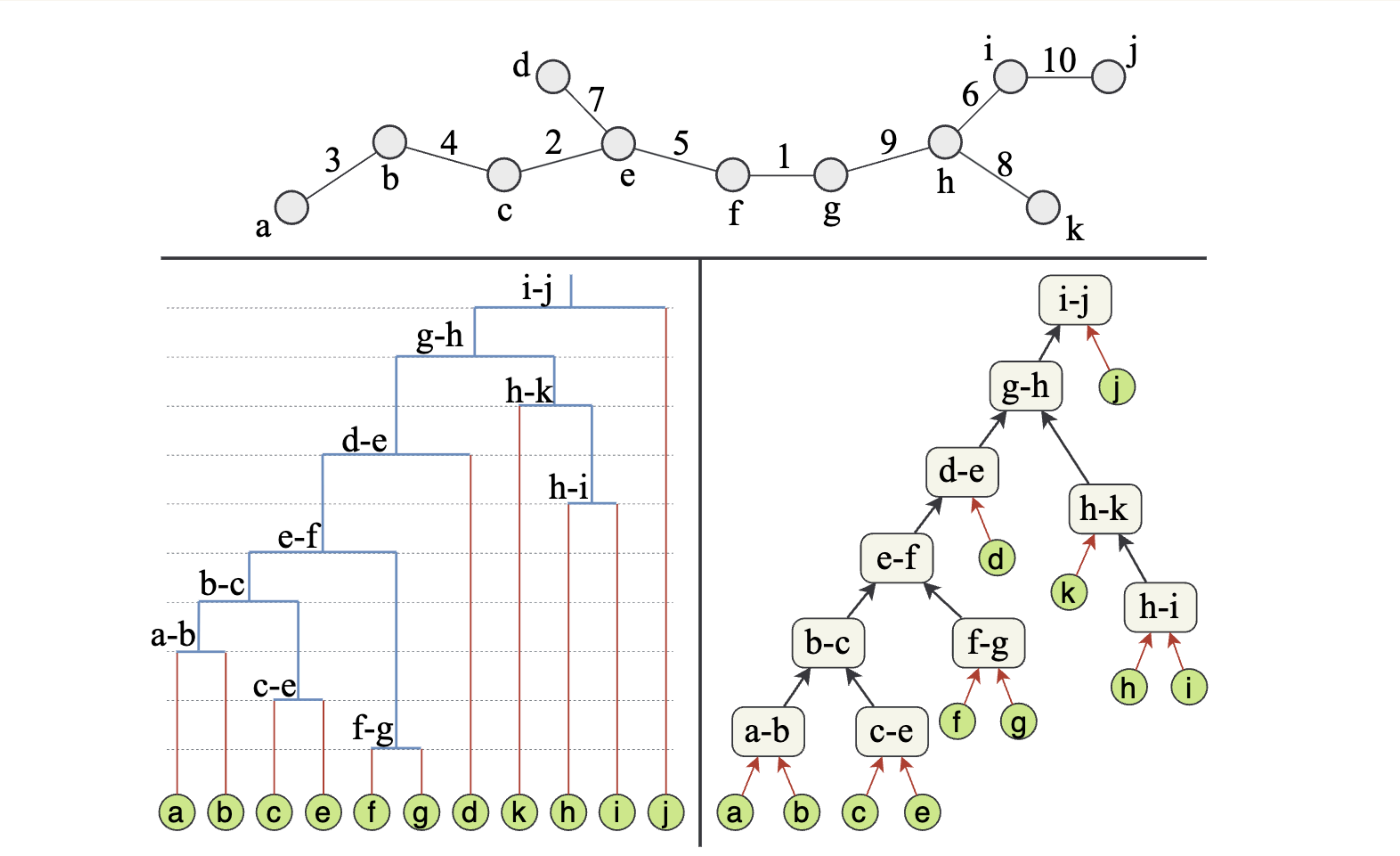
Problem:
Merge the edges in the tree in order of increasing weight, and determine the maximum number of merges "visible" to a vertex in the tree.

https://en.wikipedia.org/wiki/Single-linkage_clustering

Equal to the *height* of the single-linkage dendrogram.

