Dinic’s Algorithm

Shusen Wang

Comparisons

- m: the number of edges.
- n: the number of vertices.
- Time complexity of Edmonds–Karp algorithm [1] is $O(m^2 \cdot n)$.

Reference

1. Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM.* 19 (2): 248–264, 1972.

Comparisons

- m: the number of edges.
- n: the number of vertices.
- Time complexity of Edmonds–Karp algorithm [1] is $O(m^2 \cdot n)$.
- Time complexity of Dinic’s algorithm [2] is $O(m \cdot n^2)$. (Faster because m is bigger than n.)

Reference

1. Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM.* 19 (2): 248–264, 1972.
2. Yefim Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Proceedings of the USSR Academy of Sciences*, 11: 1277–1280, 1970.

Dinic's Algorithm

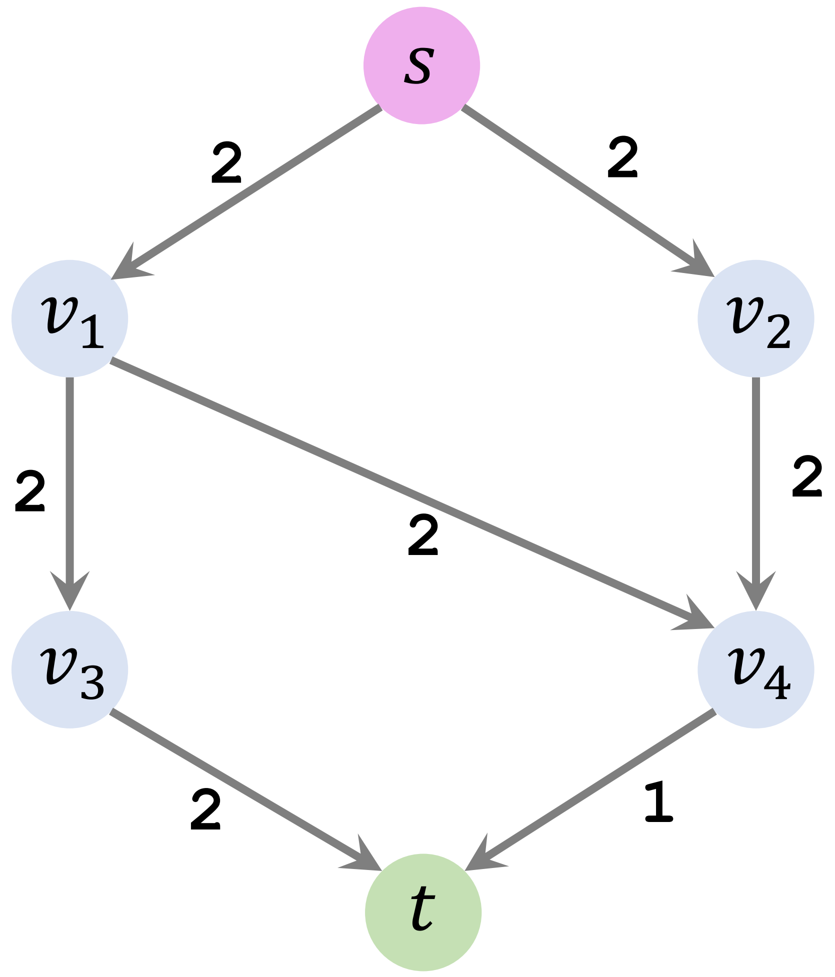
- Yefim Dinitz published "Dinitz's Algorithm" in USSR, 1970 [1].
- "Dinitz's Algorithm" was introduced to the westerners by Even & Tarjan's 1975 paper [2].
- Even & Tarjan misspelled Dinitz's name as "Dinic" [2].

Reference

1. Yefim Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Proceedings of the USSR Academy of Sciences*, 11: 1277–1280, 1970.
2. Shimon Even and R. Endre Tarjan. Network Flow and Testing Graph Connectivity. *SIAM Journal on Computing*, 4 (4): 507–518, 1975.

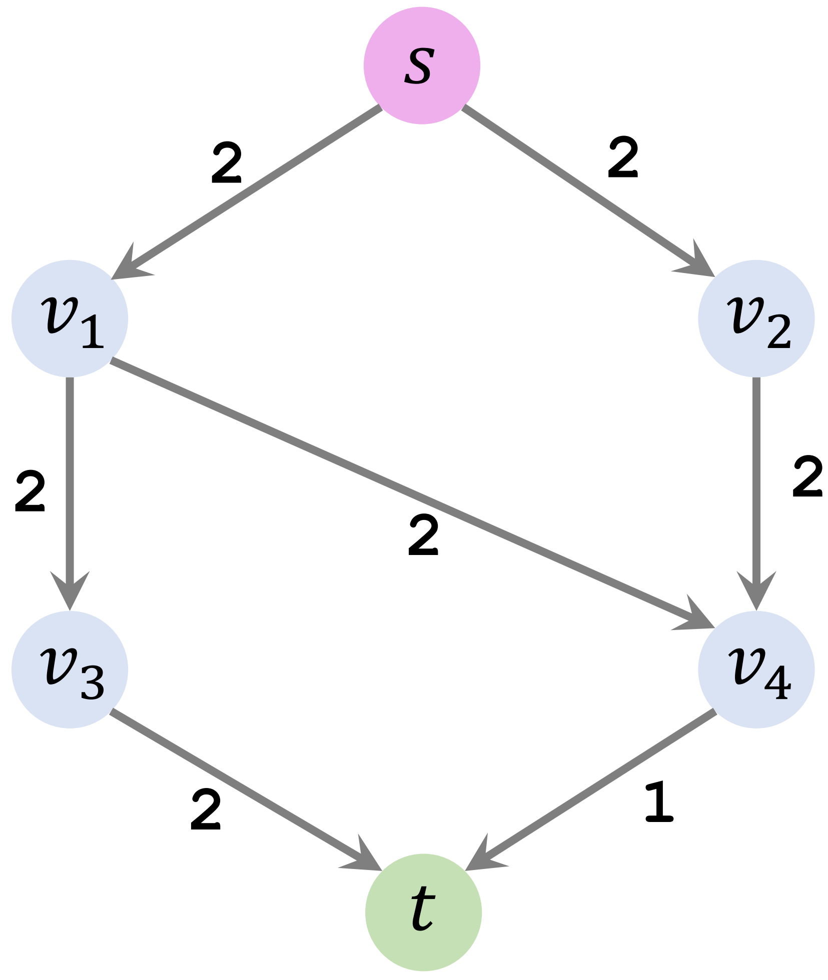
Key Concept: Blocking Flow

Blocking Flow

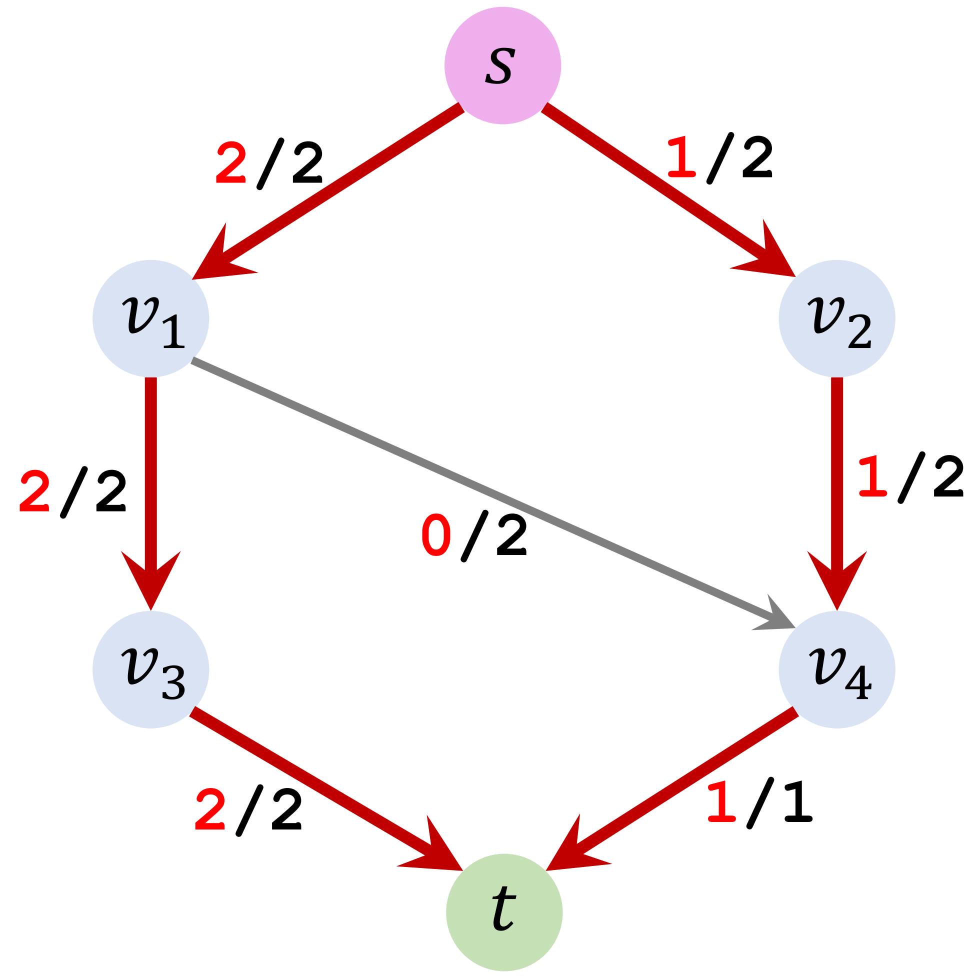


- A flow is **blocking flow** if no more flow from source to sink can be found.
- Max flow is blocking flow; blocking flow may not be max flow.
- Blocking flow can be found by the naïve algorithm.

Blocking Flow

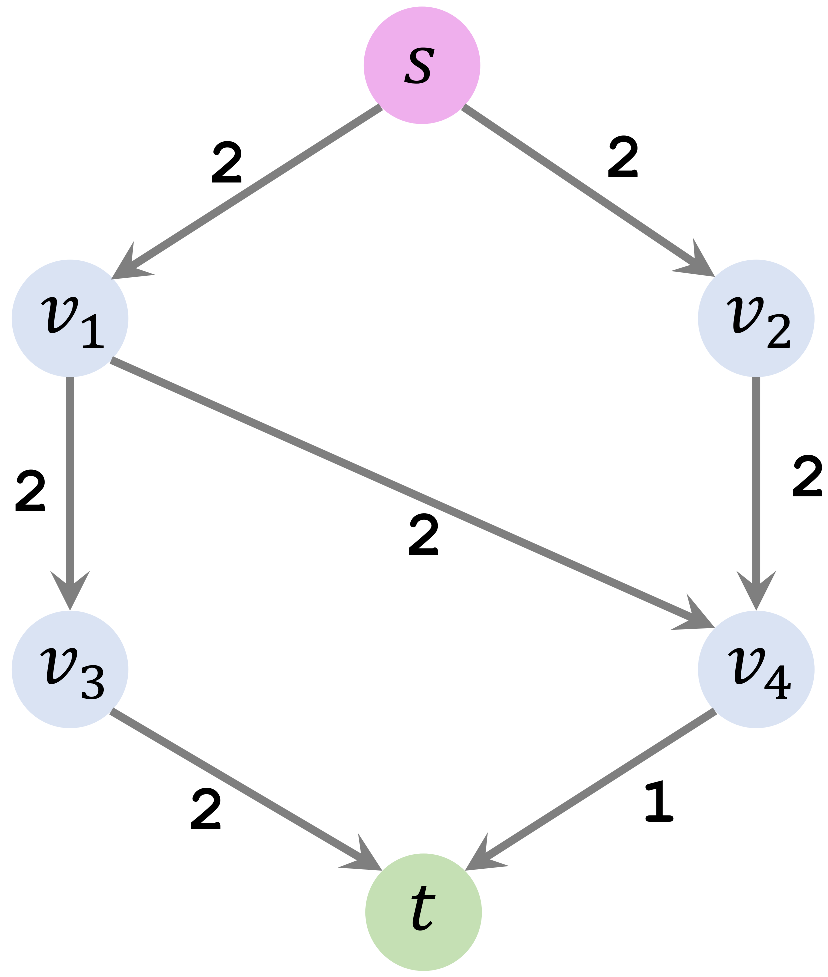


Original Graph

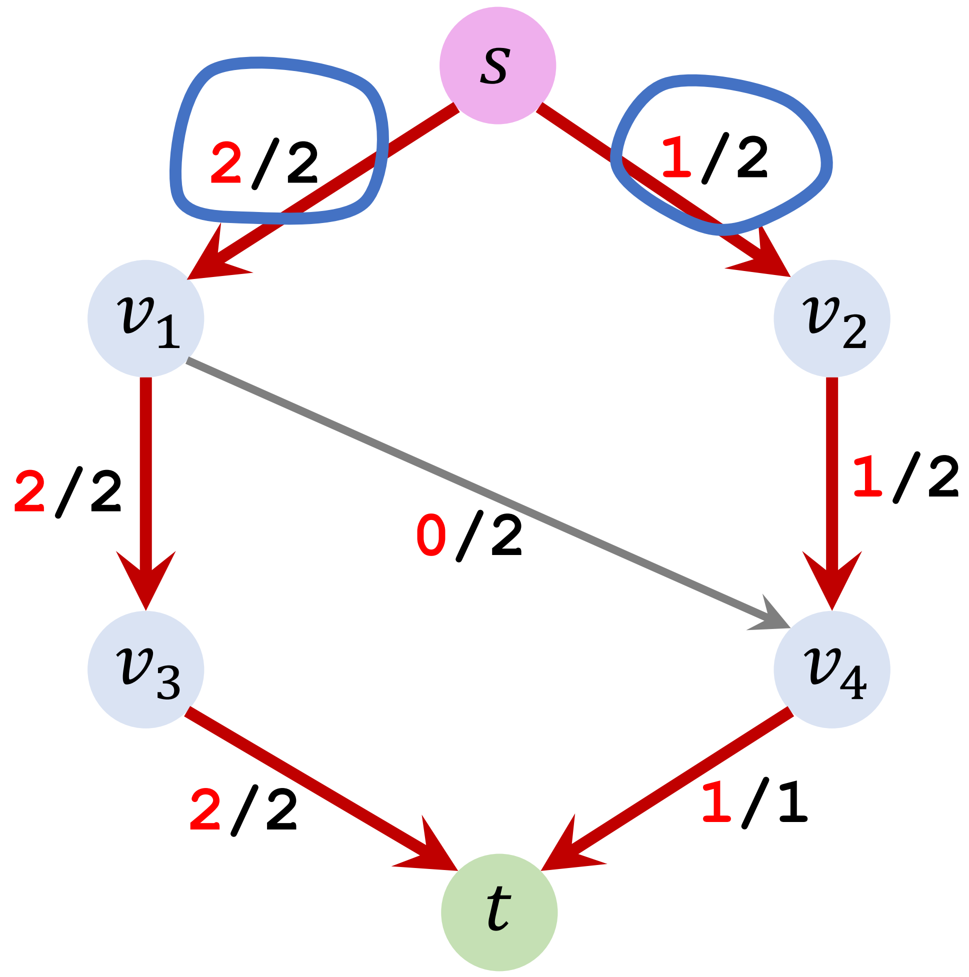


A Blocking Flow.

Blocking Flow

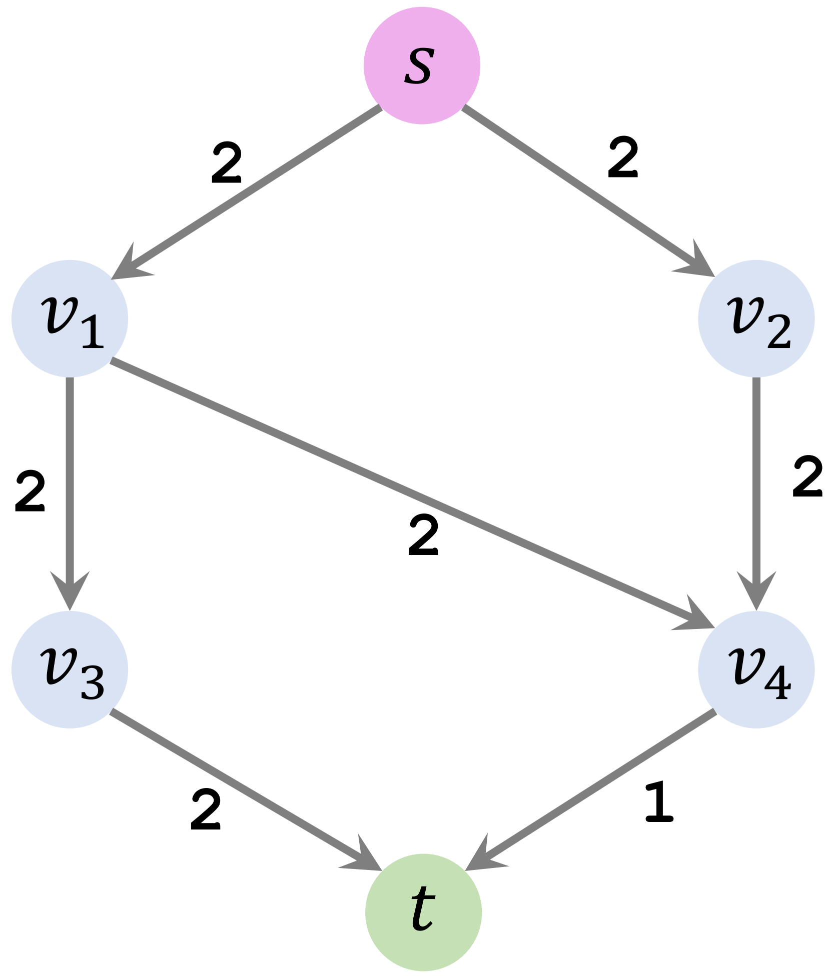


Original Graph

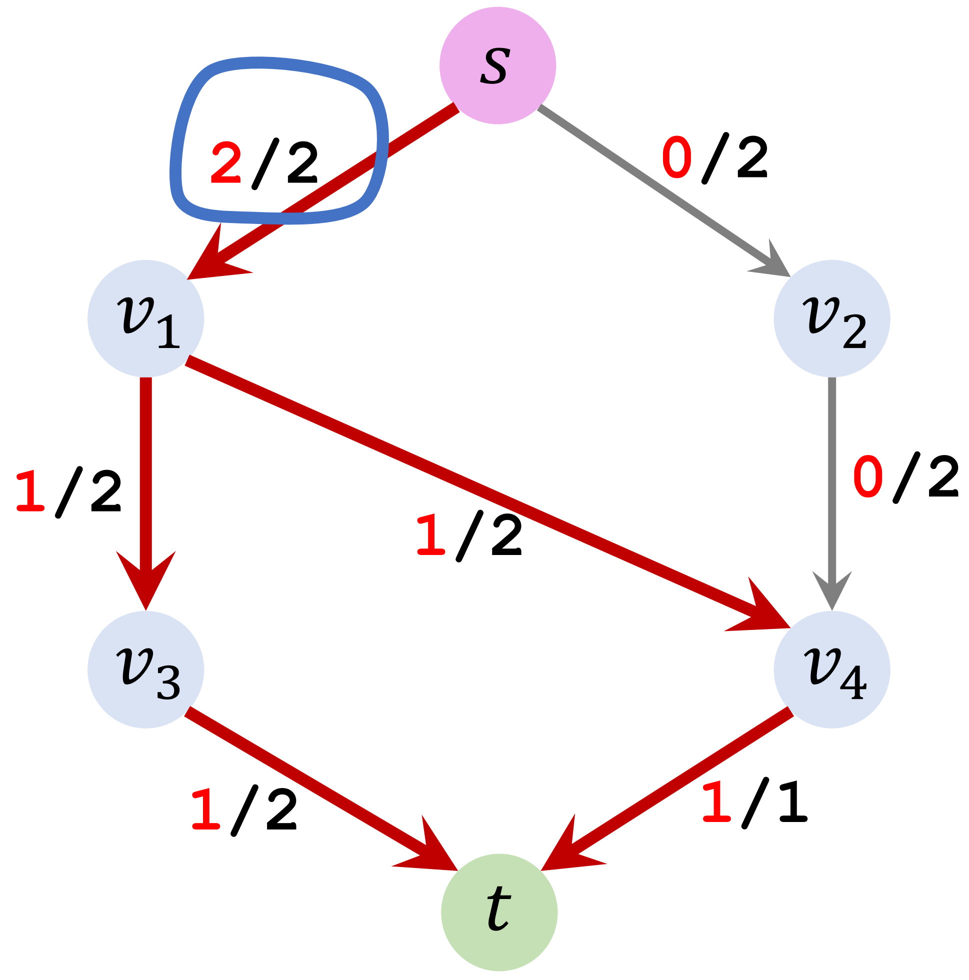


A Blocking Flow.

Blocking Flow



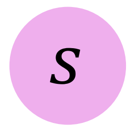
Original Graph

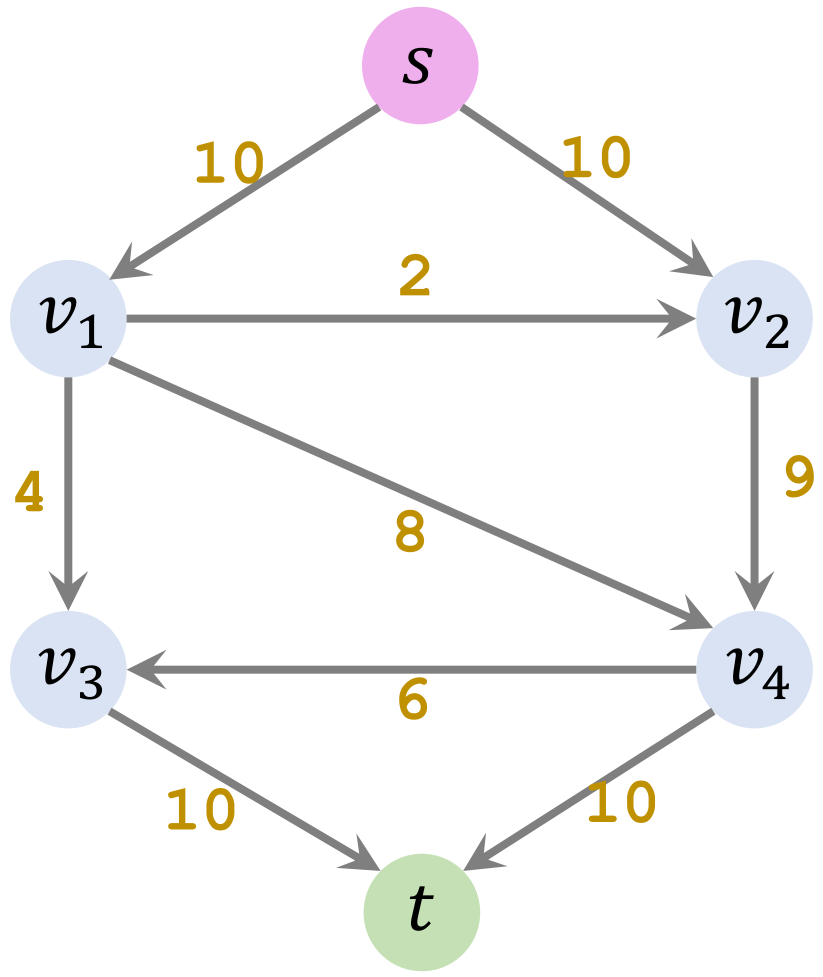


Another Blocking Flow.

Key Concept: Level Graph

Level Graph: Example 1

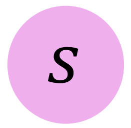




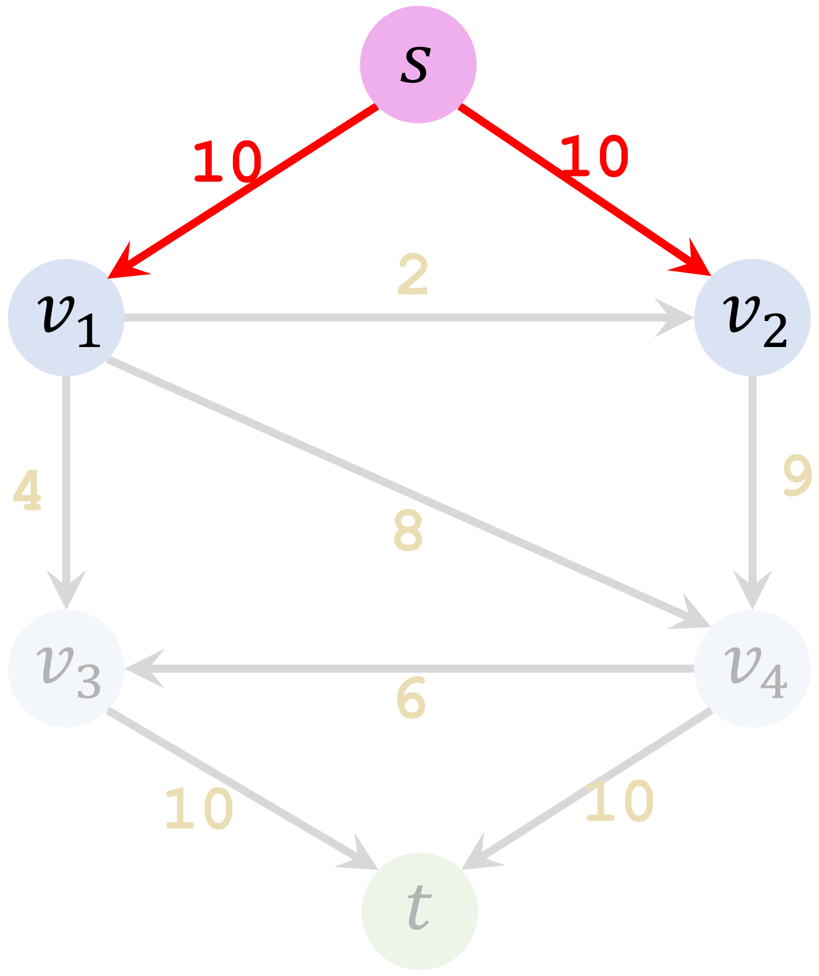
Level Graph

Original Graph

Level Graph: Example 1

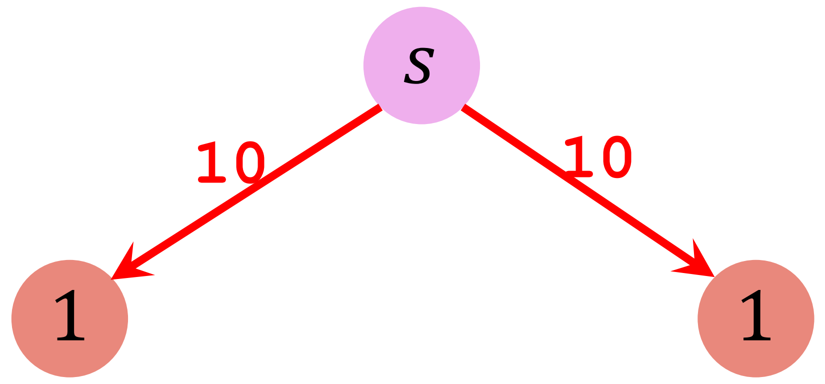


Level Graph

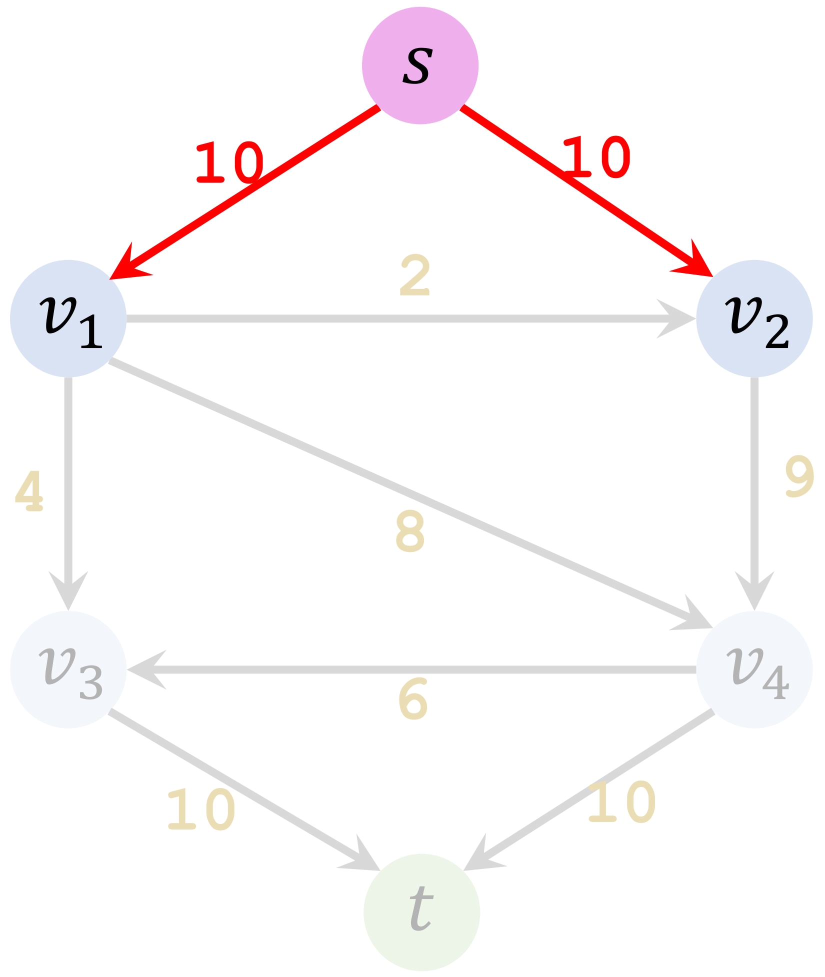


Original Graph

Level Graph: Example 1

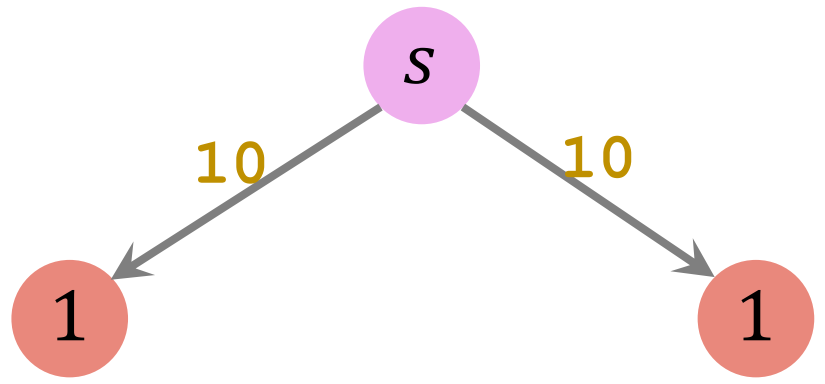


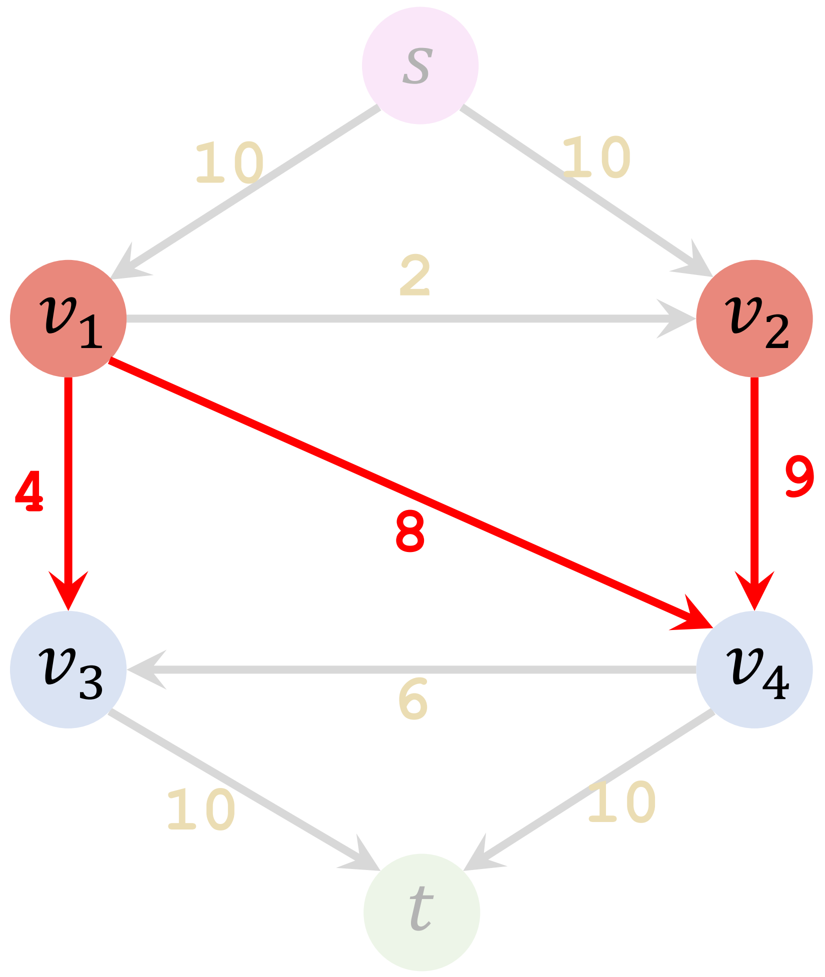
Level Graph



Original Graph

Level Graph: Example 1

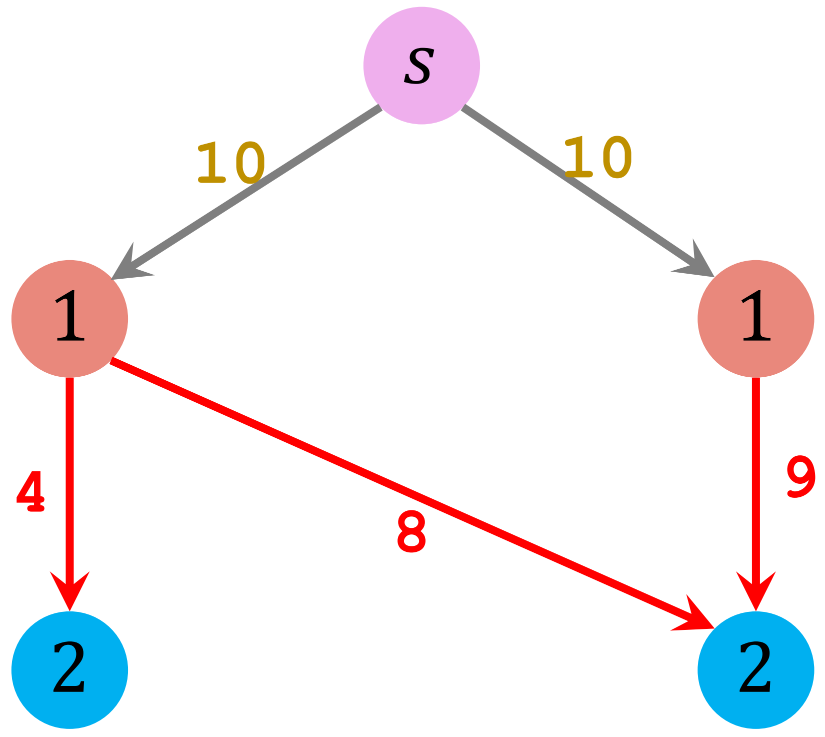




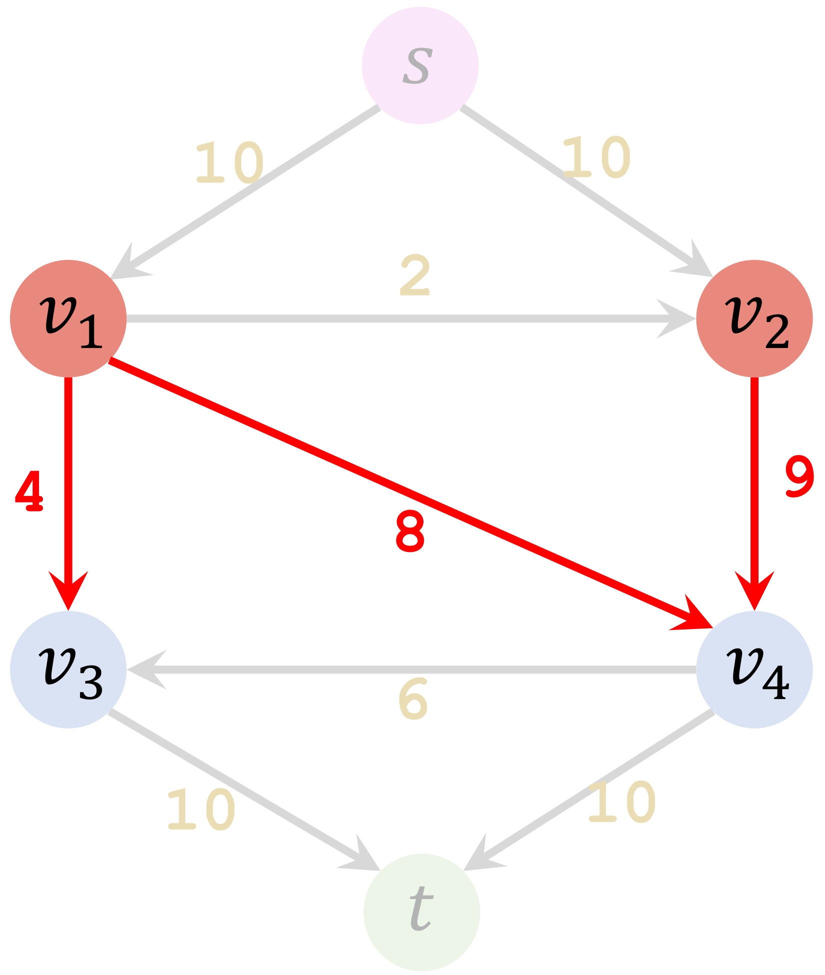
Level Graph

Original Graph

Level Graph: Example 1

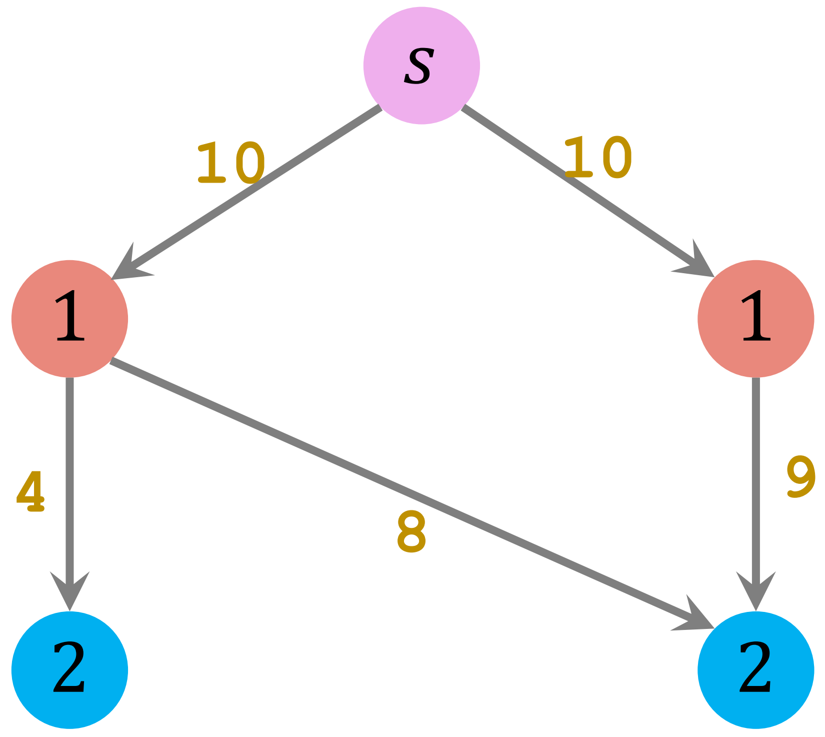


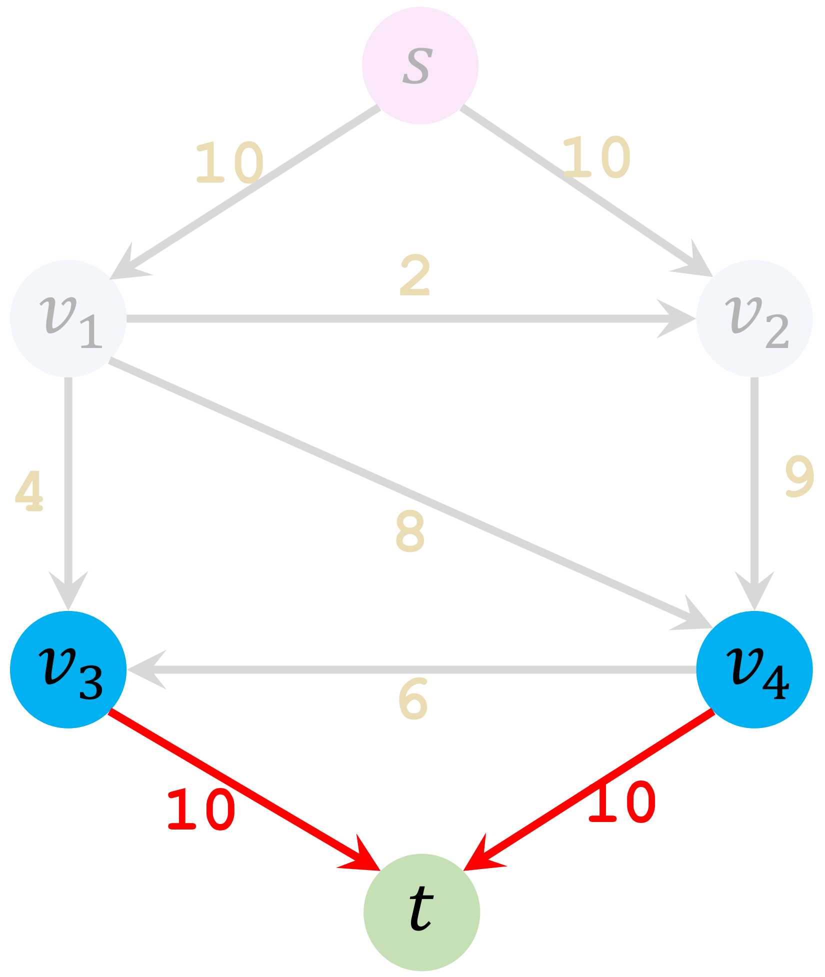
Level Graph



Original Graph

Level Graph: Example 1

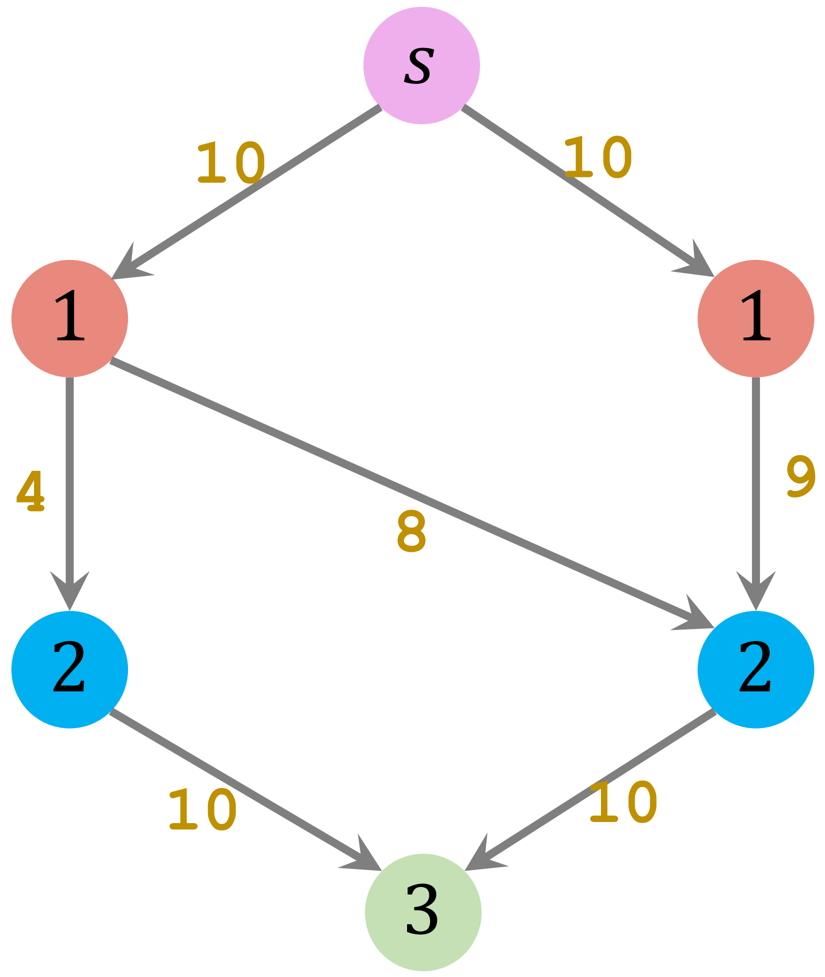




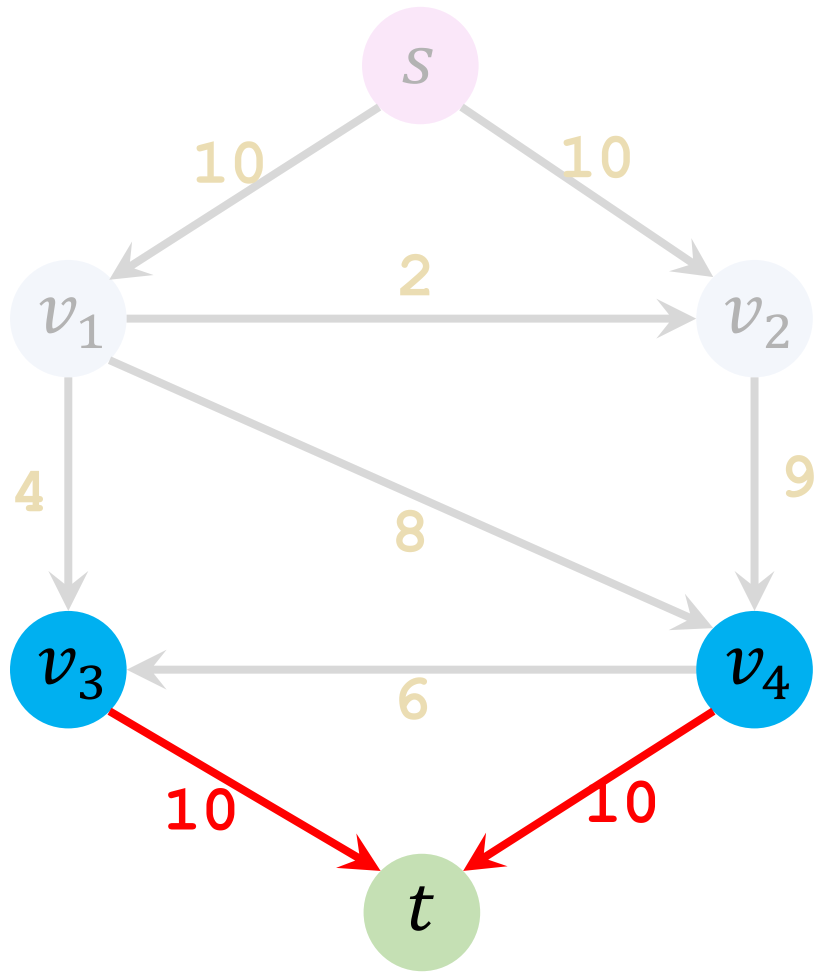