Time complexity is $O(n \log n)$ due to sorting.

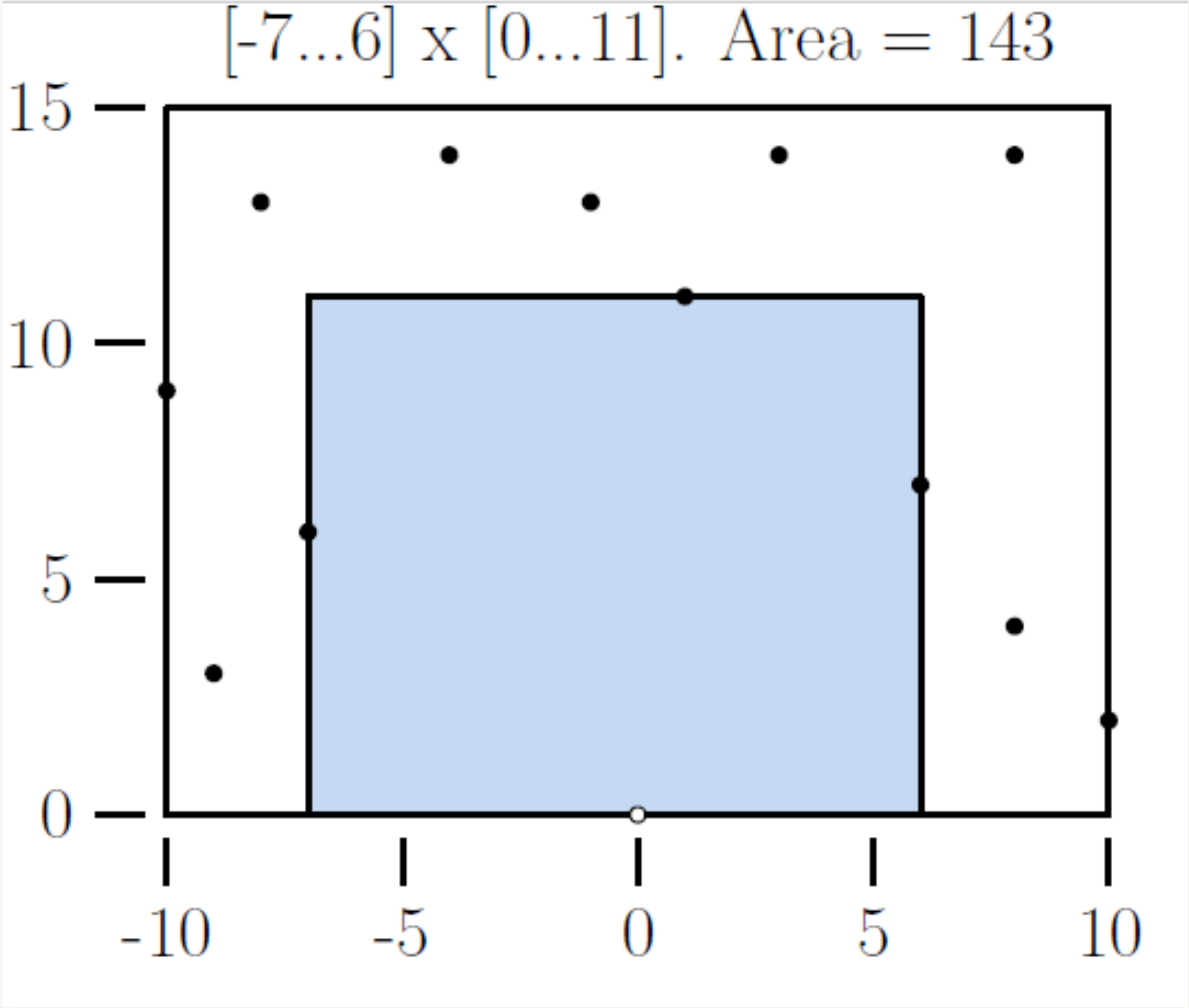
Turns out if the weights are pre-sorted, $O(n)$ is possible using decremental tree connectivity! See Demaine et al. *"On Cartesian trees and range minimum queries"*