Level Graph

Original Graph

Level Graph: Example 1

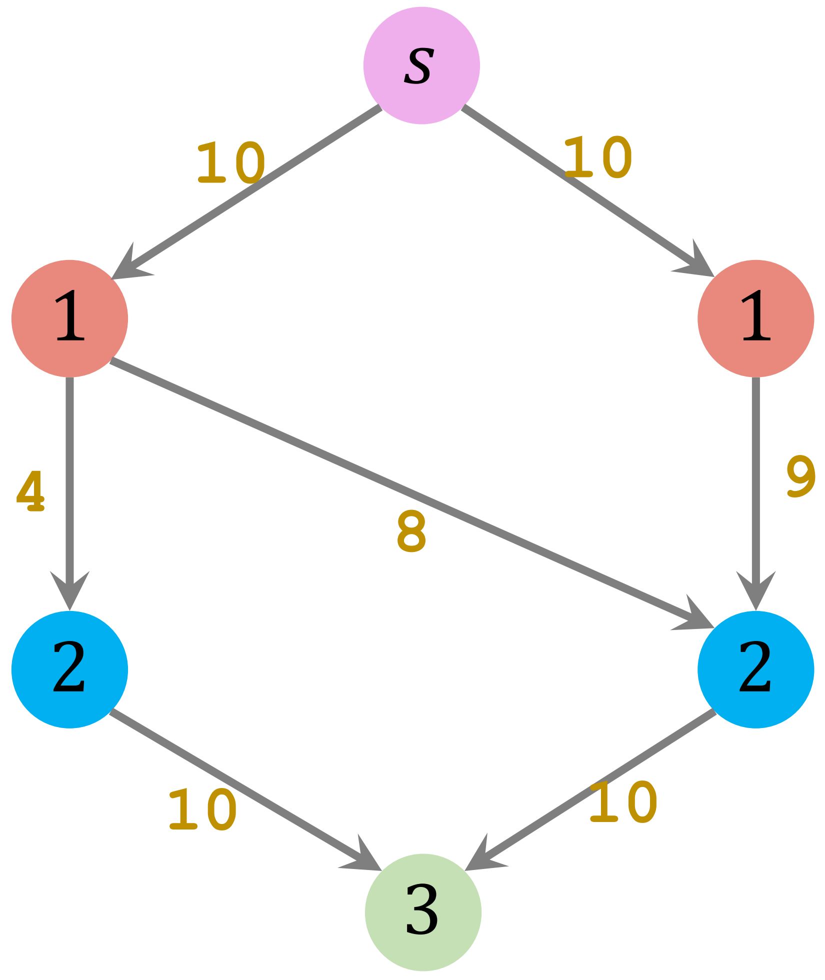


Level Graph

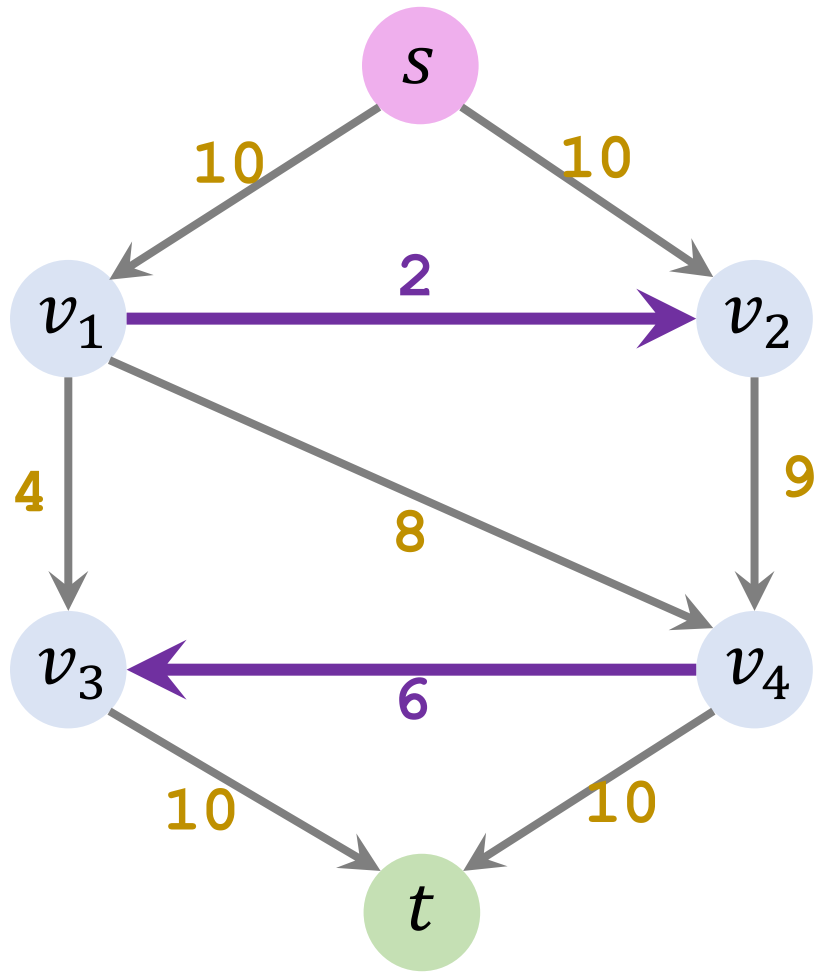


Original Graph

Level Graph: Example 1

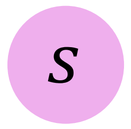


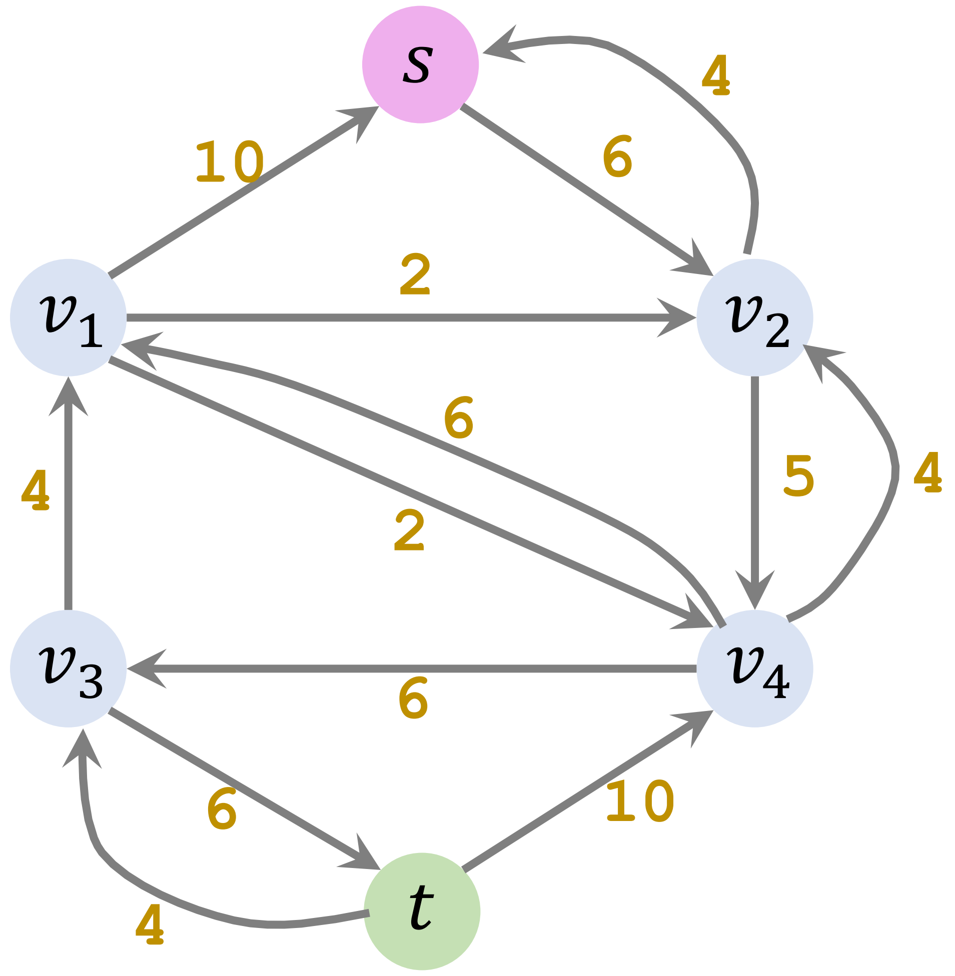
Level Graph



Original Graph

Level Graph: Example 2

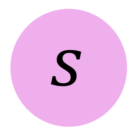




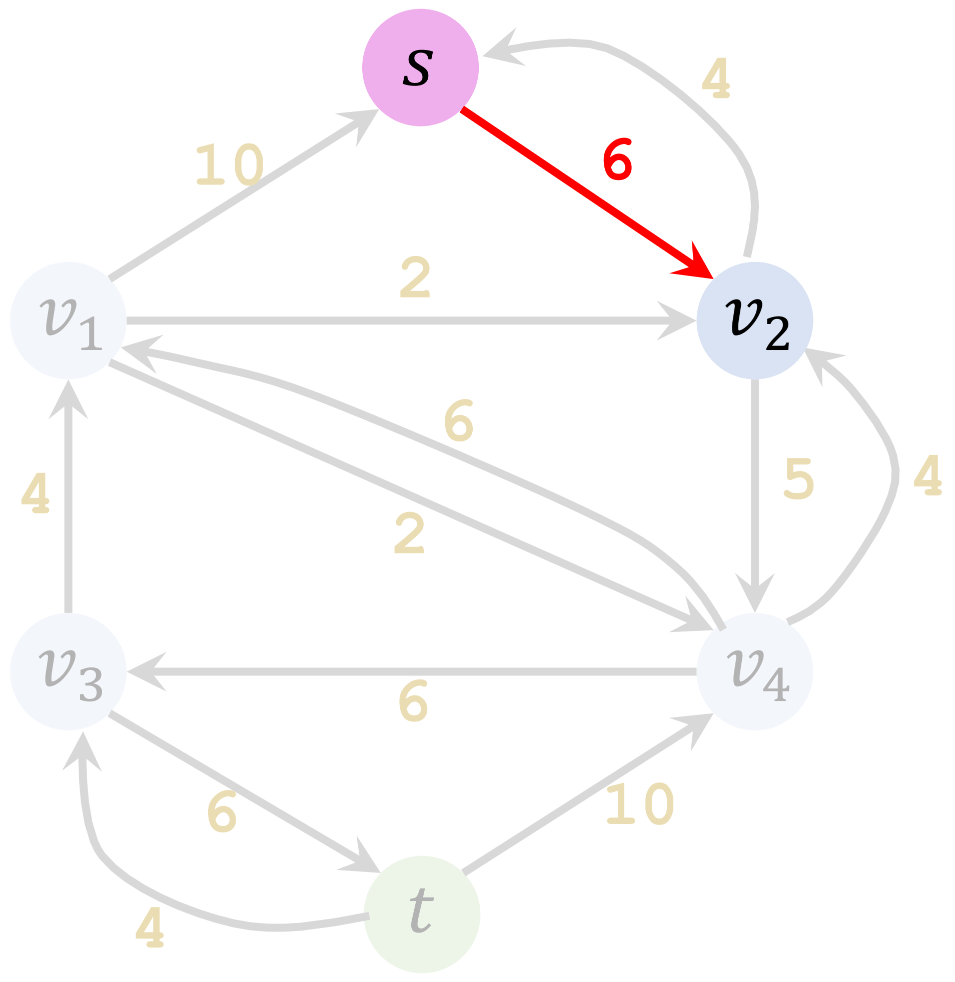
Level Graph

Original Graph

Level Graph: Example 2

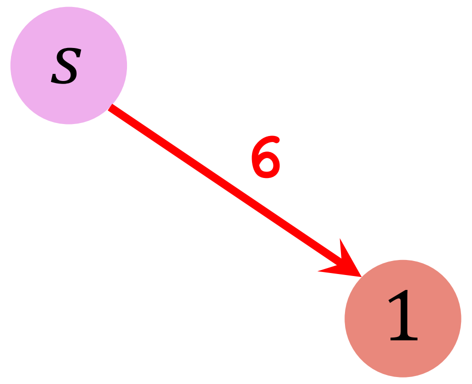


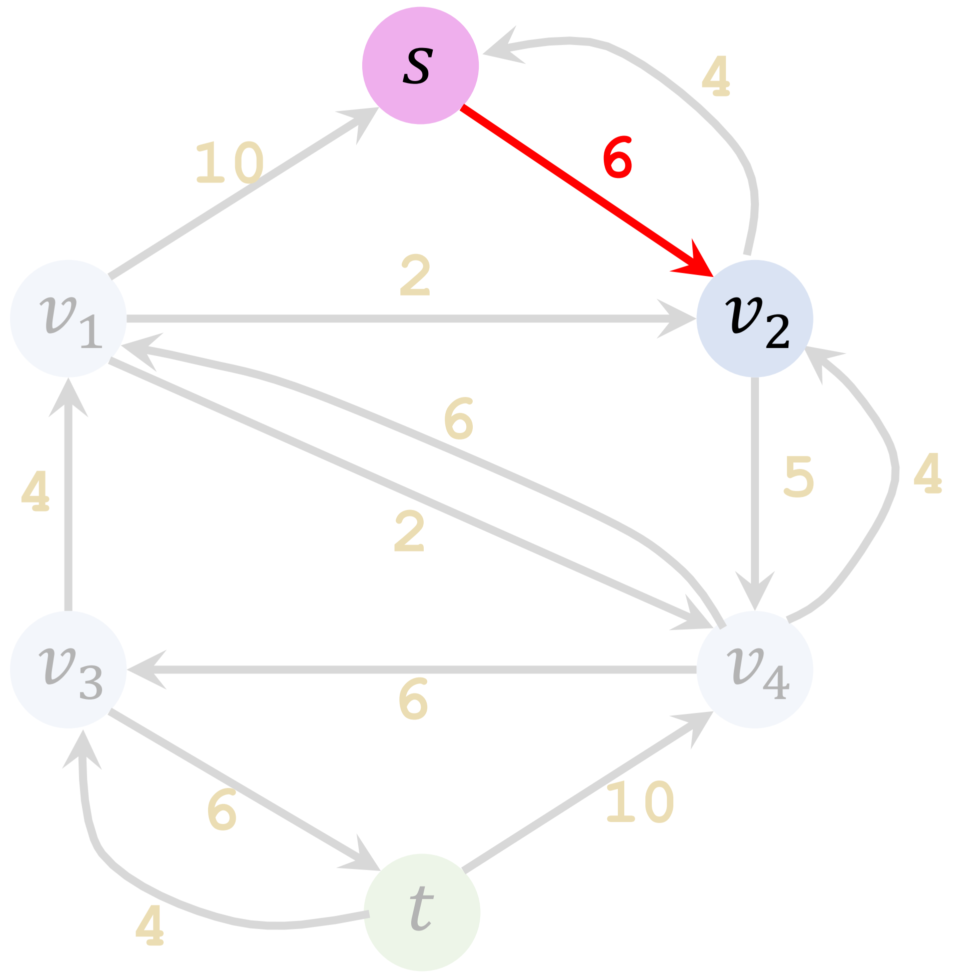
Level Graph



Original Graph

Level Graph: Example 2

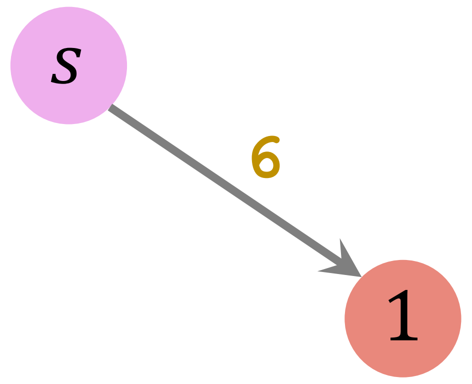


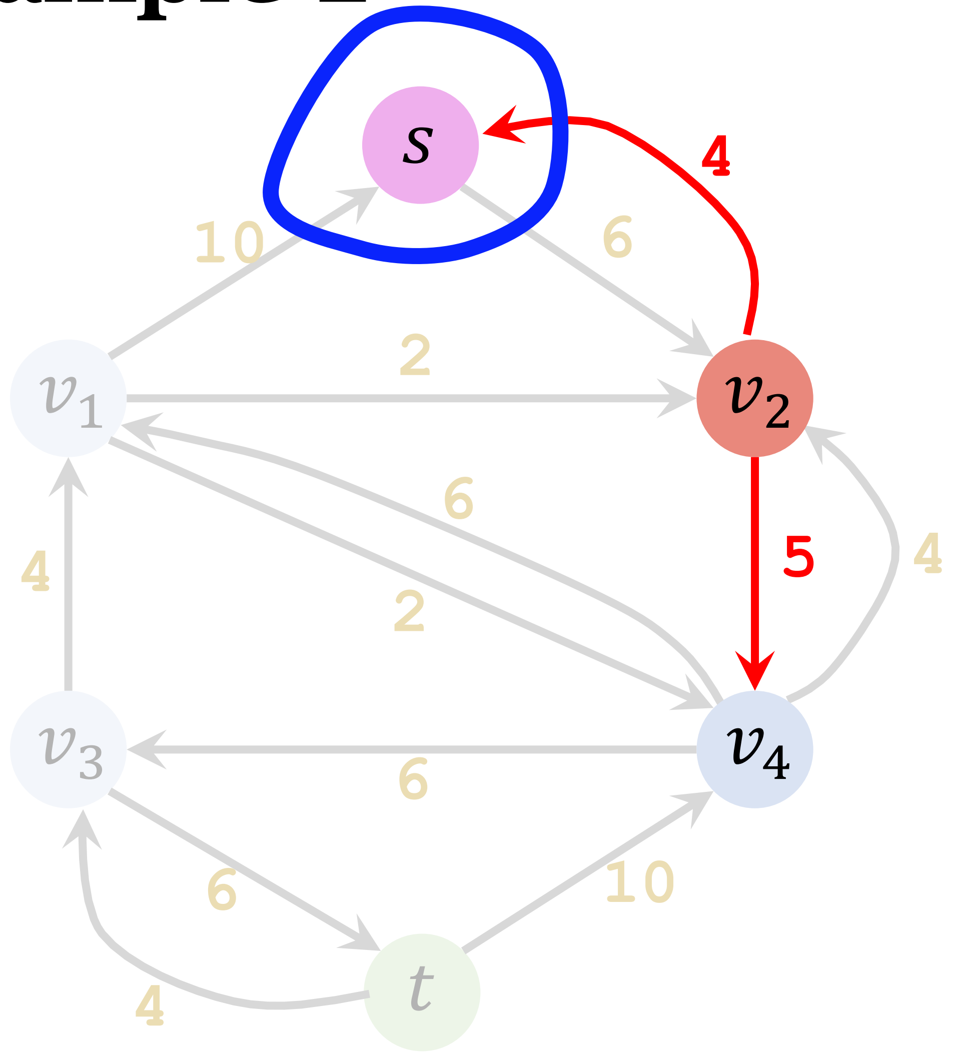


Level Graph

Original Graph

Level Graph: Example 2

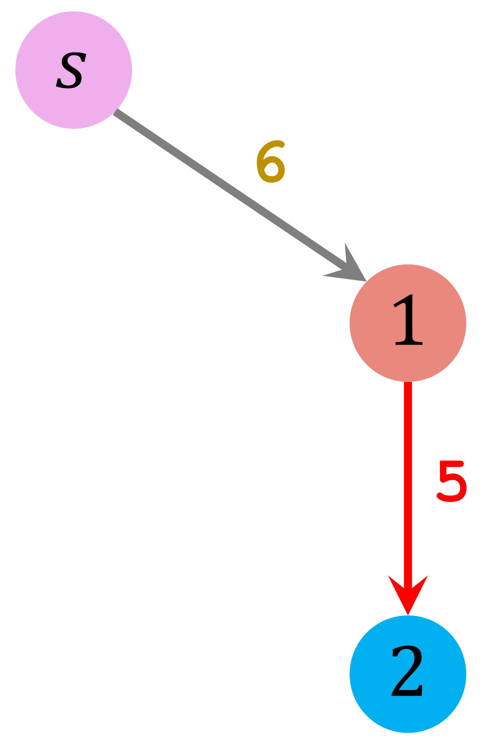