$P6$: Magical Platform Planning for the Star Festival

Problem: Find the largest rectangle (by area) that contains the origin, but does not contain any flowers (points on the the 2D plane). The output rectangle should satisfy:

- -maxWidth <= xMin < 0 < xMax <= maxWidth
- 0 < yMax <= maxHeight



xMin = -7, xMax = 6, and yMax = 11

Solution:

- Key idea: the solution is determined by the flowers, since points (flowers) will be on the boundary.
- Use a *sweepline* technique:
  - sort the points by y-coordinate and sweep upward
  - maintain the largest xMin coordinate (left wall) and the smallest xMax coordinate (right wall) with height smaller than the current y coordinate
  - if the current point's x coordinate is "inside" the current rectangle, shrink the left (or right) wall for the next iteration.
- Time complexity: $O(n \log n)$ (for sorting by y)

$P7$: Chompy Chain Clusters

Problem: Vertices are on the 2D grid. Some of the grid cells have "Chompy Chains" that grow clusters starting at those locations. Clusters start at their *start time* and grow at unit rate. If a cluster is already covered when it is supposed to start it doesn't grow. The first cluster that reaches a vertex captures it in its cluster. The problem is to output the **final size of each cluster.**

Solution:

- Build a grid graph on W^2 vertices with unit-weight edges.
- Create a super-source src, and connect to each chain vertex with weight equal to the start time of the chain
- Run Dijkstra, and keep track of the last cluster that reduced the distance to a vertex. When we process a vertex, increase the size of the cluster corresponding to the last visitor.

Cost is $O(W^2 \log W)$ to run Dijkstra. With a little care, $O(W^2)$ is possible.

Related to the *low-diameter decomposition problem*—a very useful primitive for designing fast parallel and distributed graph algorithms: see

[Submitted on 14 Jul 2013]