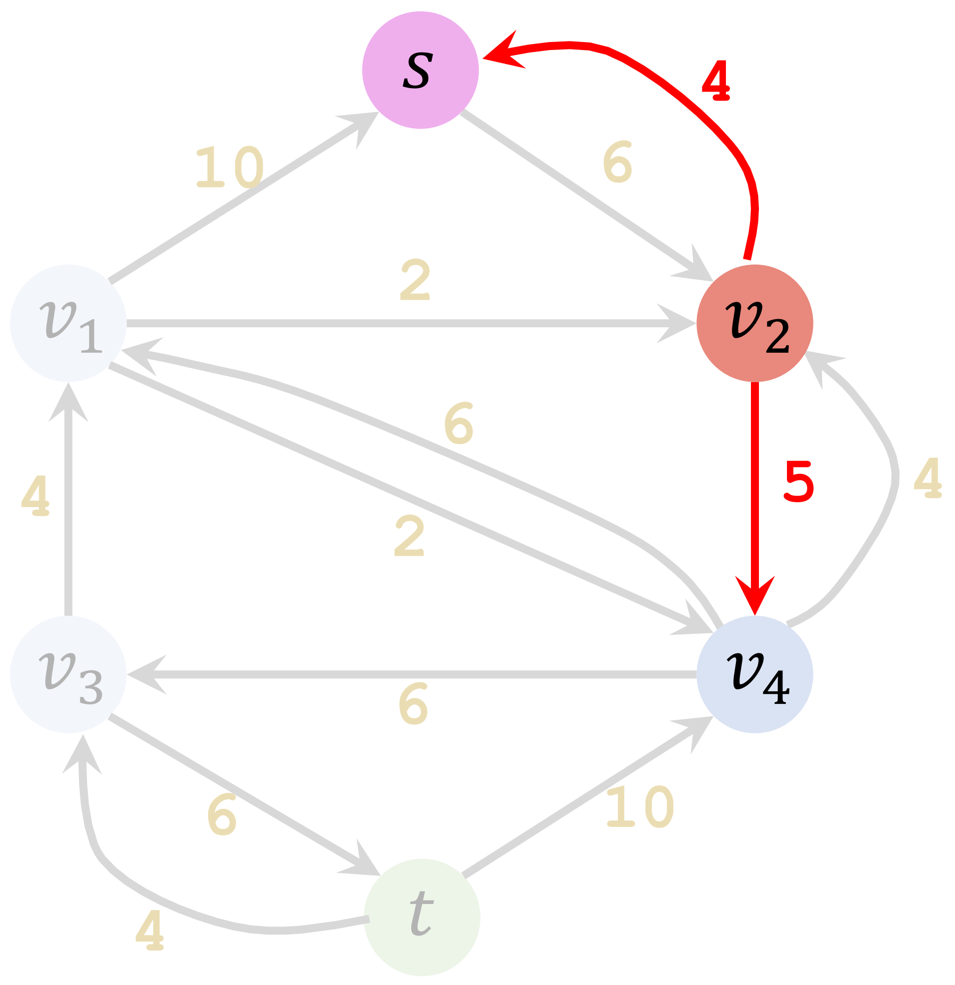
Level Graph

Original Graph

Level Graph: Example 2

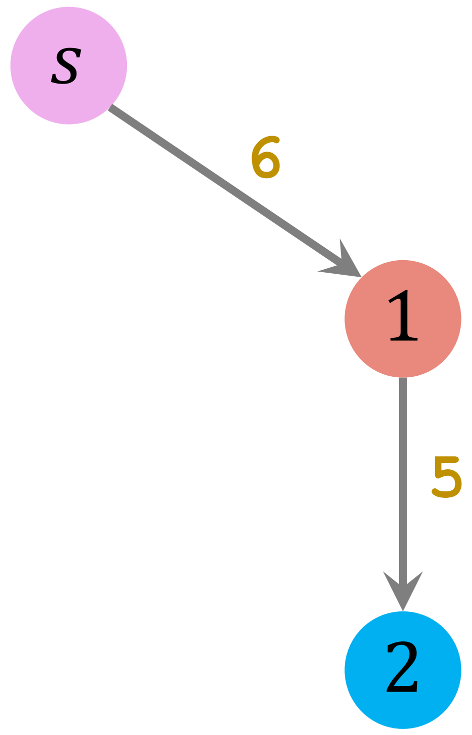


Level Graph

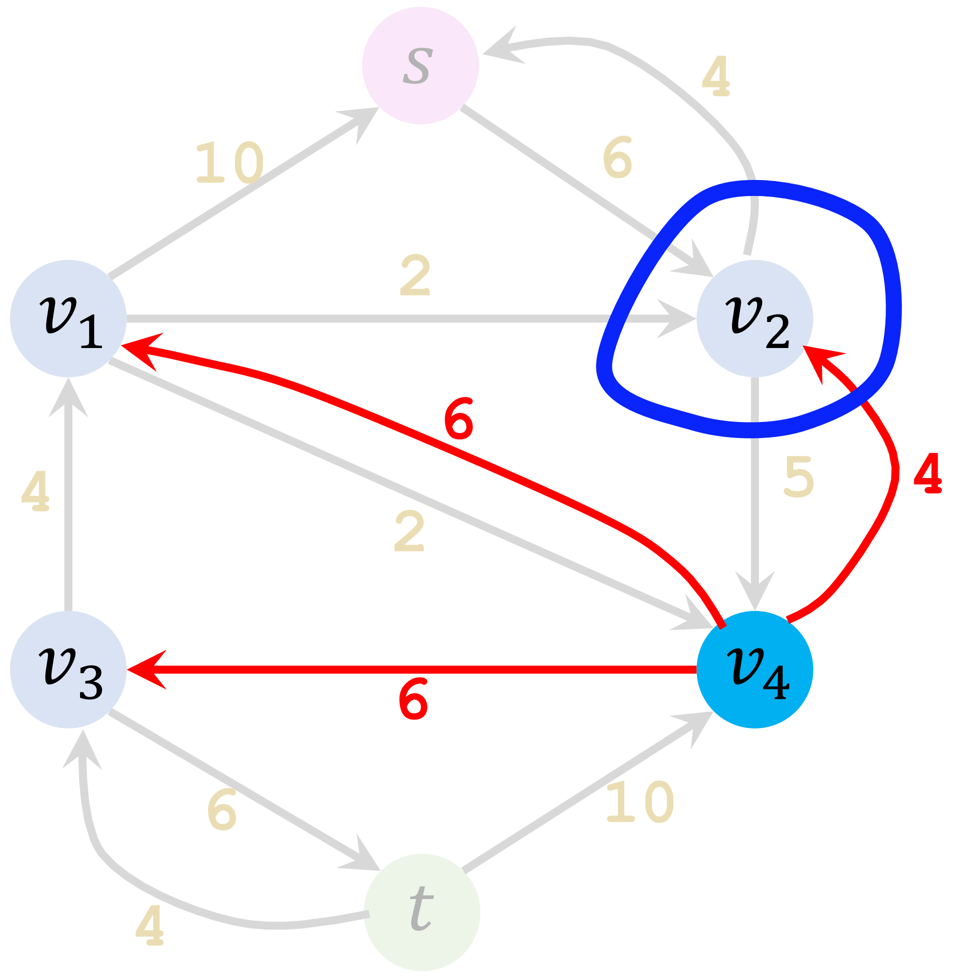


Original Graph

Level Graph: Example 2

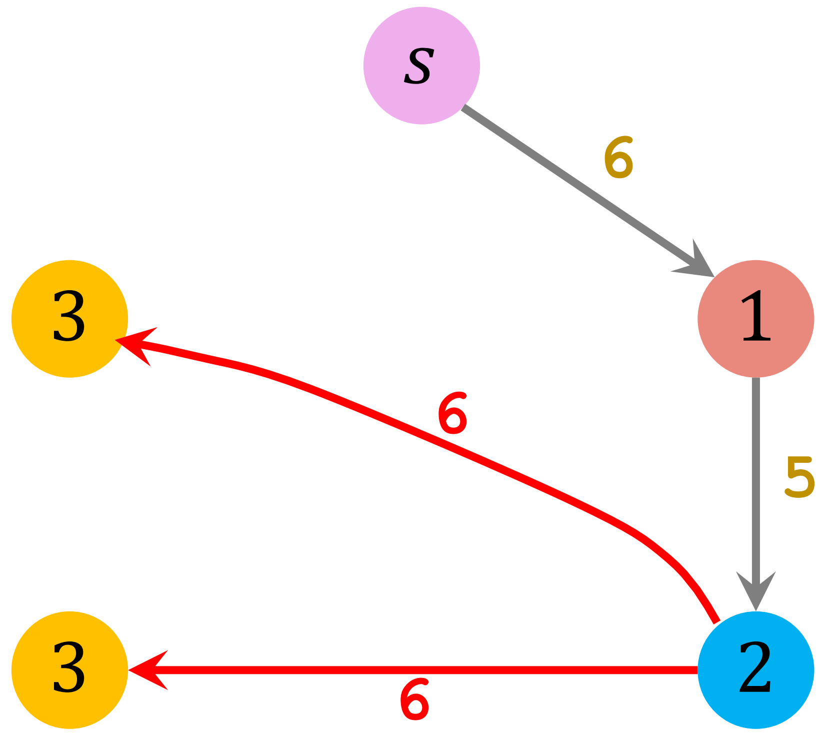


Level Graph

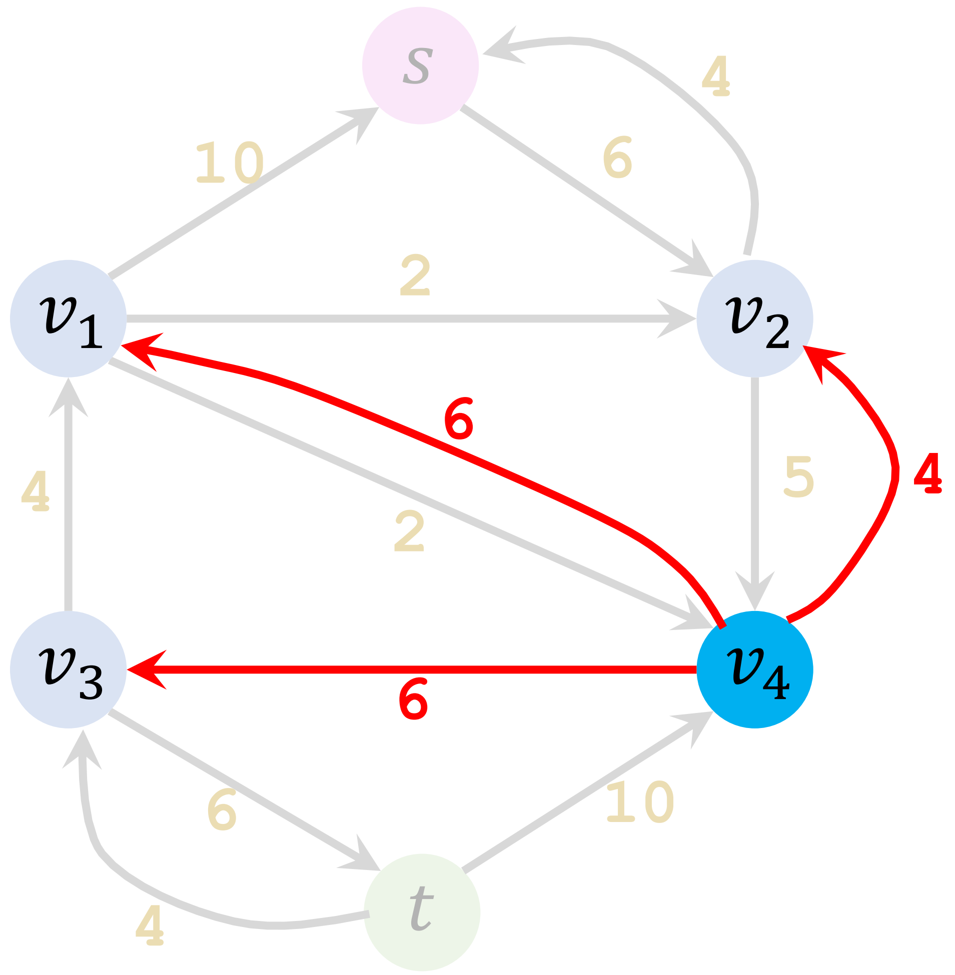


Original Graph

Level Graph: Example 2

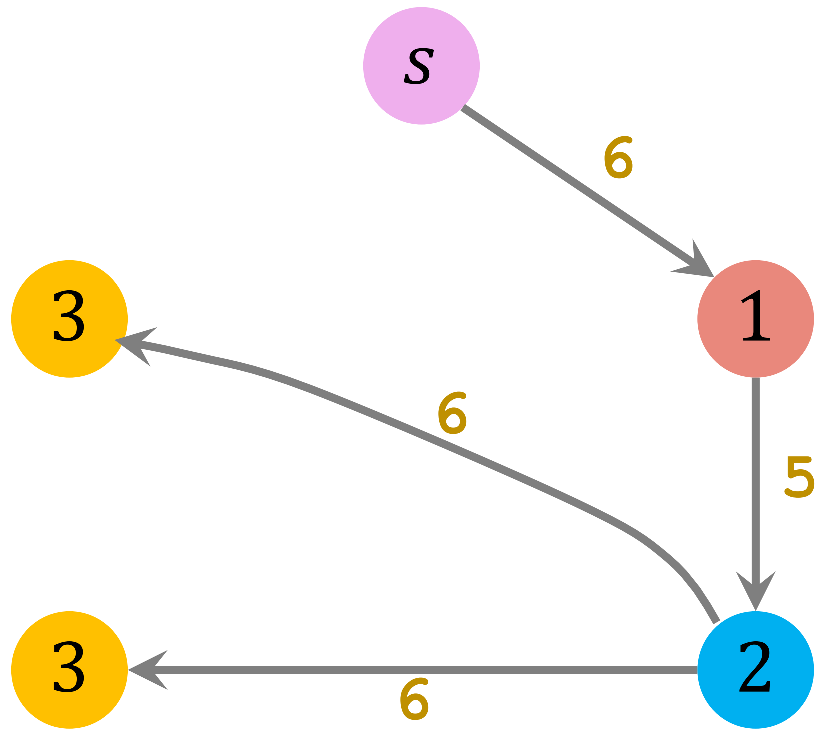


Level Graph

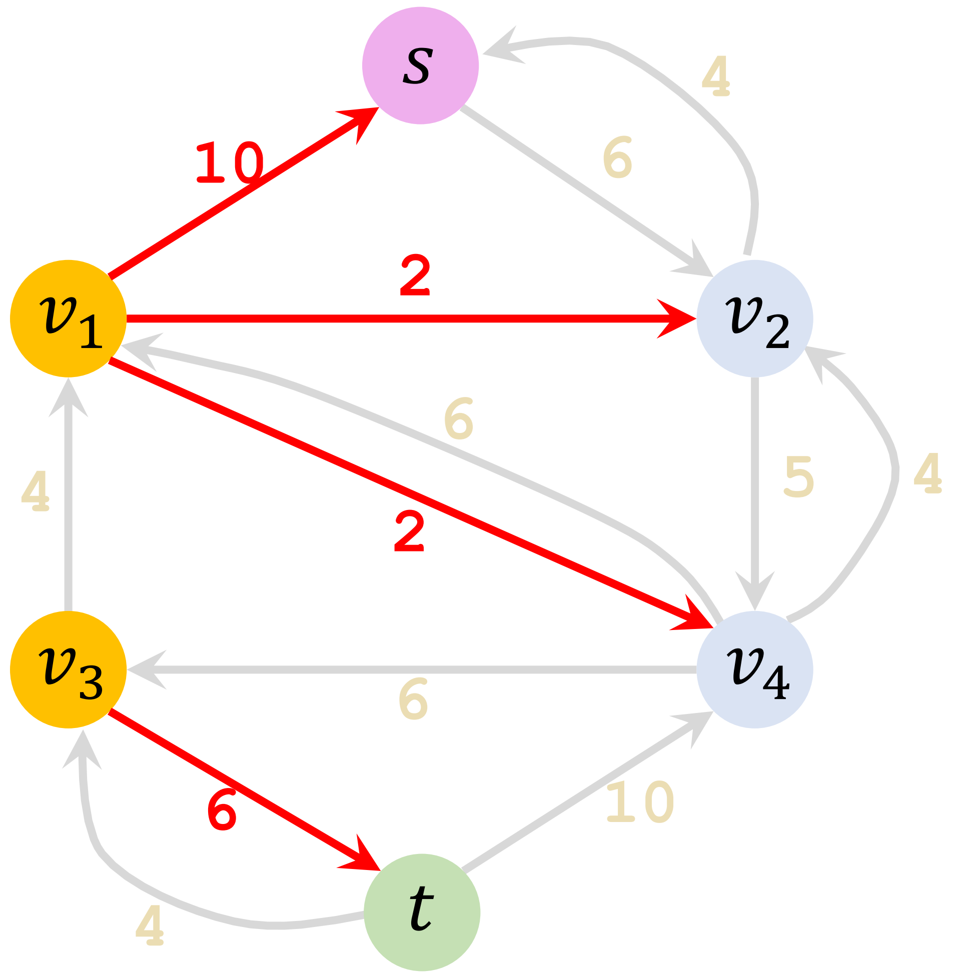


Original Graph

Level Graph: Example 2

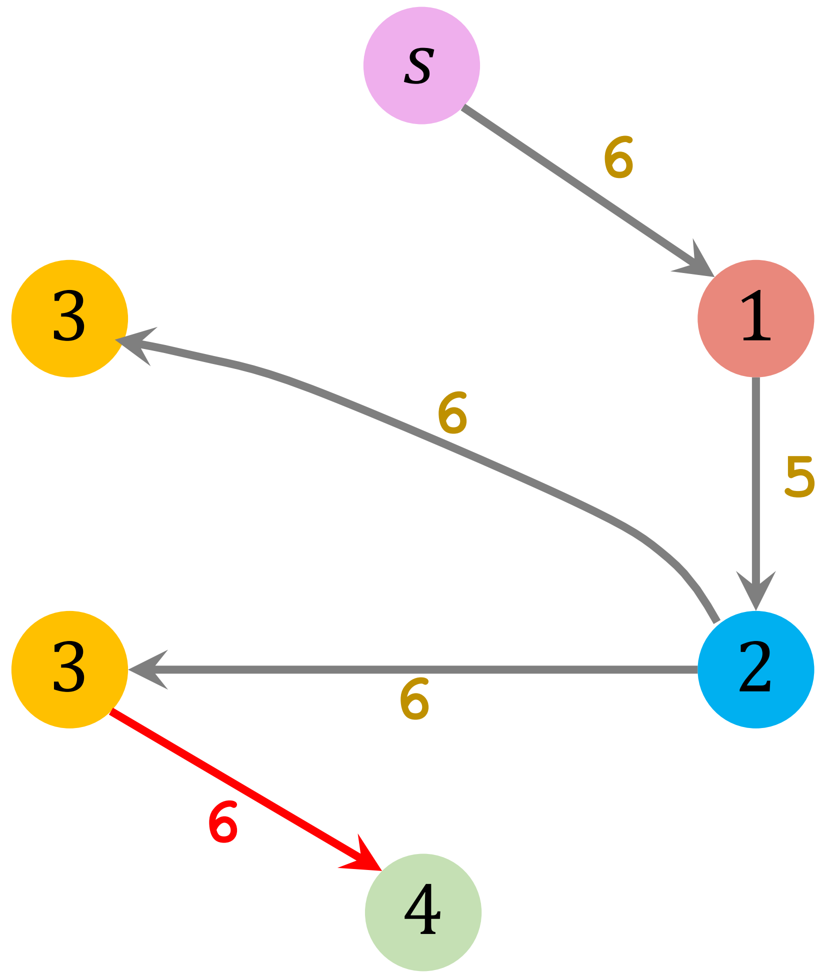


Level Graph

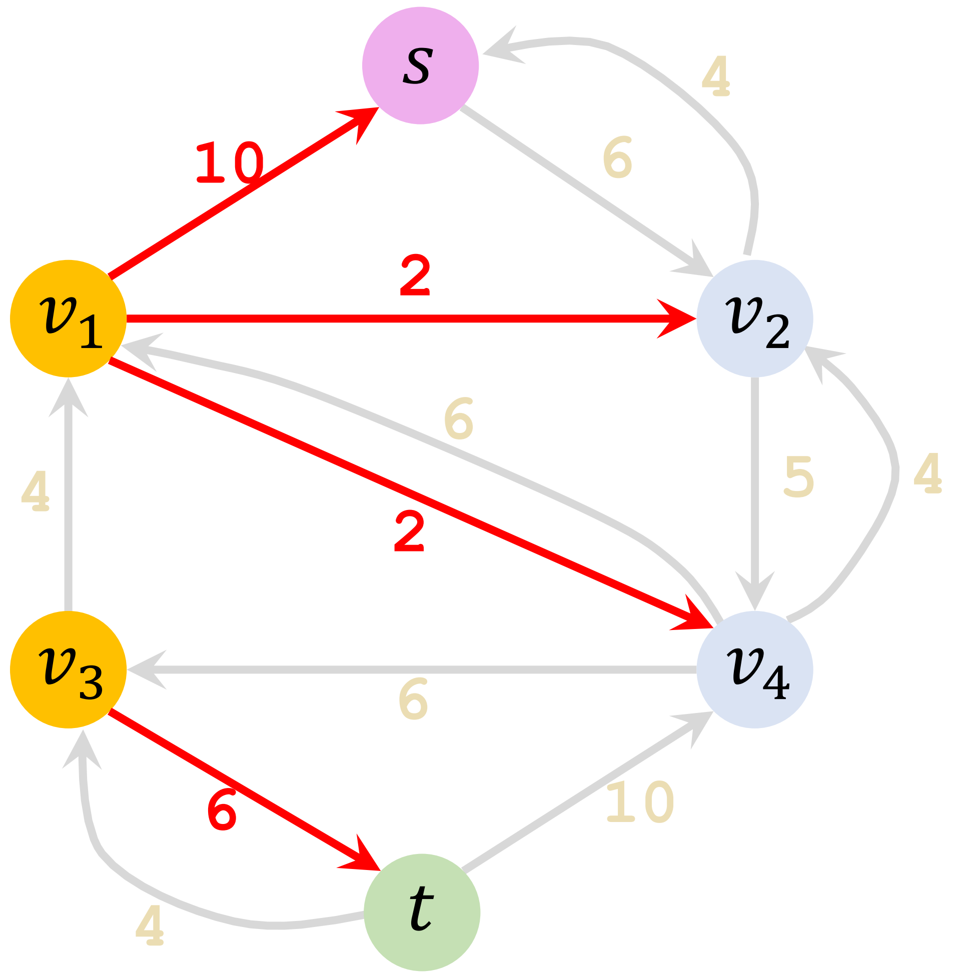


Original Graph

Level Graph: Example 2



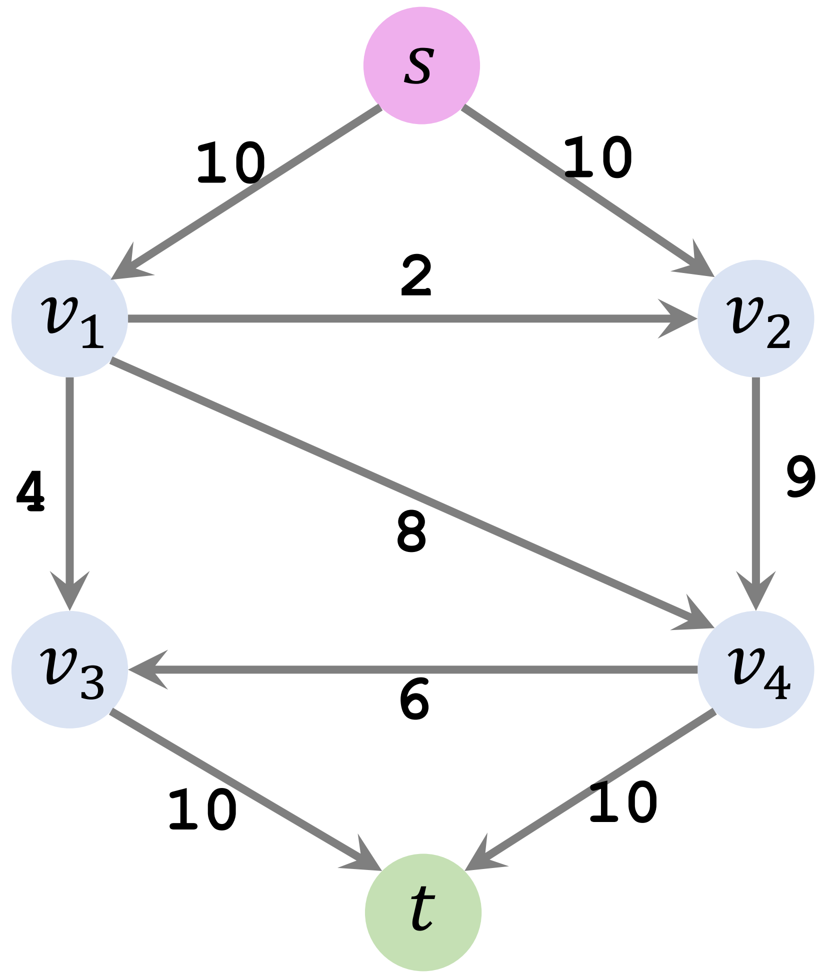
Level Graph



Original Graph

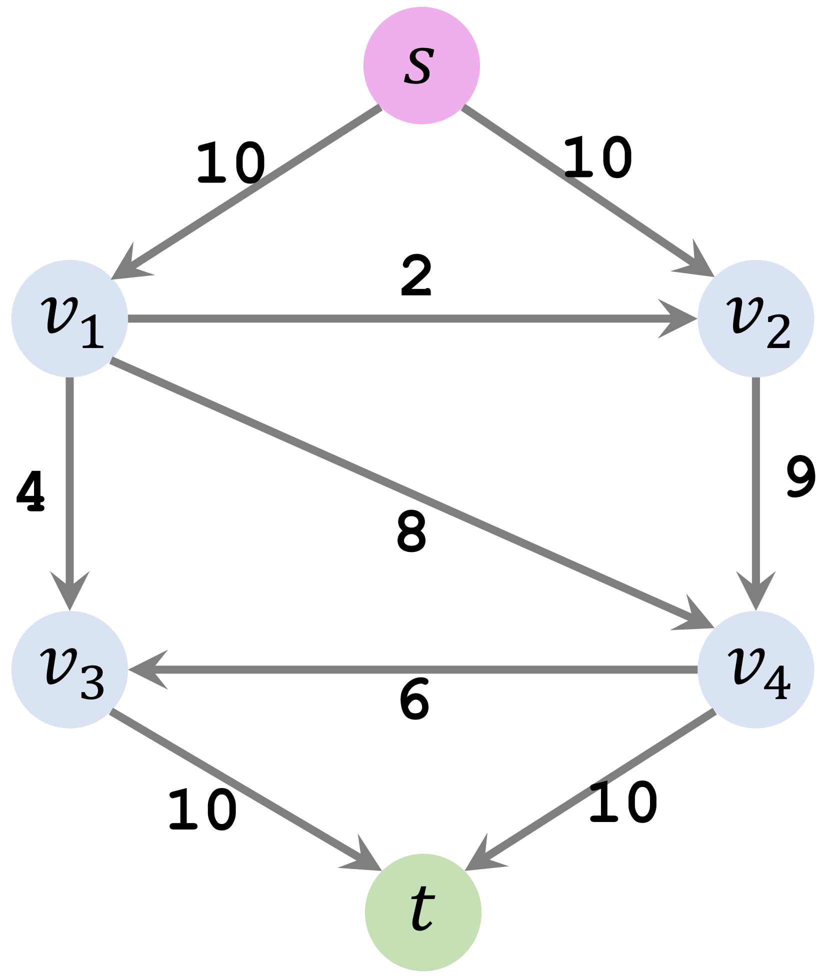
Dinic’s Algorithm

Initialization

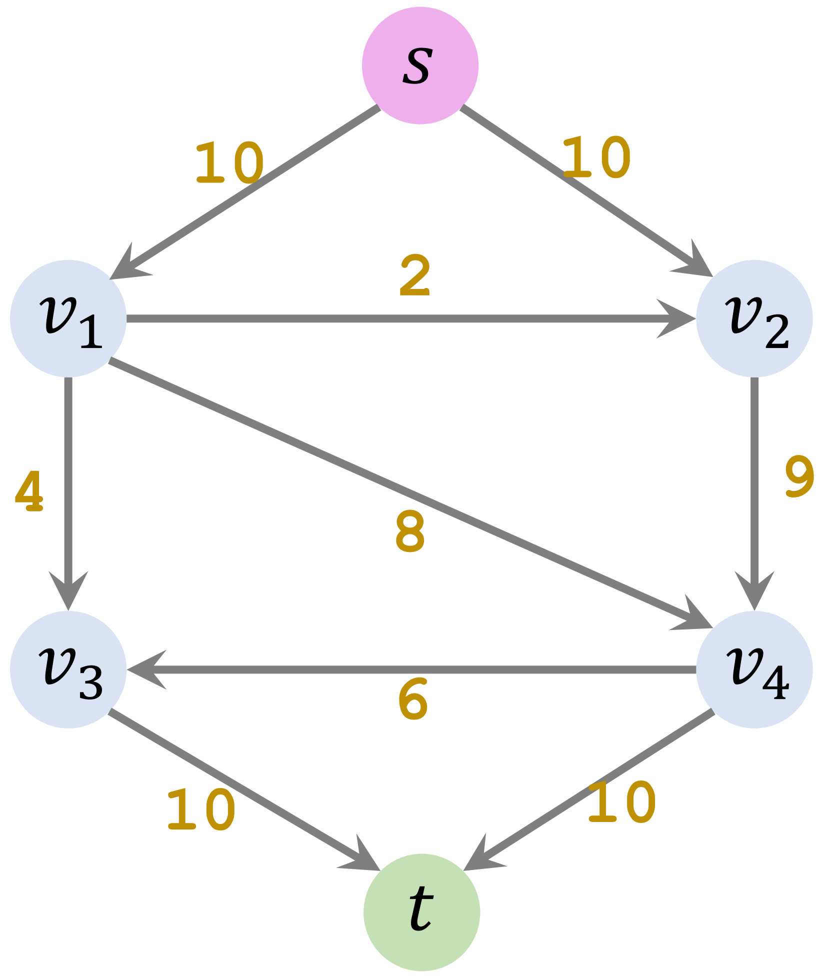


Original Graph

Initialization

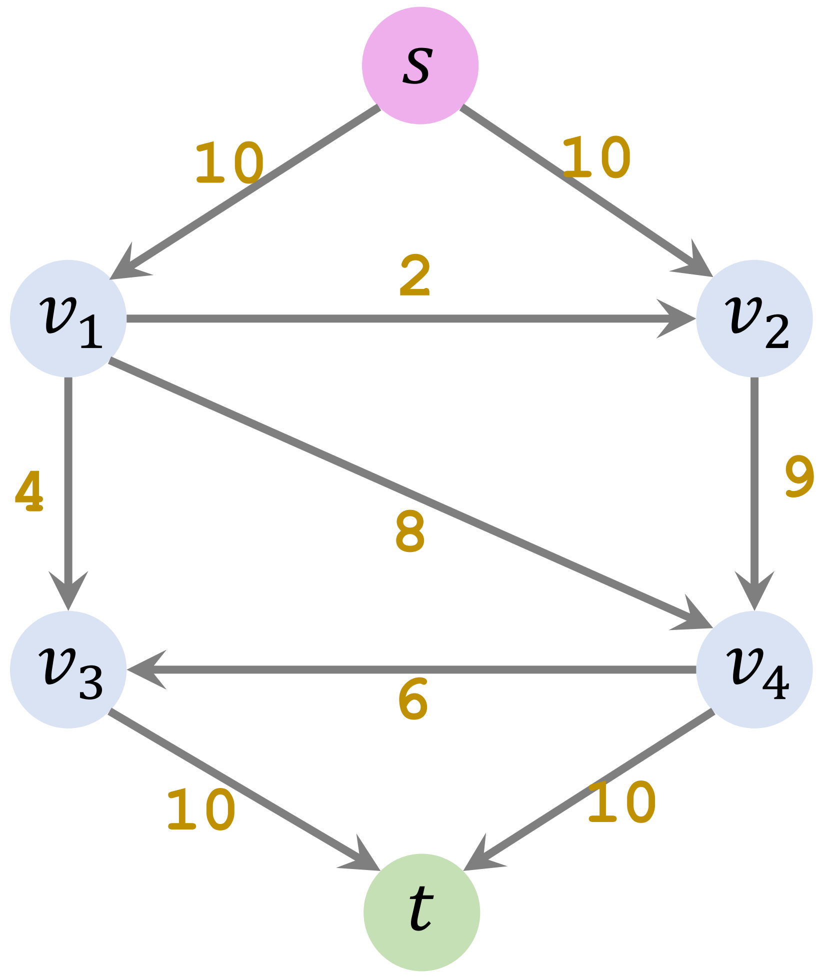


Original Graph



Residual Graph

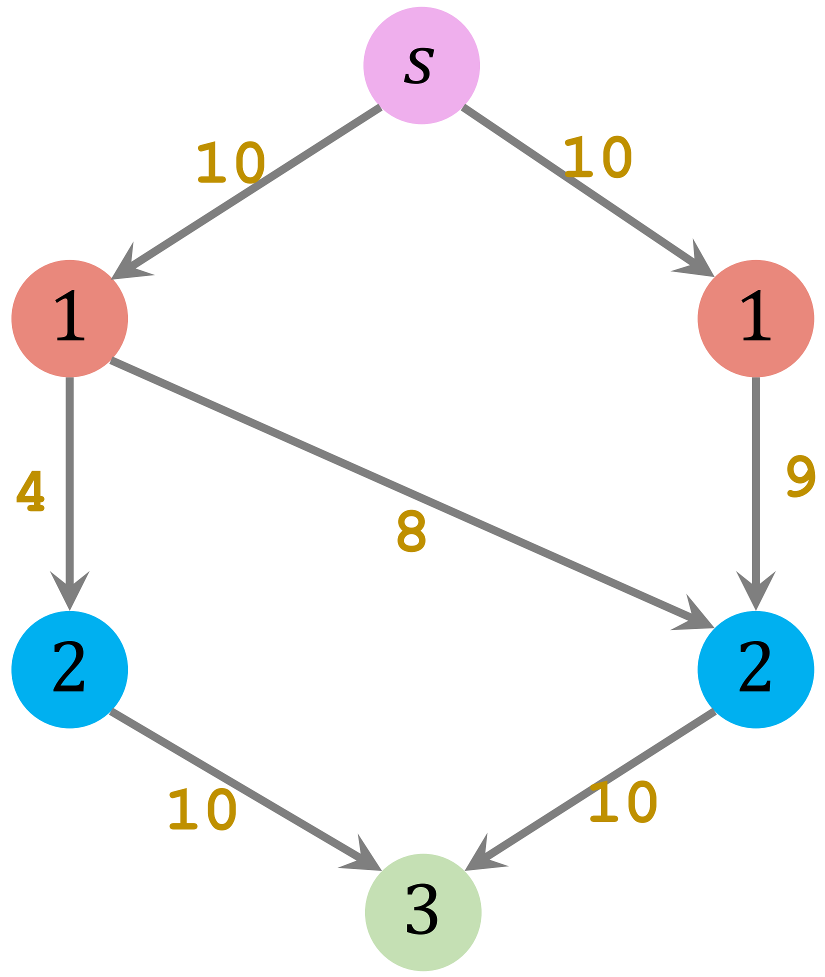
Iteration 1: Construct level graph



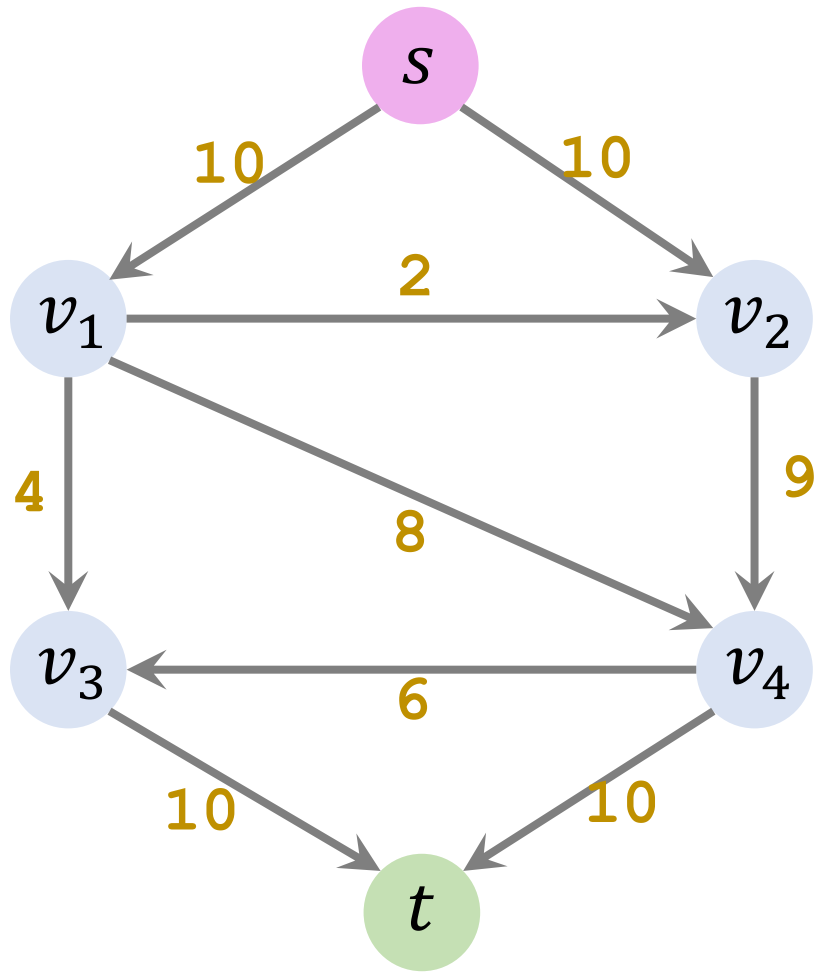
Level Graph

Residual Graph

Iteration 1: Construct level graph

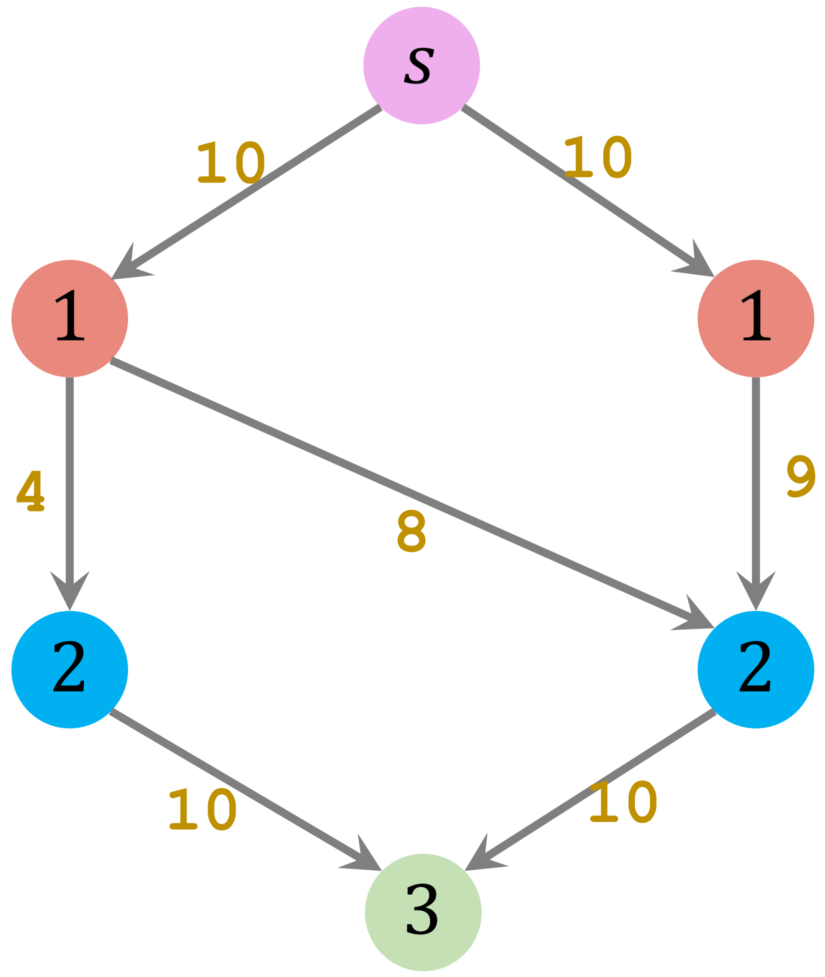


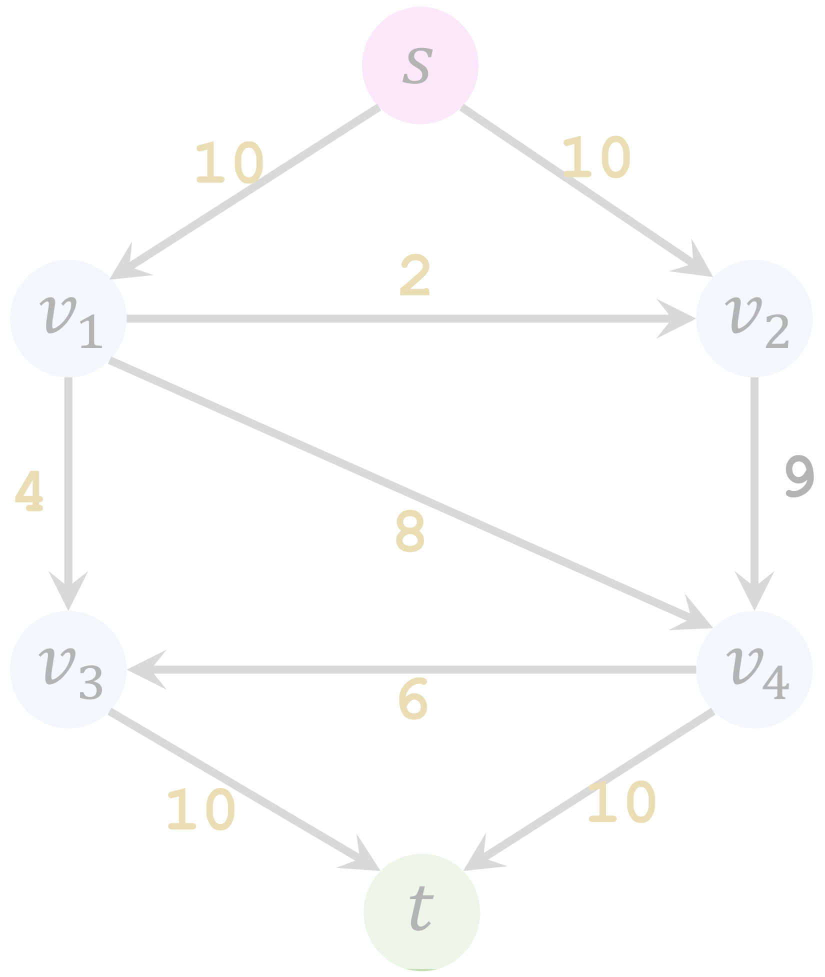
Level Graph



Residual Graph

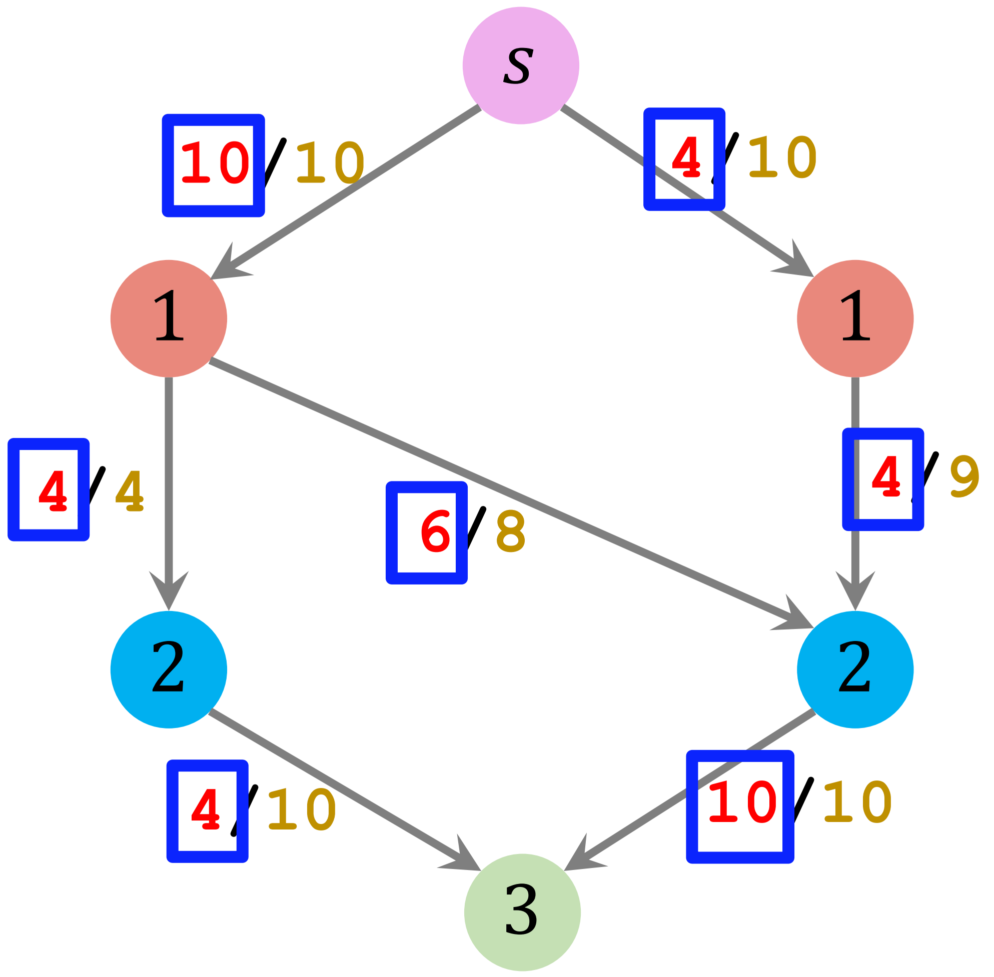
Iteration 1: Find **blocking flow** in **level graph**

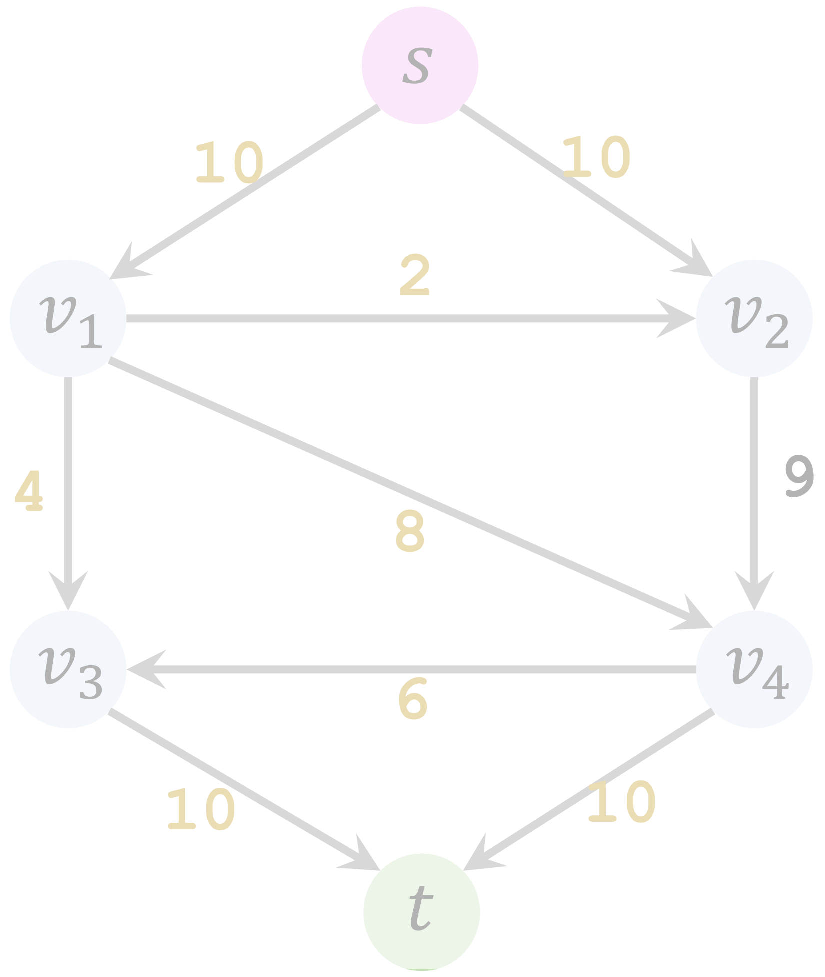




A flow is **blocking flow** if no more flow from source to sink can be found.

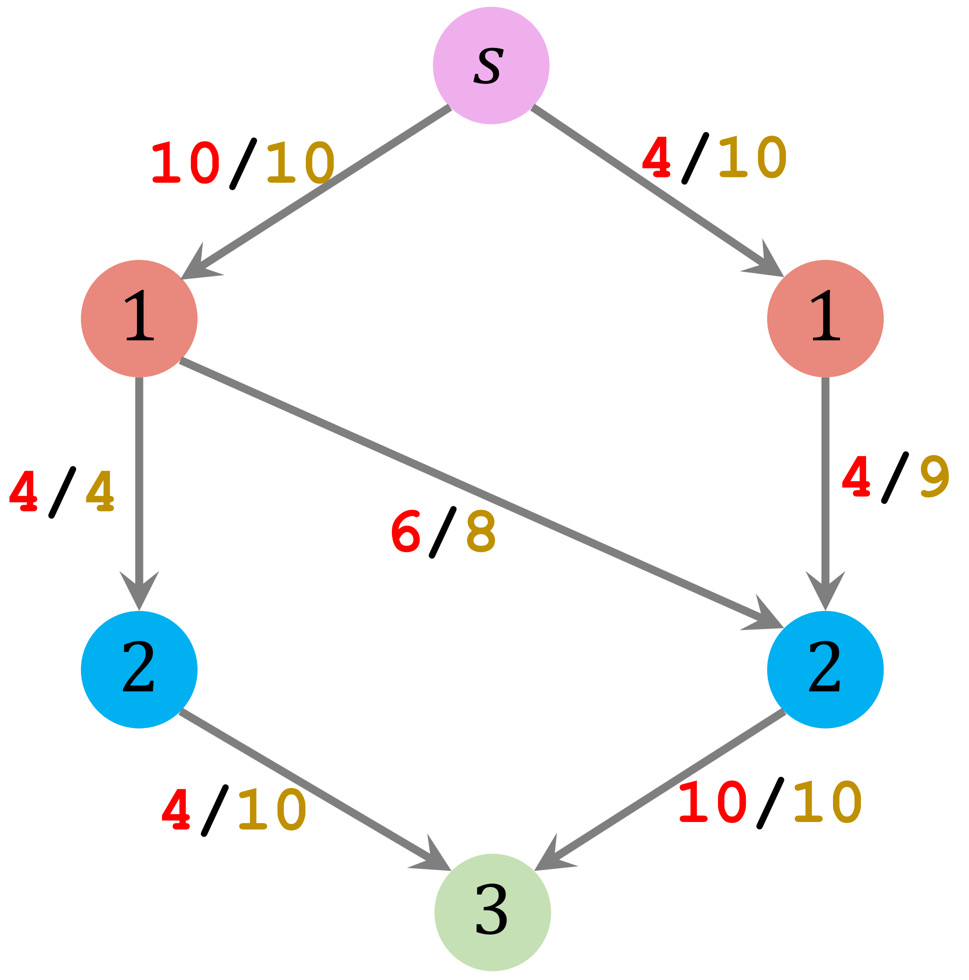
Iteration 1: Find blocking flow in level graph



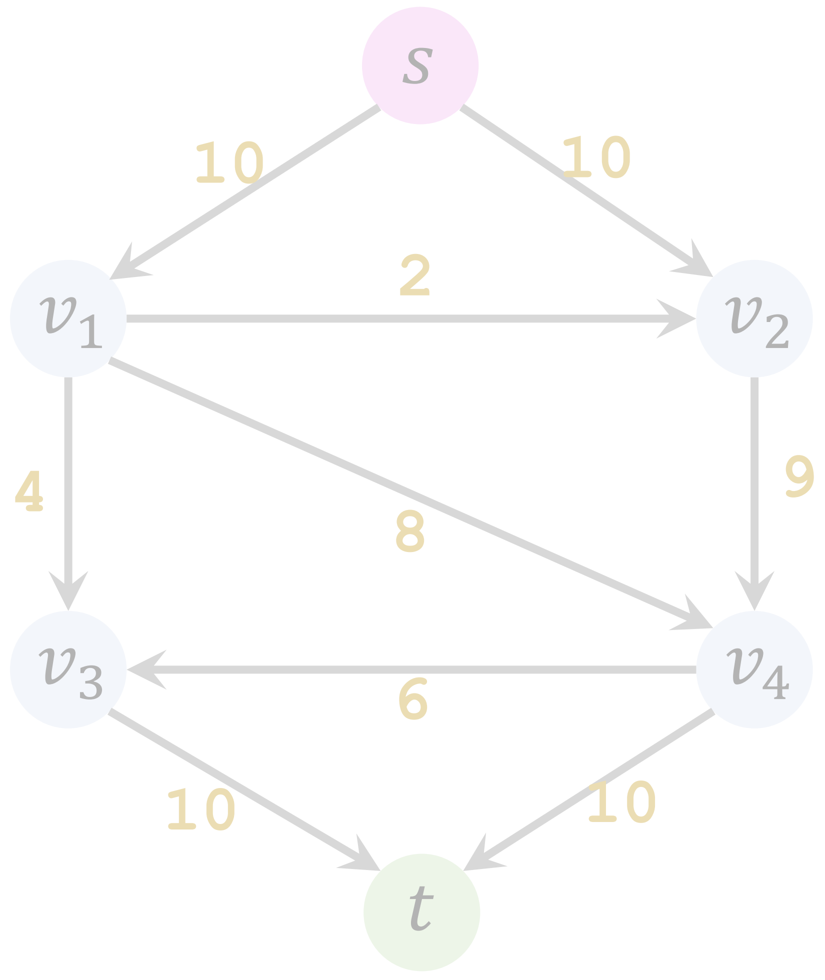


Blocking flow can be found using the naïve algorithm.