Parallel Graph Decompositions Using Random Shifts

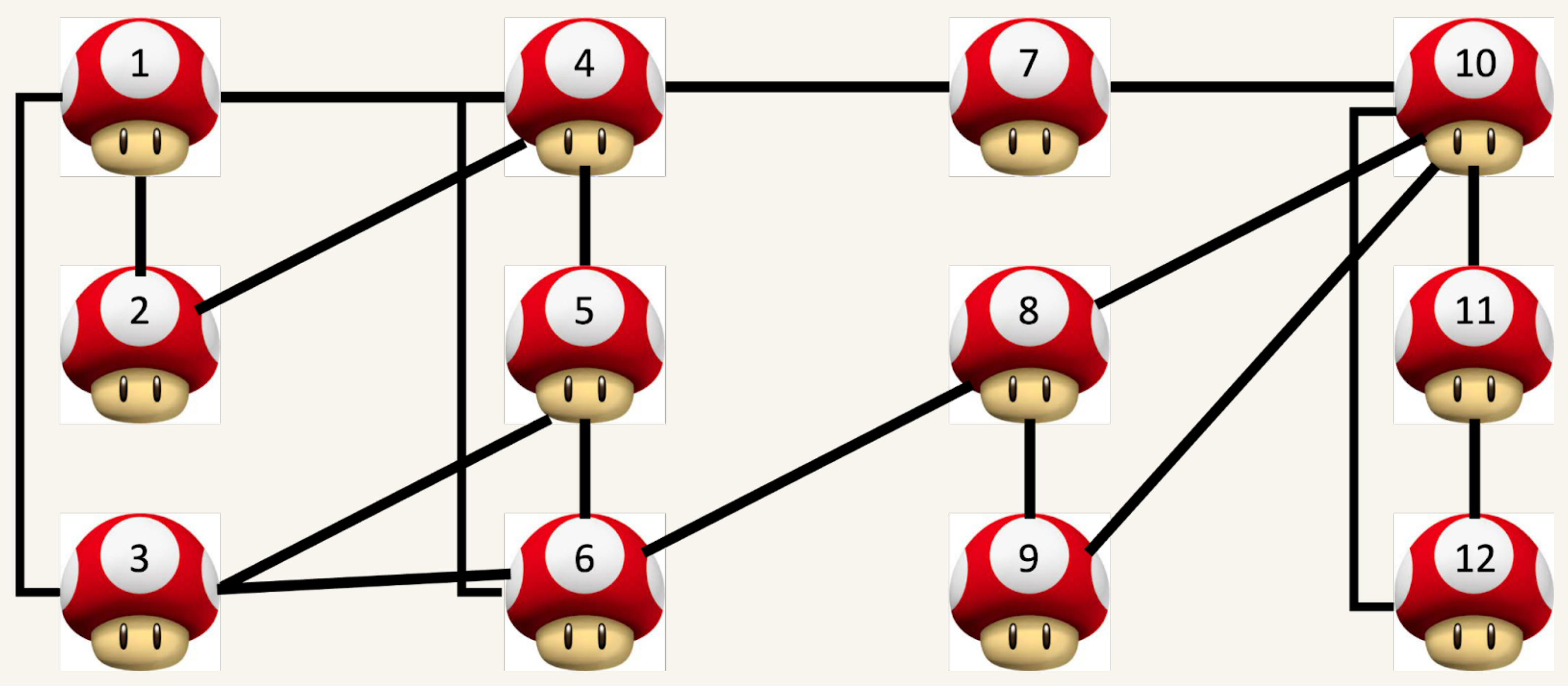
Gary L. Miller, Richard Peng, Shen Chen Xu

$P8$: Colorful Chaos in the Mushroom Kingdom!

Problem: Given a $3 \times W$ size grid (each column has 3 vertices). There can only be edges within a column, or between adjacent columns. The problem is to color the vertices using as few colors as possible (it is always possible to 6-color the vertices).

Solution:

- Fix the number of colors $C < 6$.
- Use dynamic programming. Check whether the i-th group (of the W groups) can be colored using some colors c_1, c_2, c_3 where each $c_i \leq C$.
  - ★ Check if the colors are valid within the group
  - ★ Check all valid possibilities for the previous group, and check if the coloring is consistent.
- Return true (for this guess for C) if and only if the last column can be successfully colored.



Cost (theoretically) is $O(W)$, but the constant factors are pretty large :-)

Problem 9: Balancing Parentheses

- Couple of observations:
  - Only swap consecutive ")("

    Instead of: ")))))(" → "()))))"

    Do: ")))))(" → "))))()" → ")))())" → … → "()))))"
  - Convert the string to an array of numbers:

    Total Sum: 2

) ( ) ( ) (( ) ( ) ( ) ( ) ( )

    -1 0 -1 0 -1 0 1 0 1 0 1 0 1 0 1 0
  - Every swap basically takes a "X X-1 X" triple and makes it "X X+1 X"

) (**)** **(**) (( ) ( ) (**)** **(**) ( )

    -1 **0** **-1** **0** -1 0 1 0 1 0 **1** **0** **1** 0 1 0

) (**(** **)**) (( ) ( ) (**(** **)**) ( )

    -1 **0** **1** **0** -1 0 1 0 1 0 **1** **2** **1** 0 1 0

    Total Sum: 6 → 2 moves
  - Can make moves until:

) (((((((( )))))))

    -1 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0

    Total Sum: 48 → (48-2)/2 = 23 moves