Iteration 1: Update the residual graph

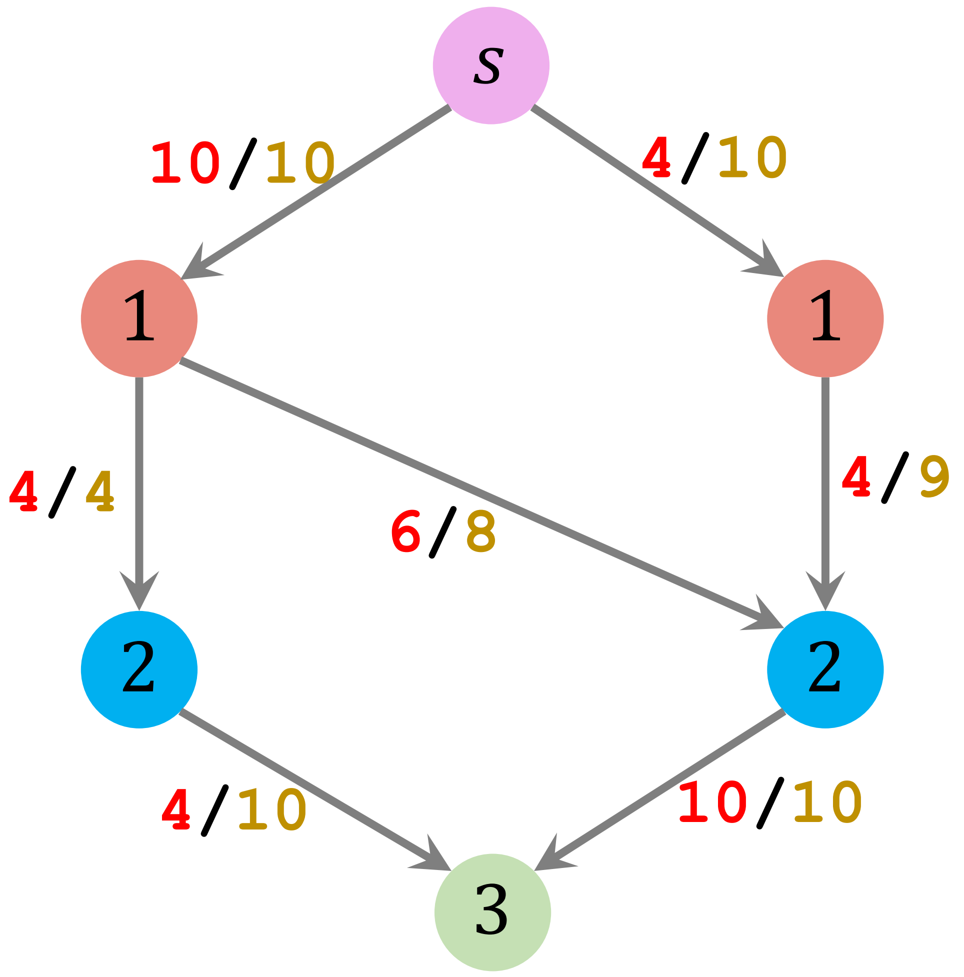


Level Graph

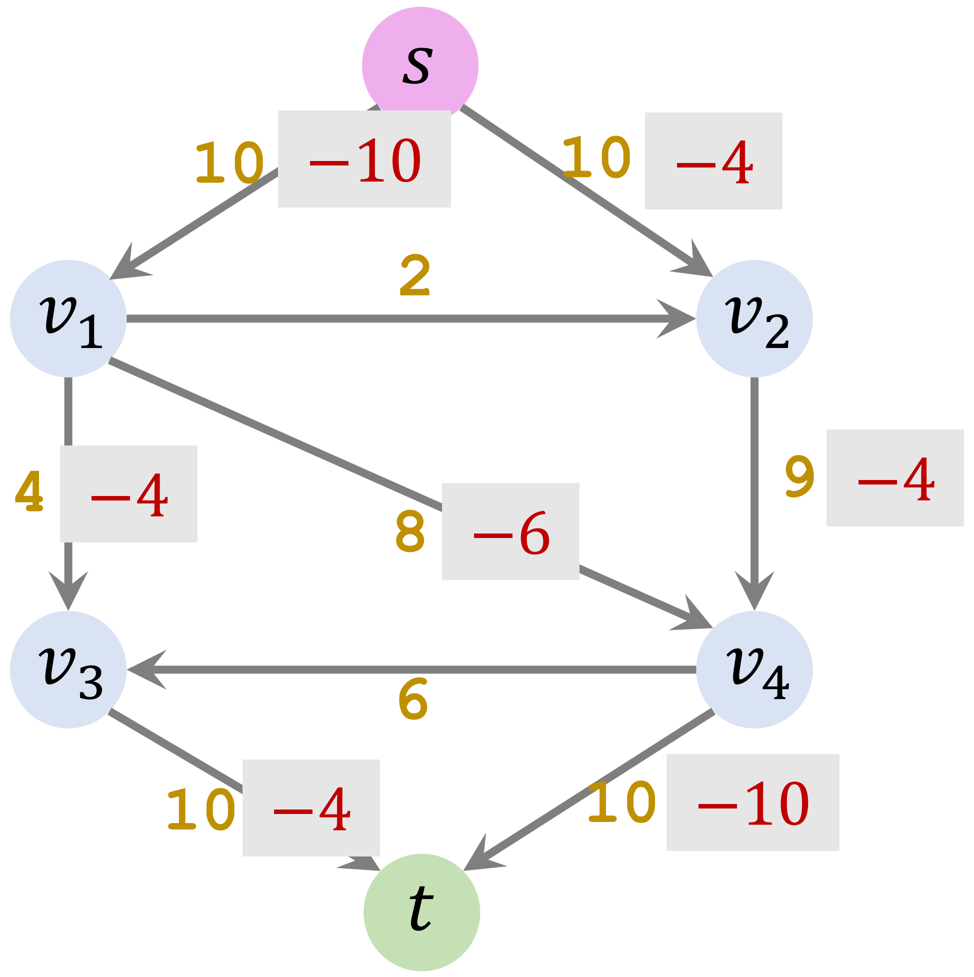


Old Residual Graph

Iteration 1: Update the residual graph

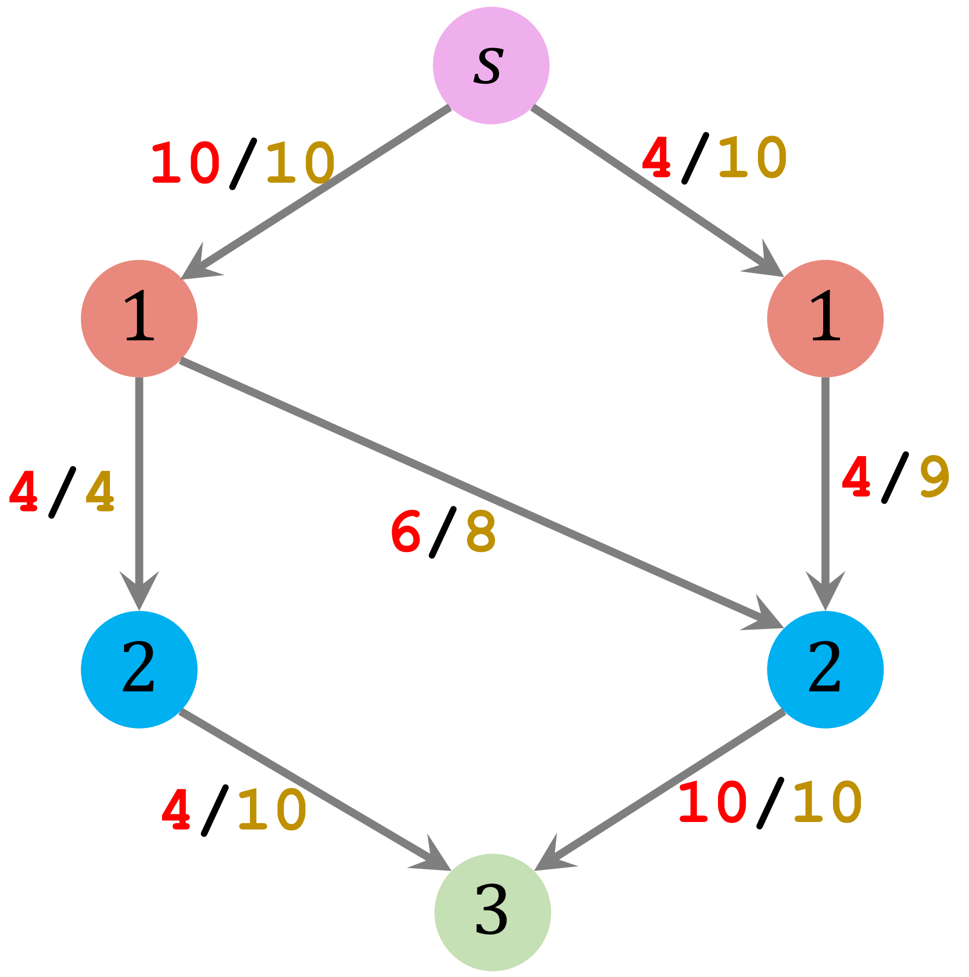


Level Graph

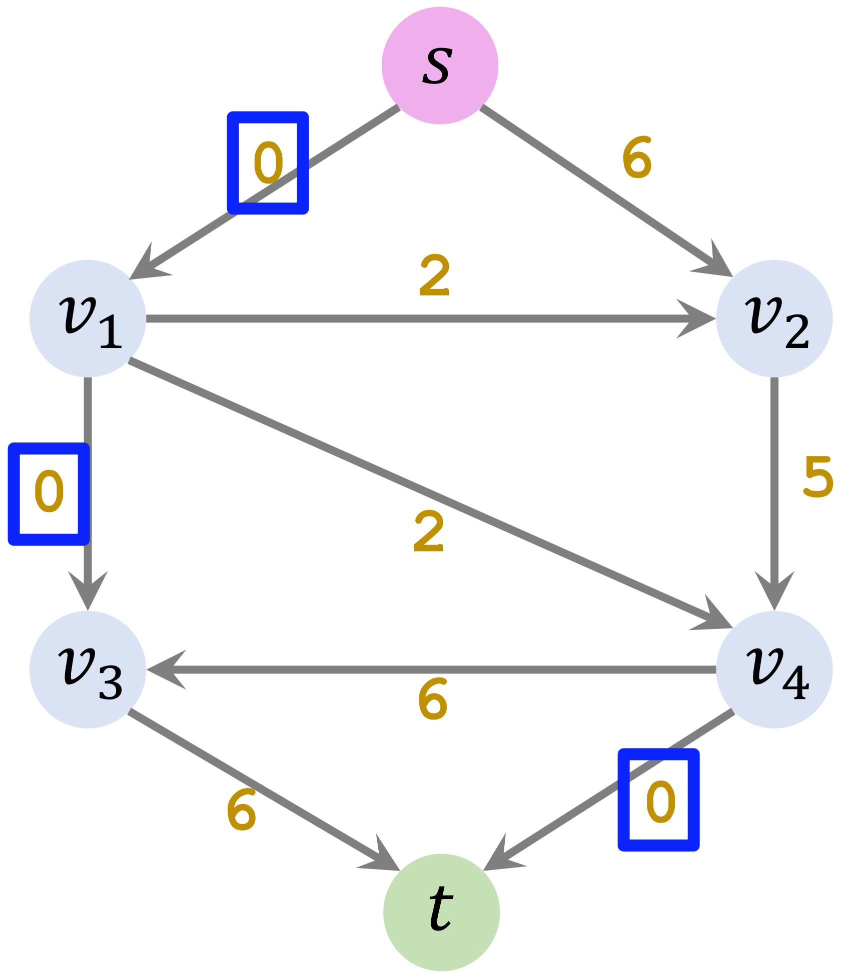


Old Residual Graph

Iteration 1: Update the residual graph

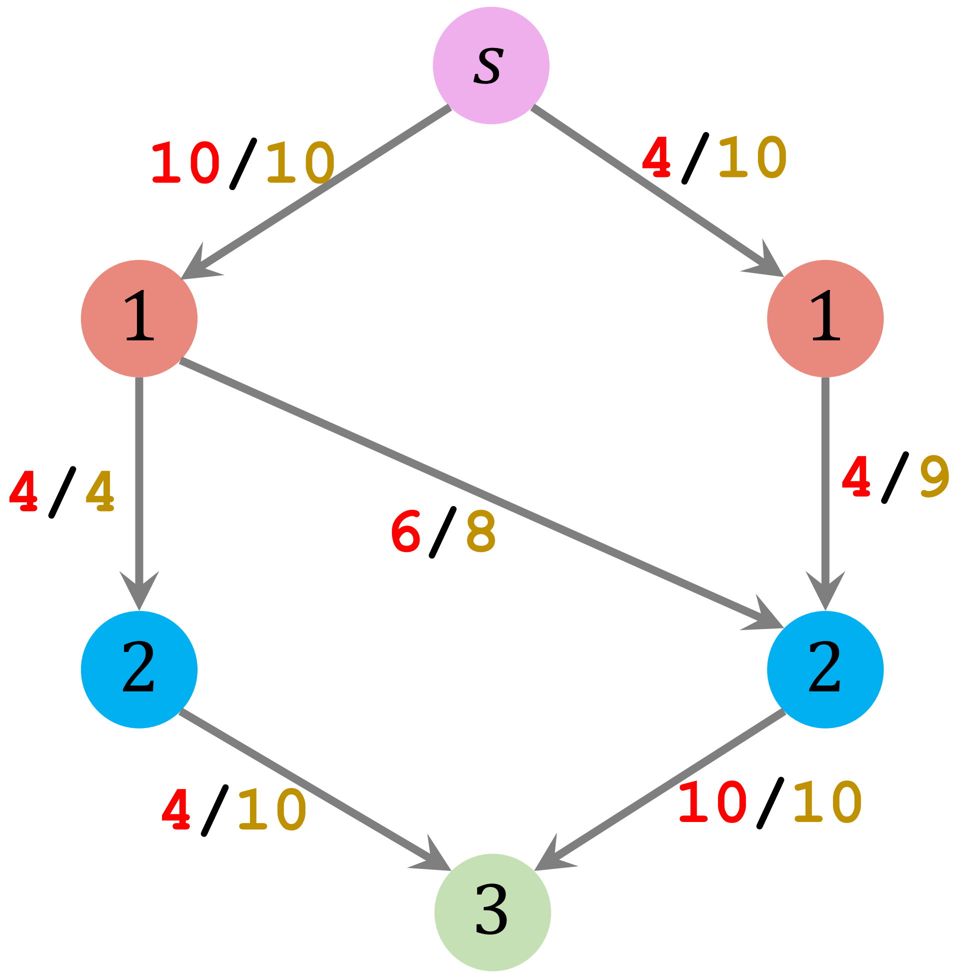


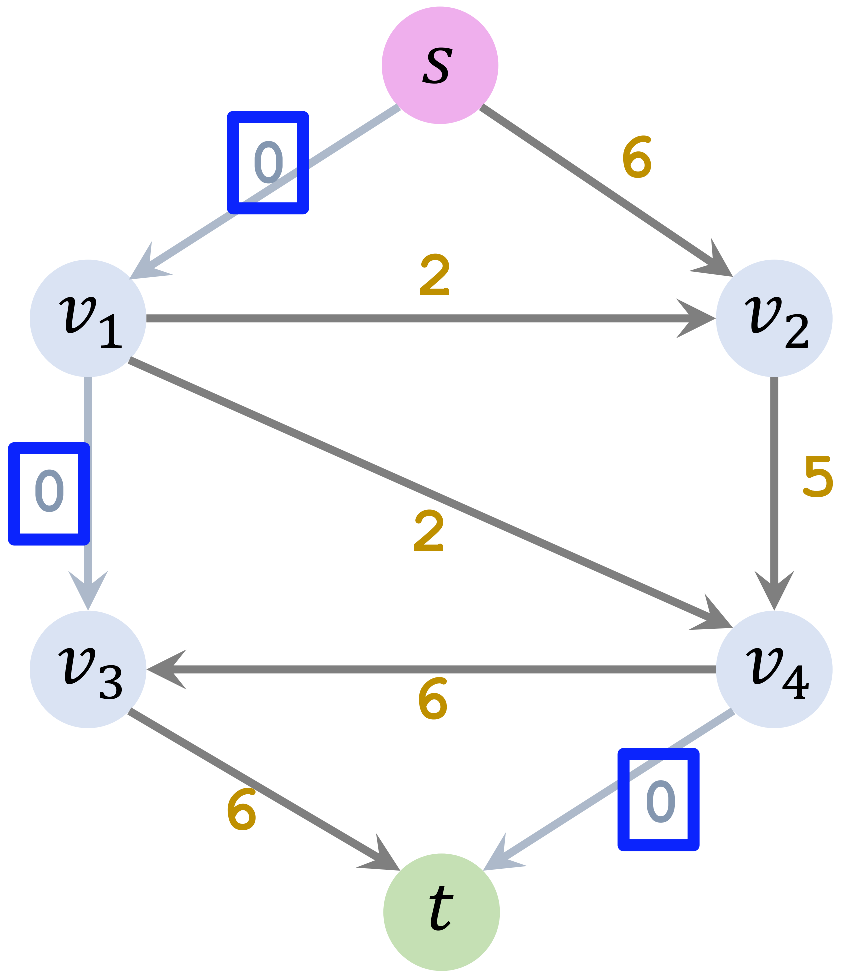
Level Graph



New Residual Graph

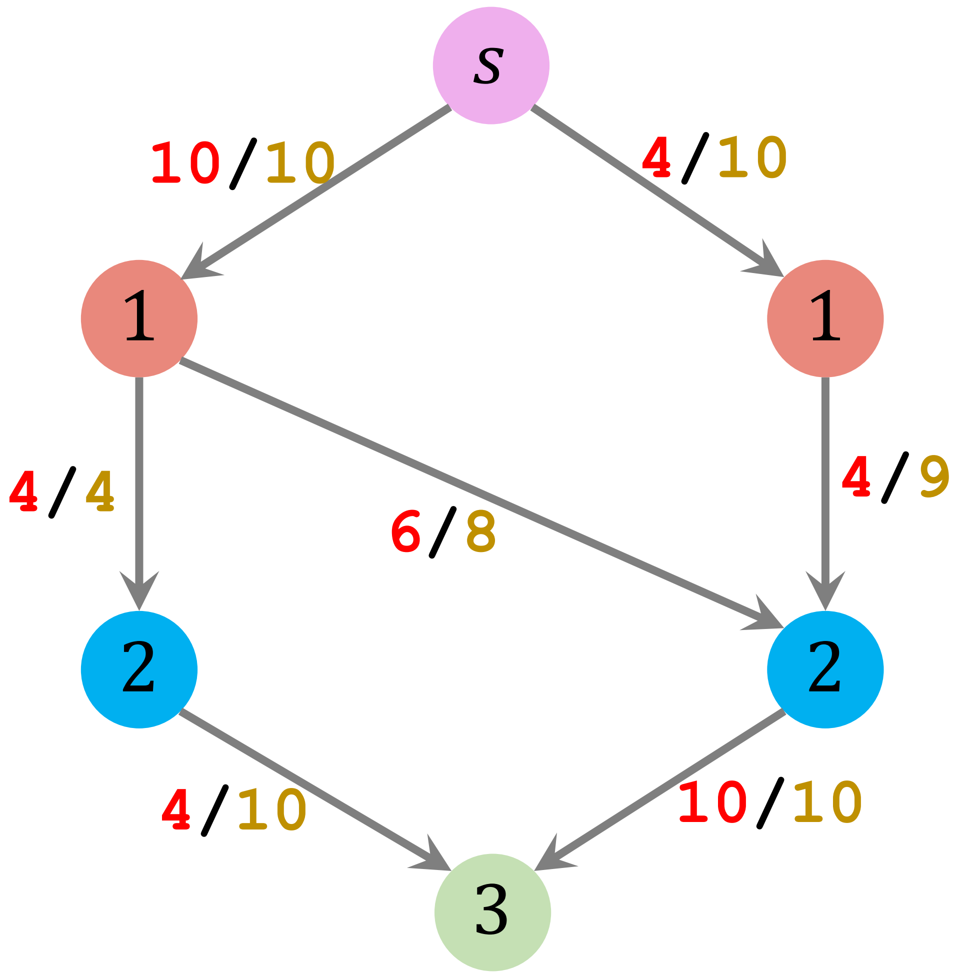
Iteration 1: Update the residual graph

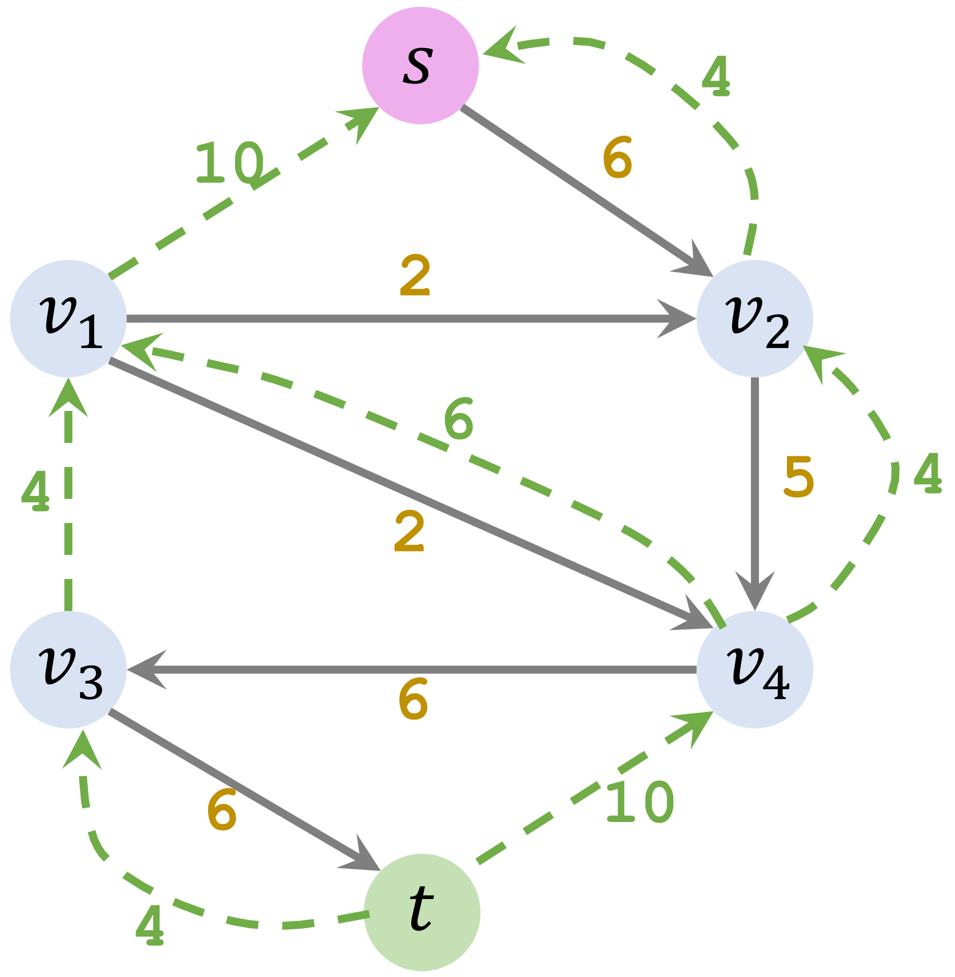




Removed saturated edges from residual graph.

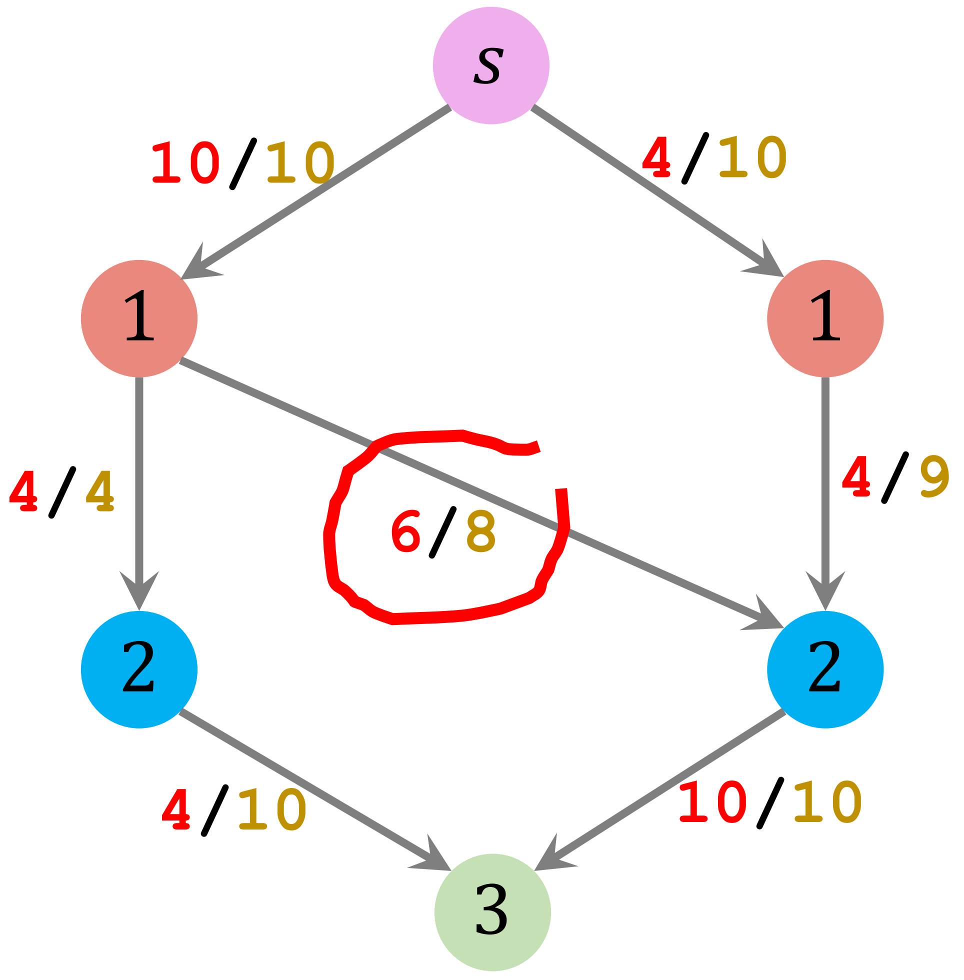
Iteration 1: Update the residual graph

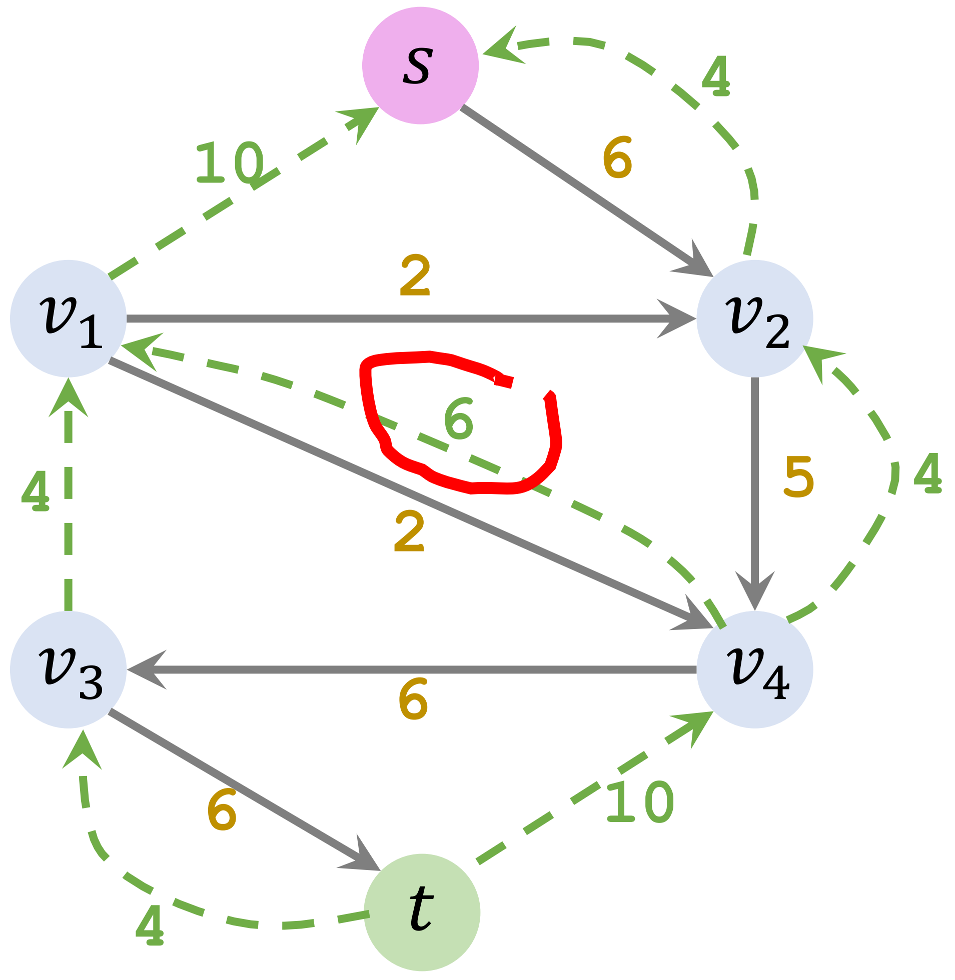




Add flows to the residual graph as backward paths.

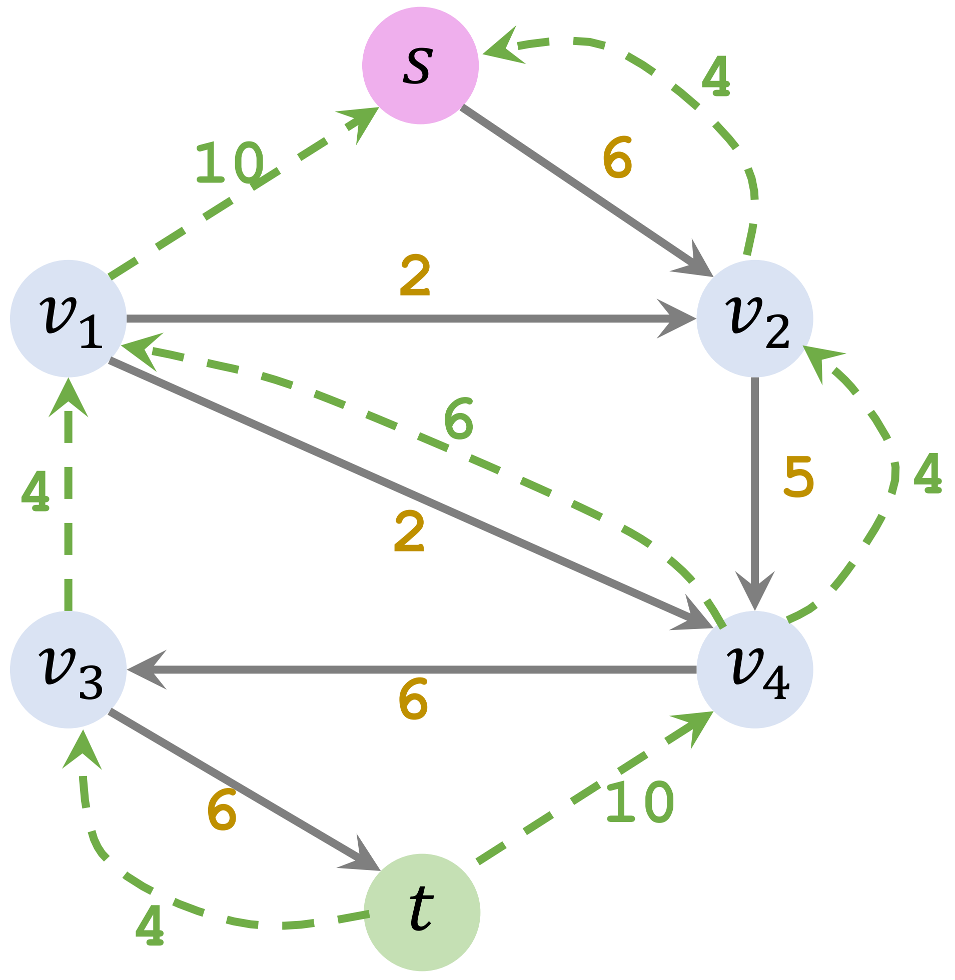
Iteration 1: Update the residual graph





Add flows to the residual graph as backward paths.

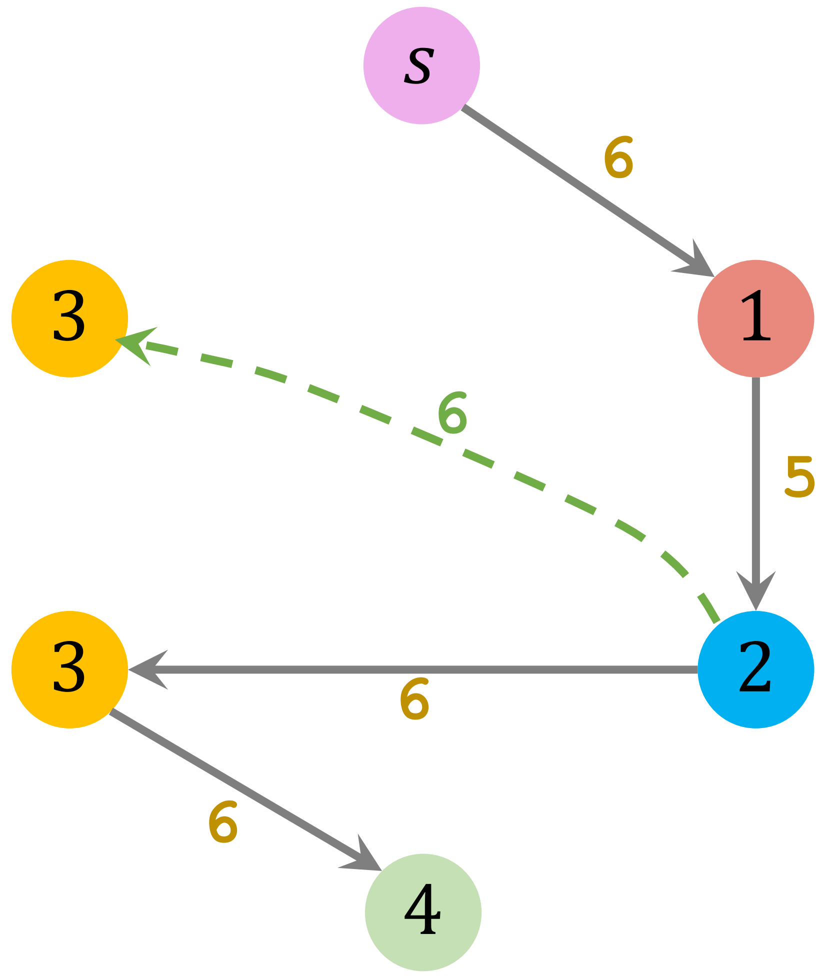
Iteration 2: Construct level graph



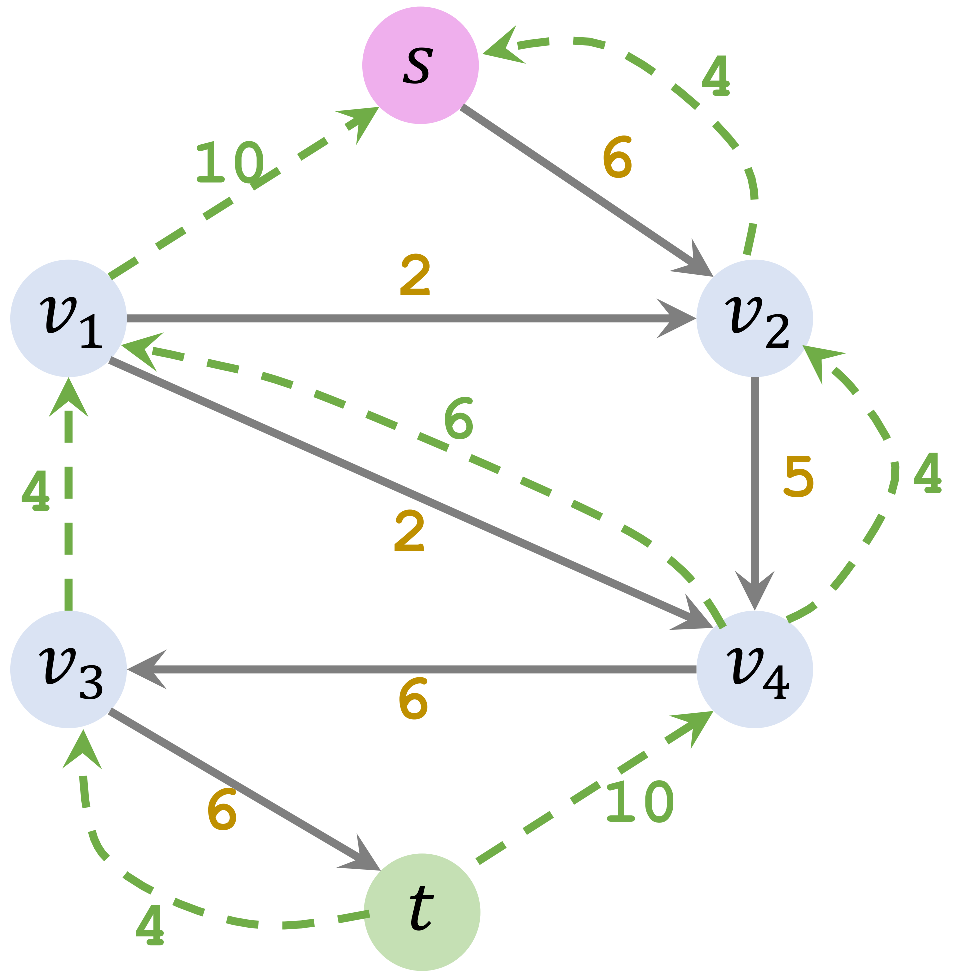
Level Graph

Residual Graph

Iteration 2: Construct level graph

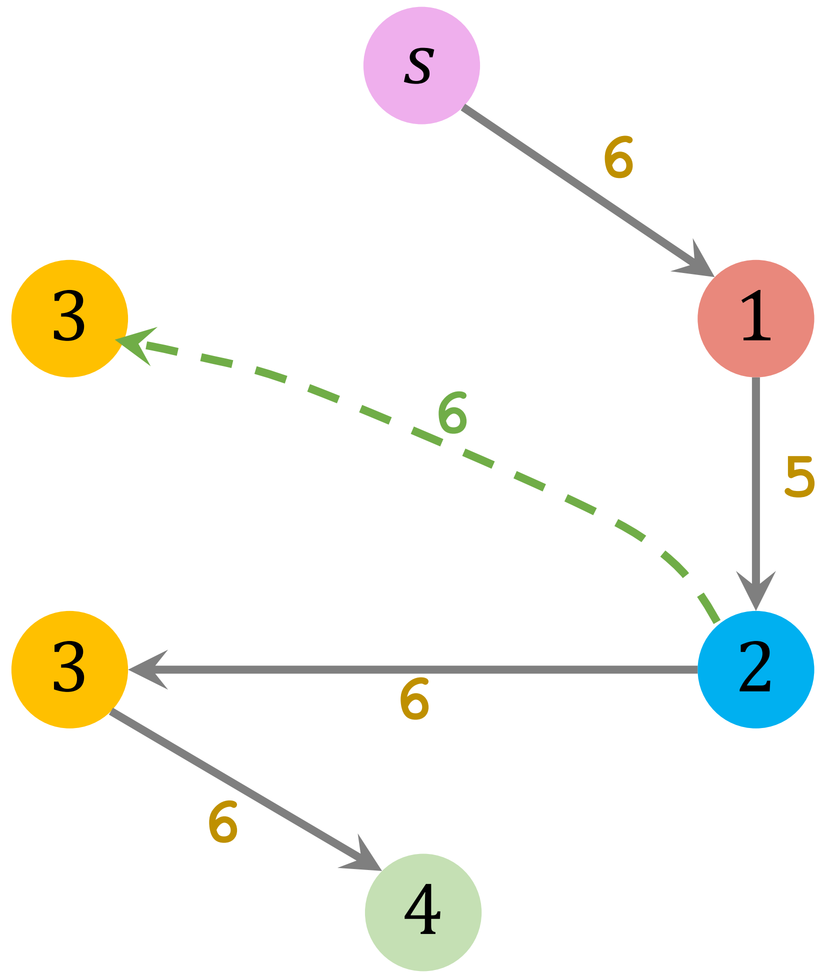


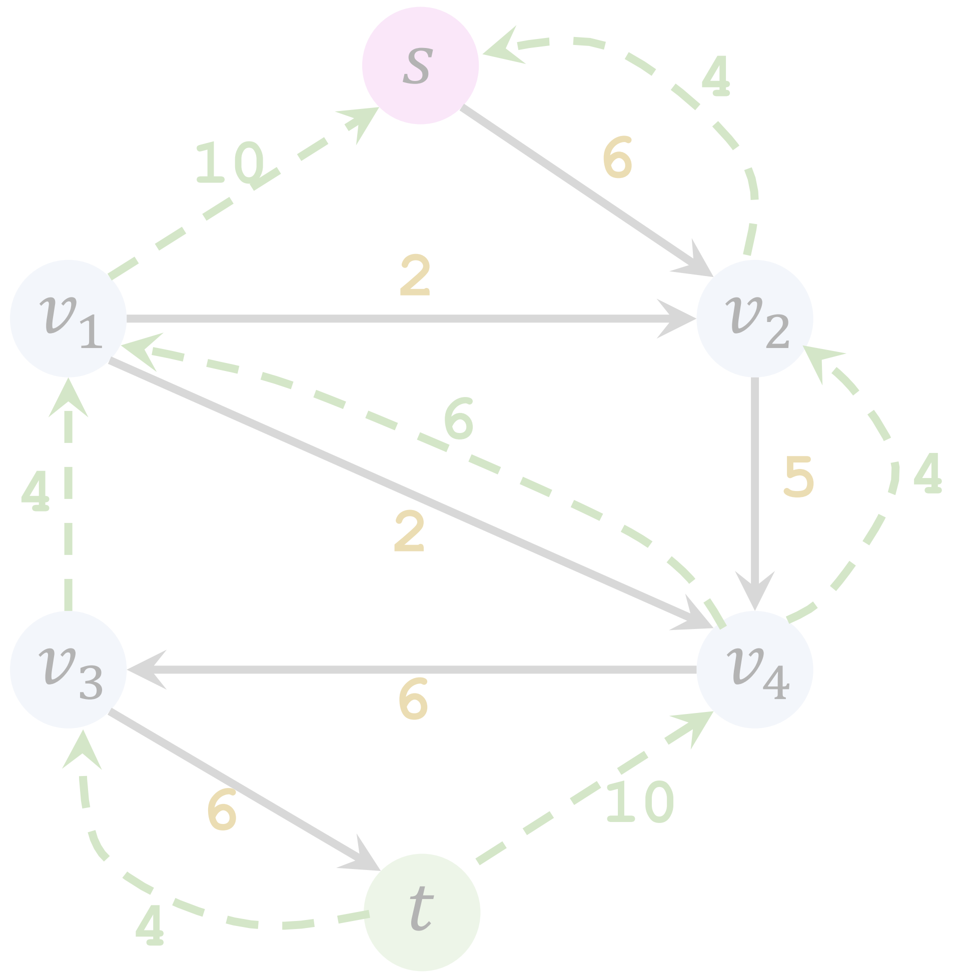
Level Graph



Residual Graph

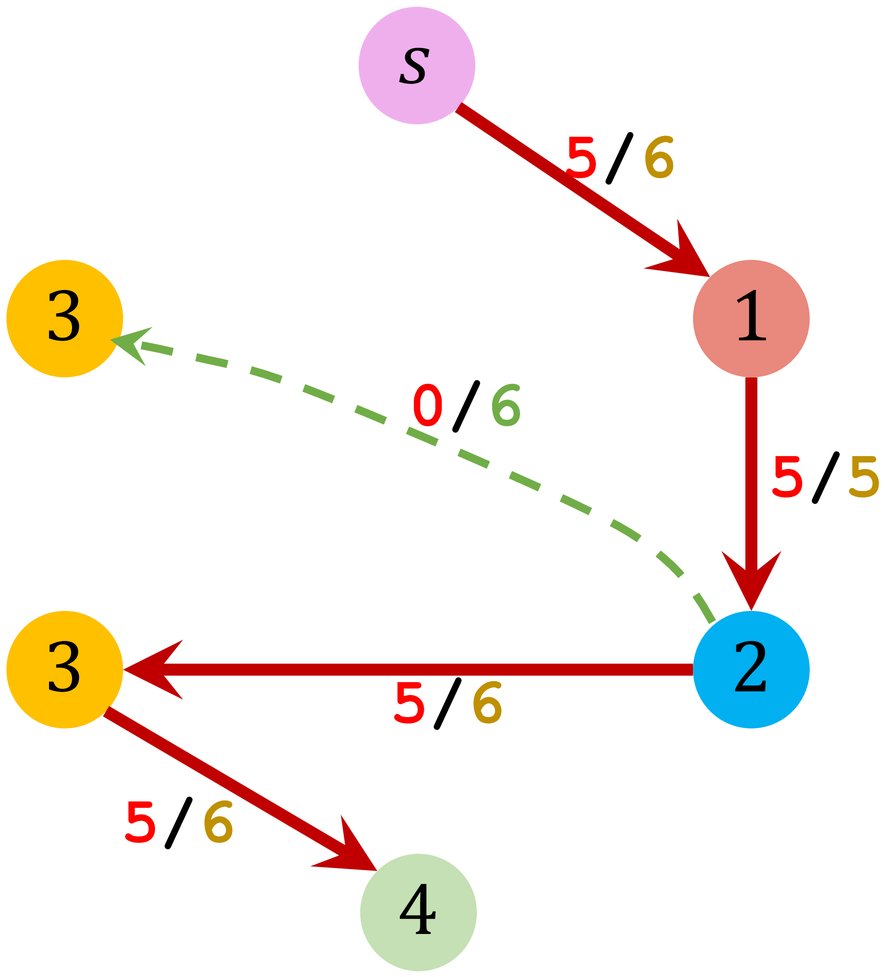
Iteration 2: Find blocking flow in level graph

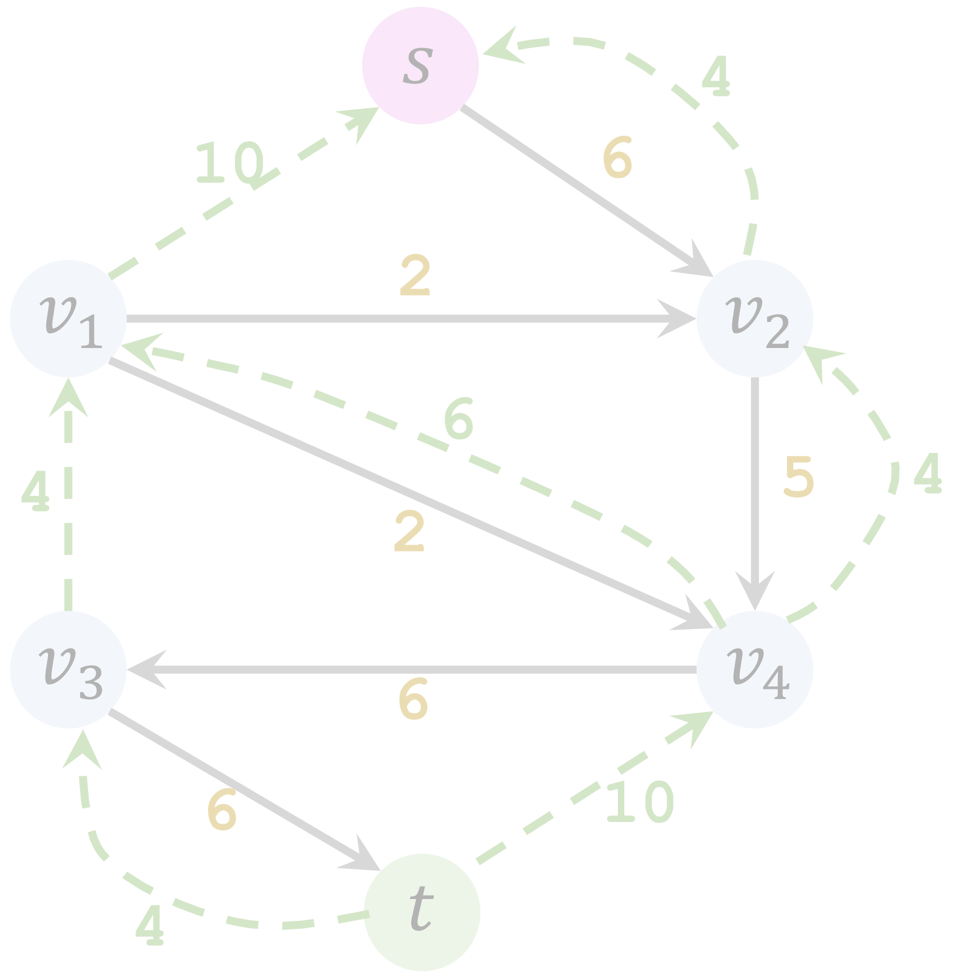




A flow is **blocking flow** if no more flow from source to sink can be found.

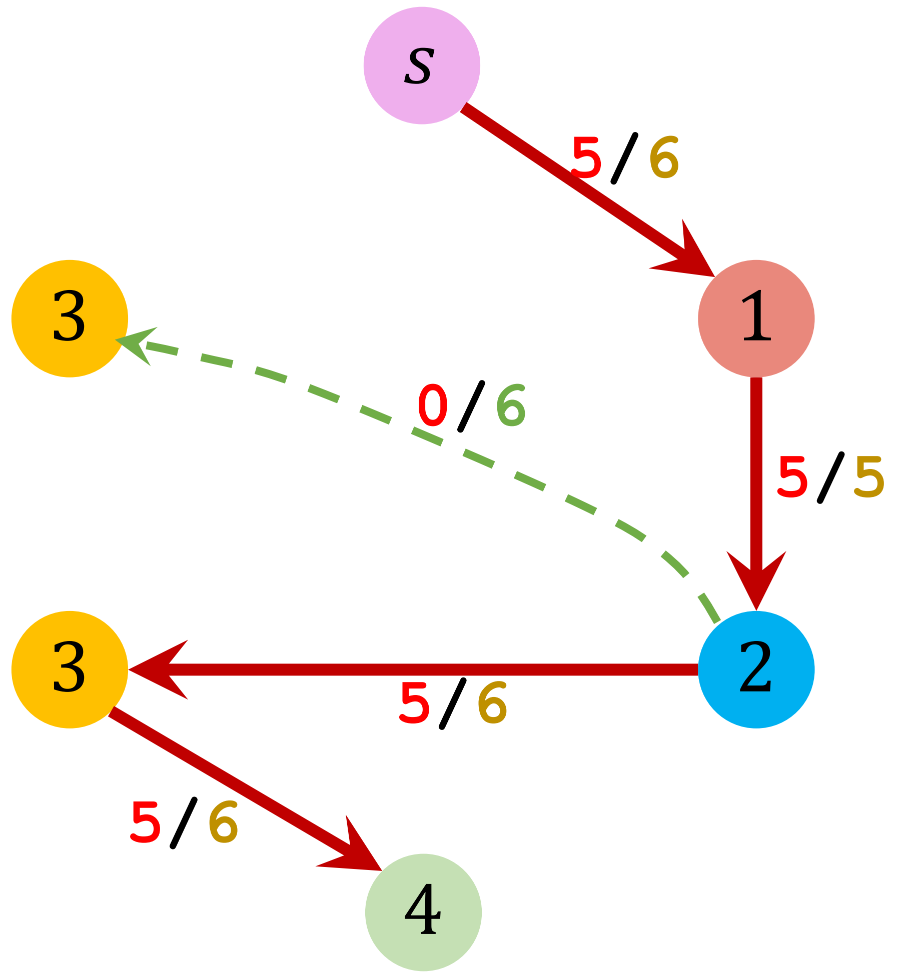
Iteration 2: Find blocking flow in level graph



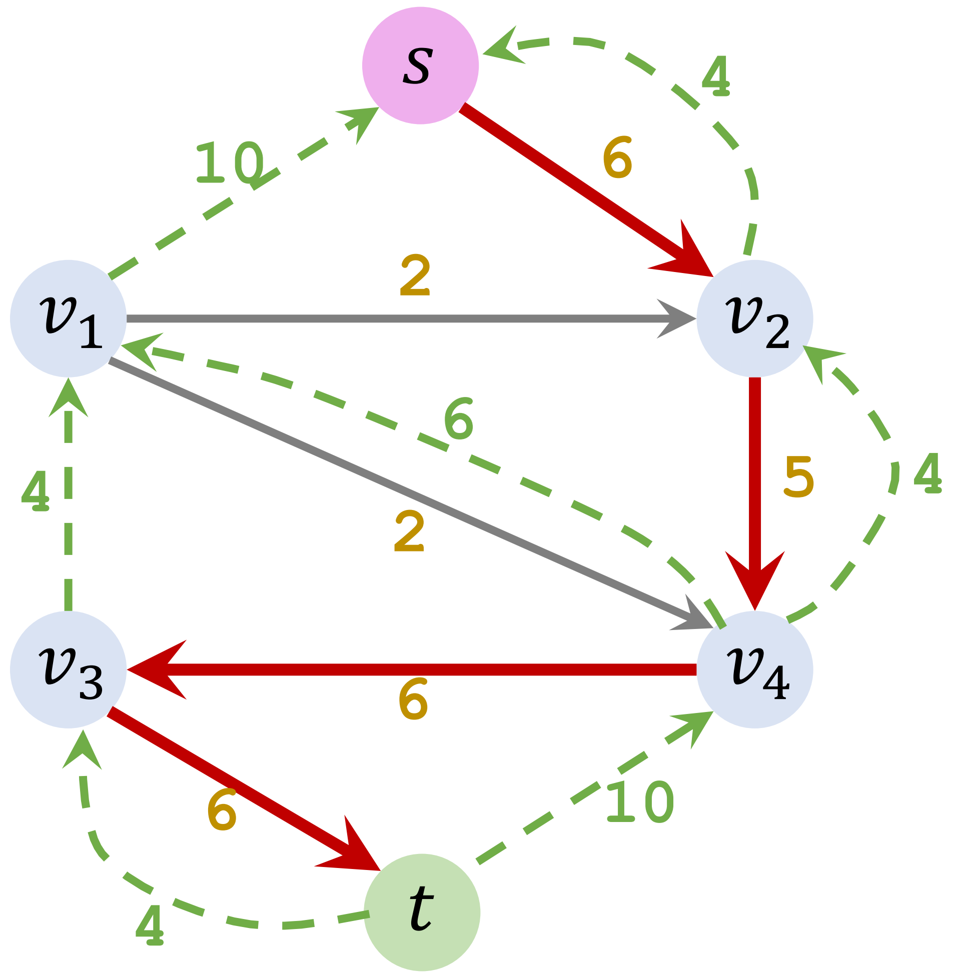


Blocking flow can be found using the naïve algorithm.

Iteration 2: Update the residual graph

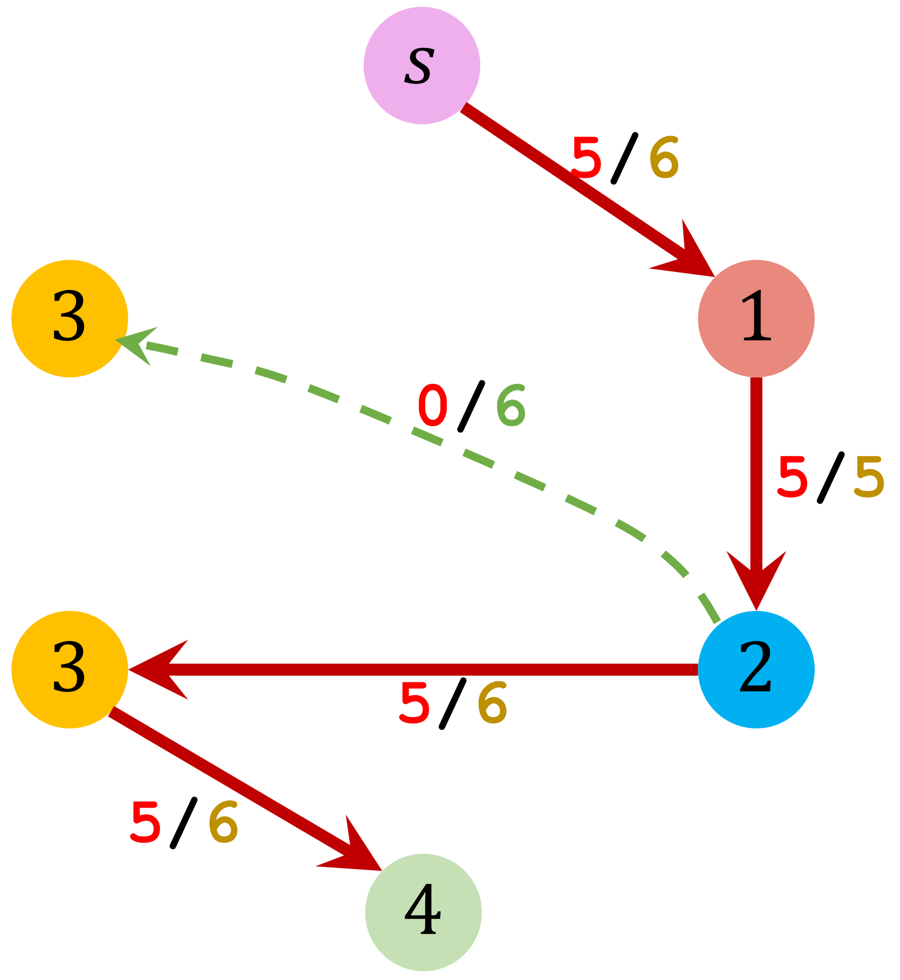


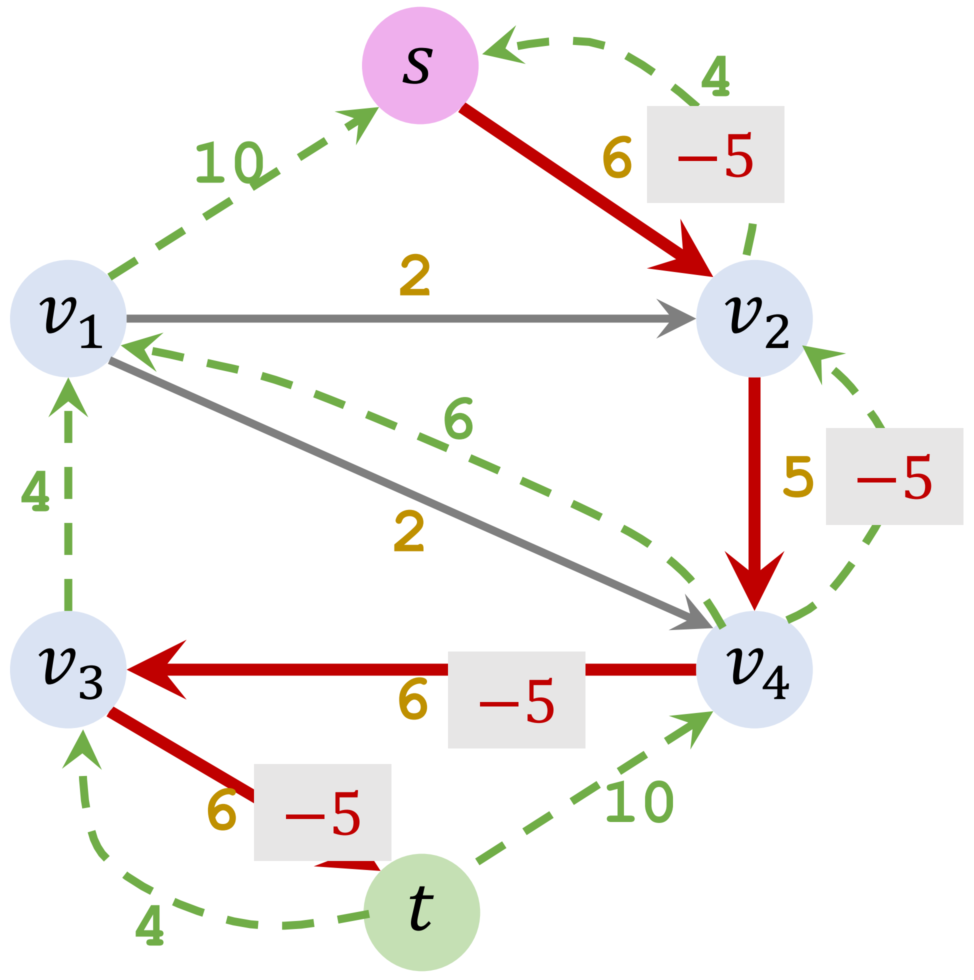
Blocking Flow



Old Residual Graph

Iteration 2: Update the residual graph

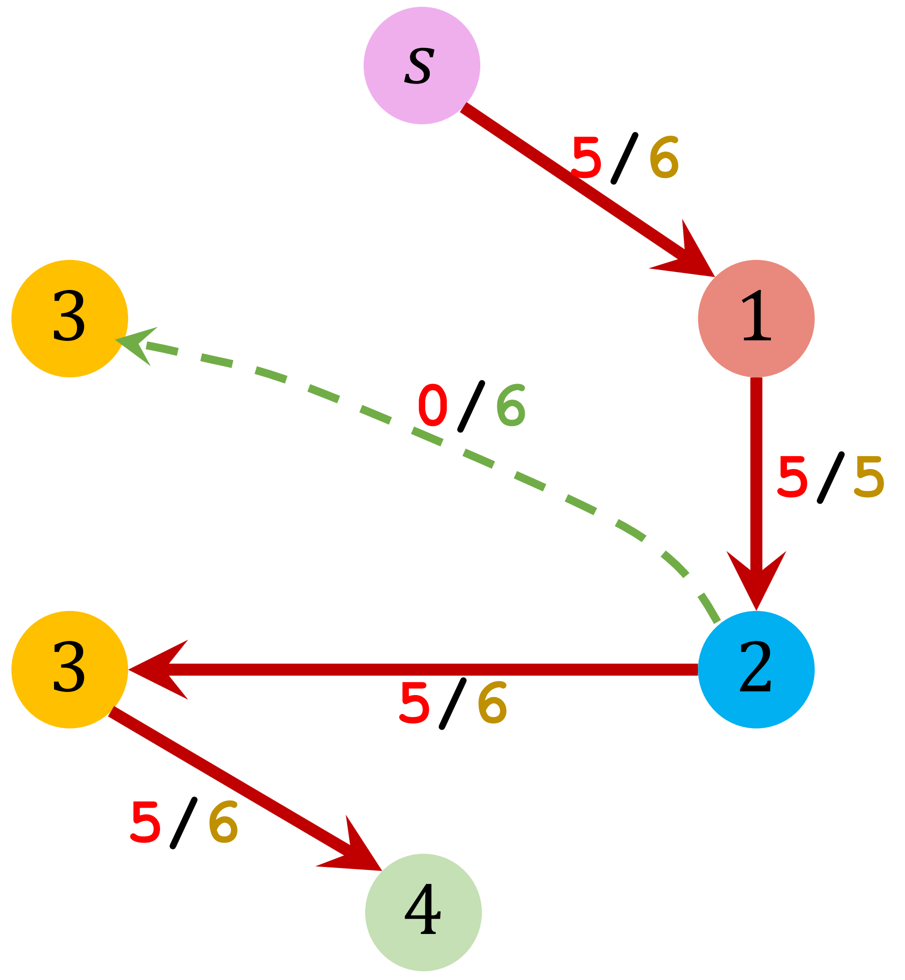


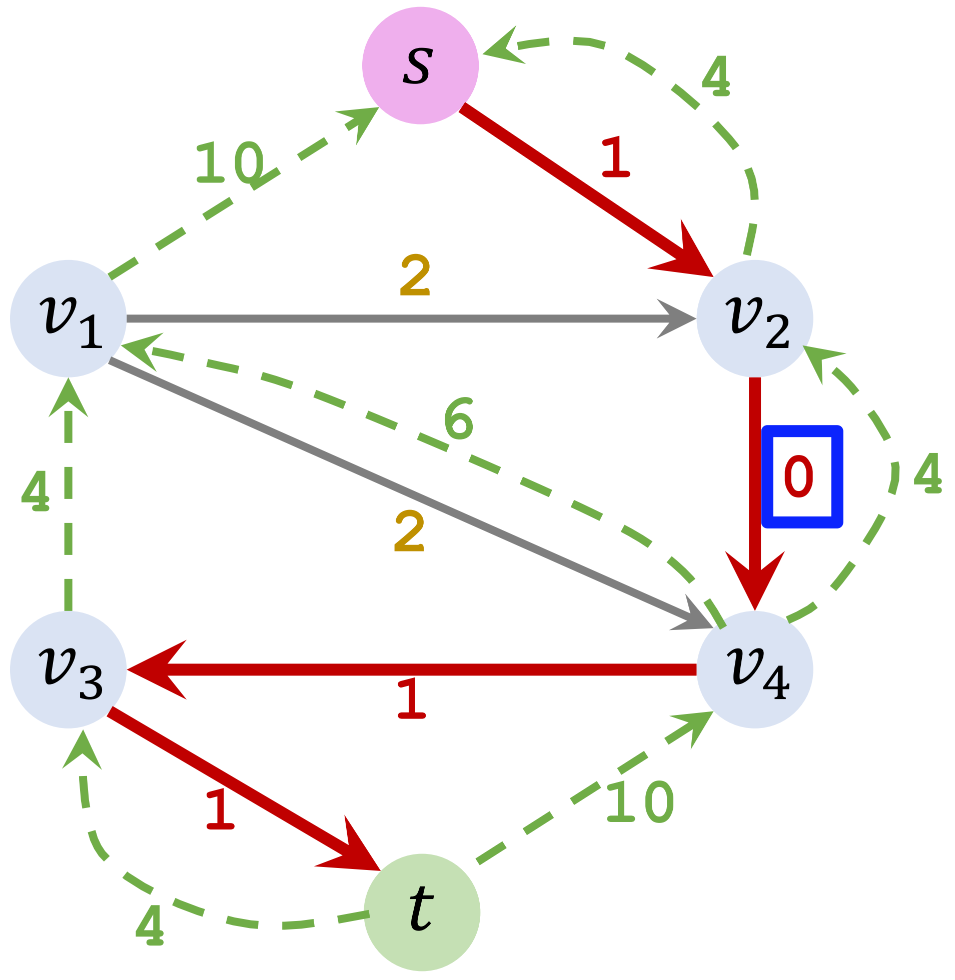


Blocking Flow

Old Residual Graph

Iteration 2: Update the residual graph

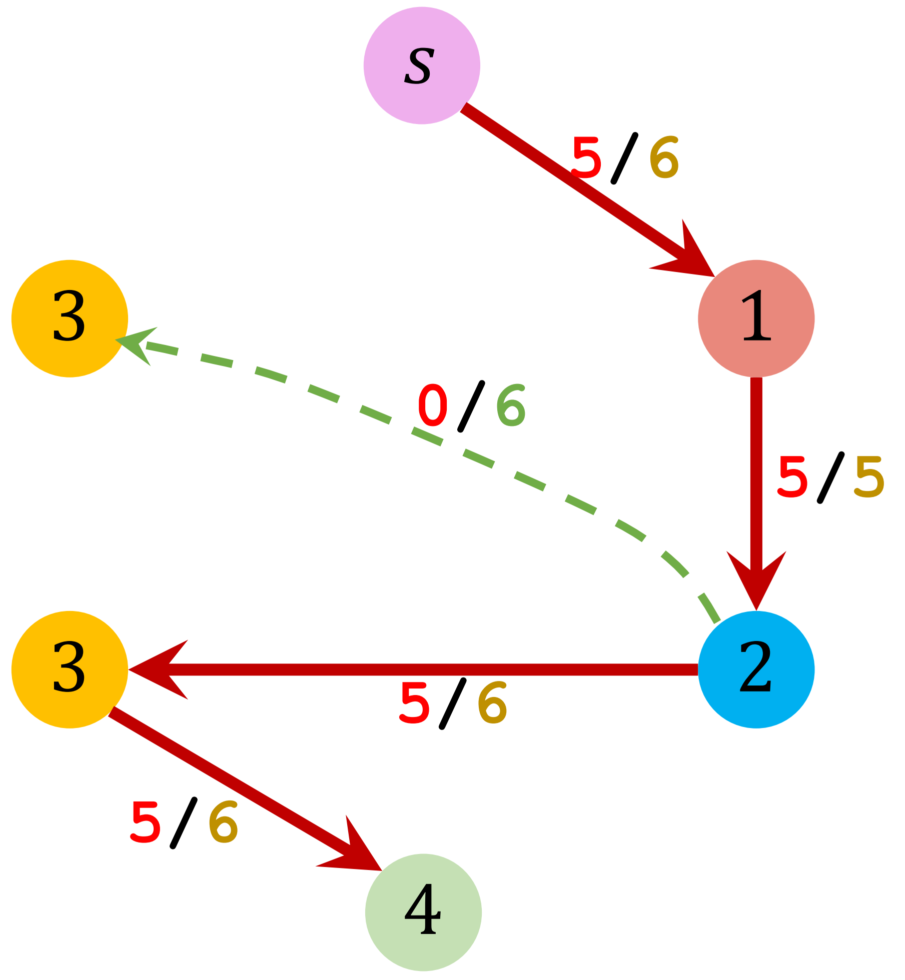


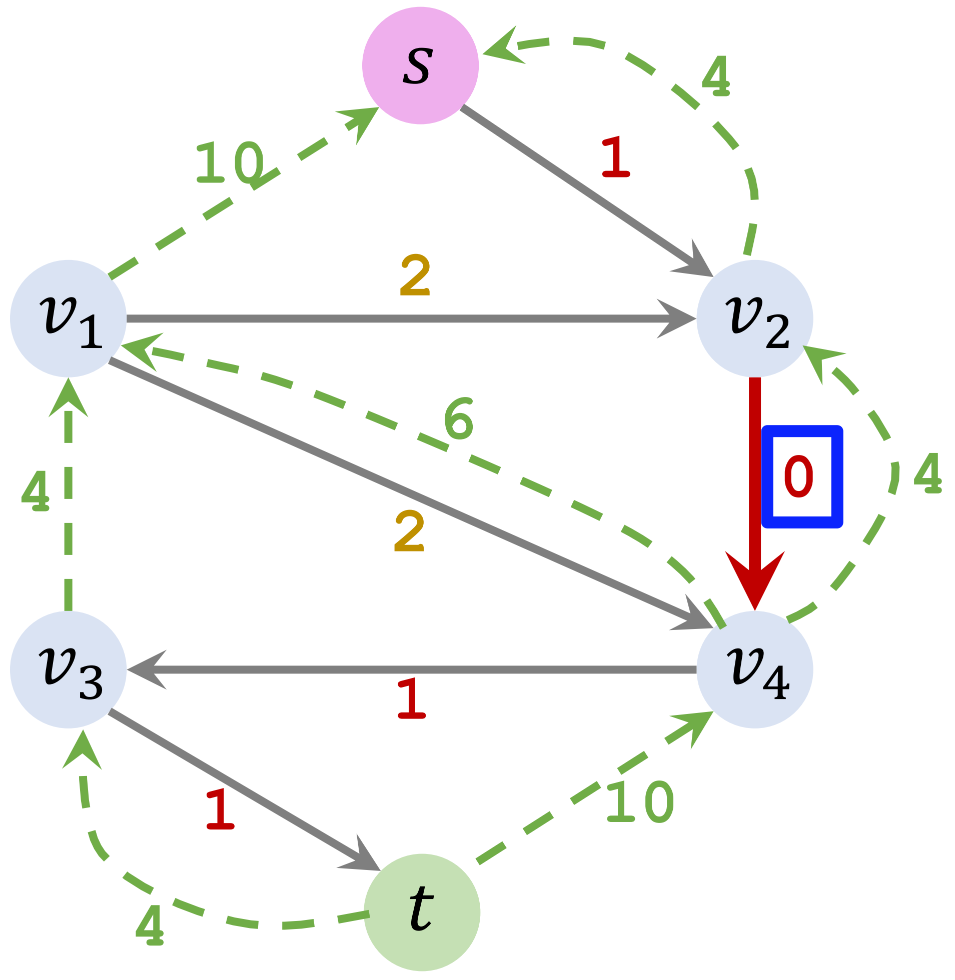


Blocking Flow

New Residual Graph

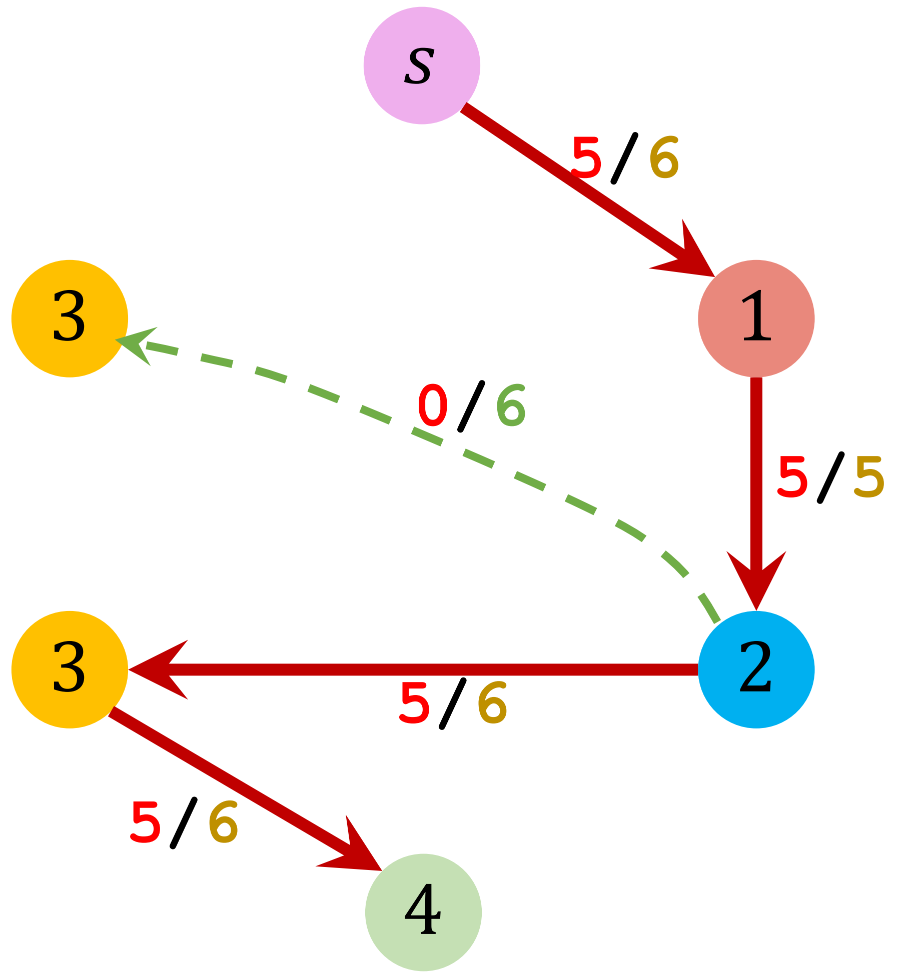
Iteration 2: Update the residual graph

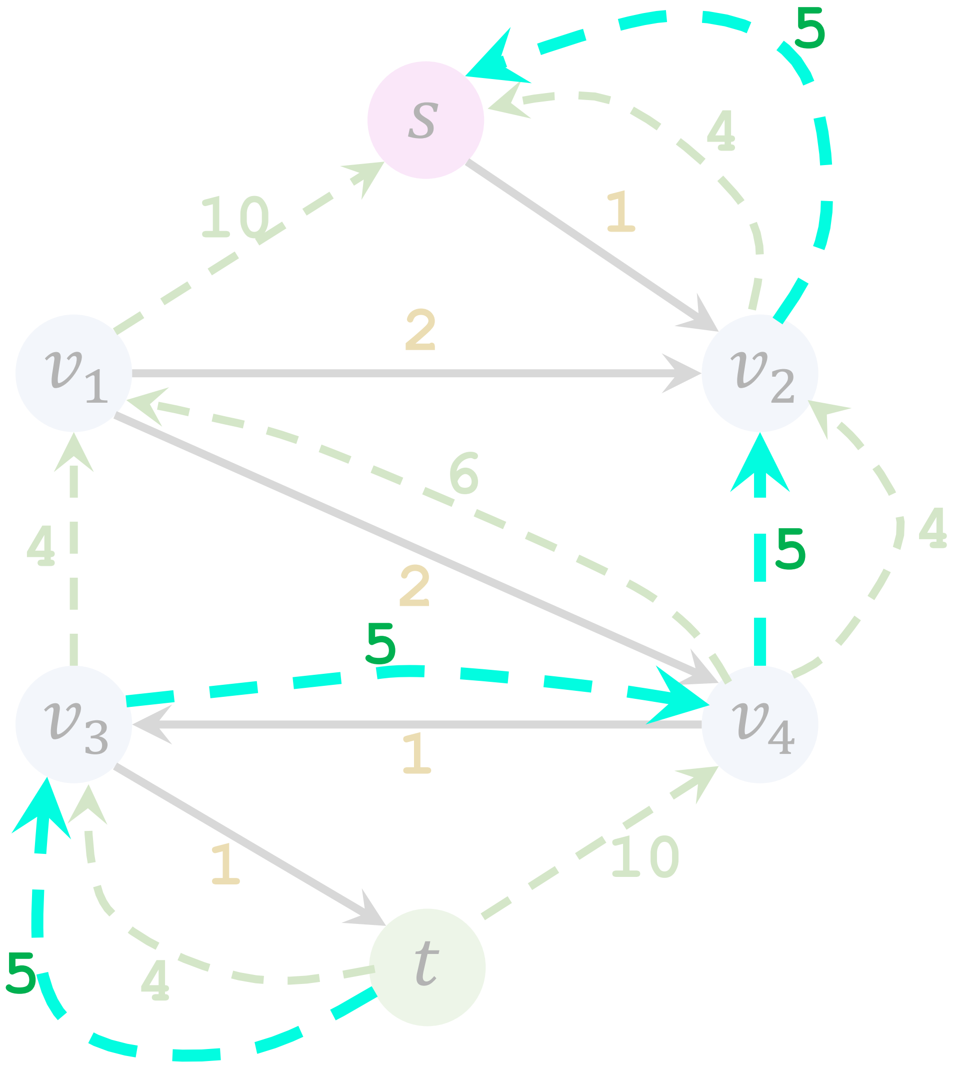




Removed saturated edges from residual graph.

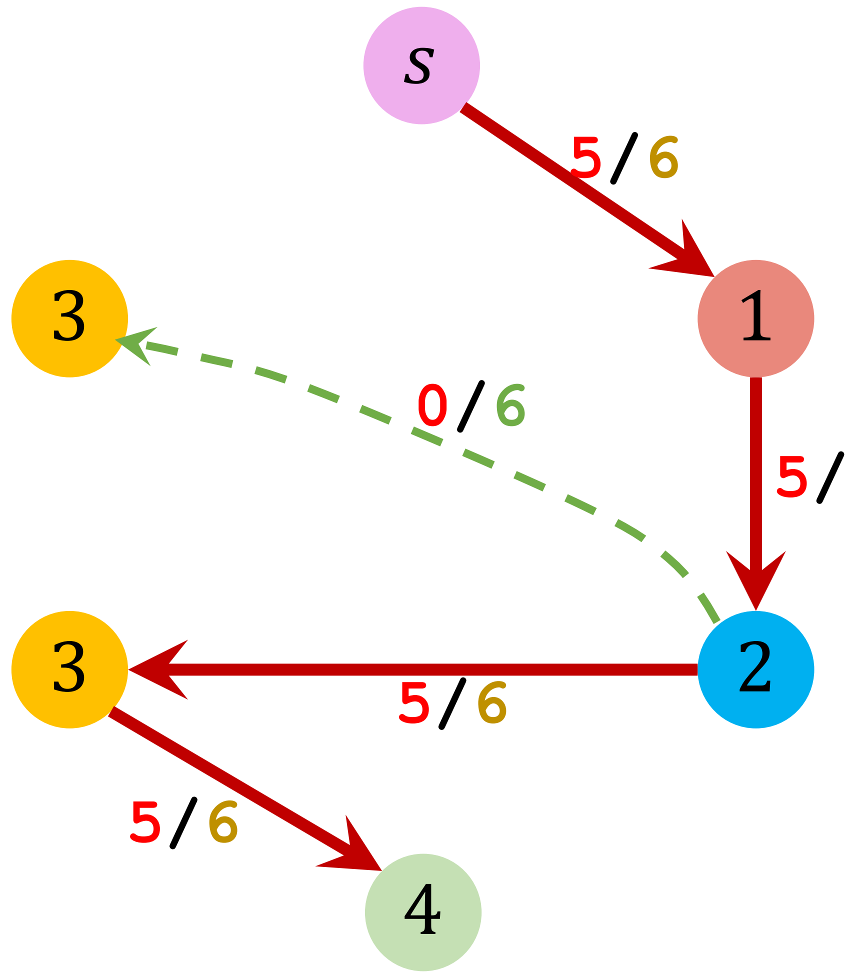
Iteration 2: Update the residual graph

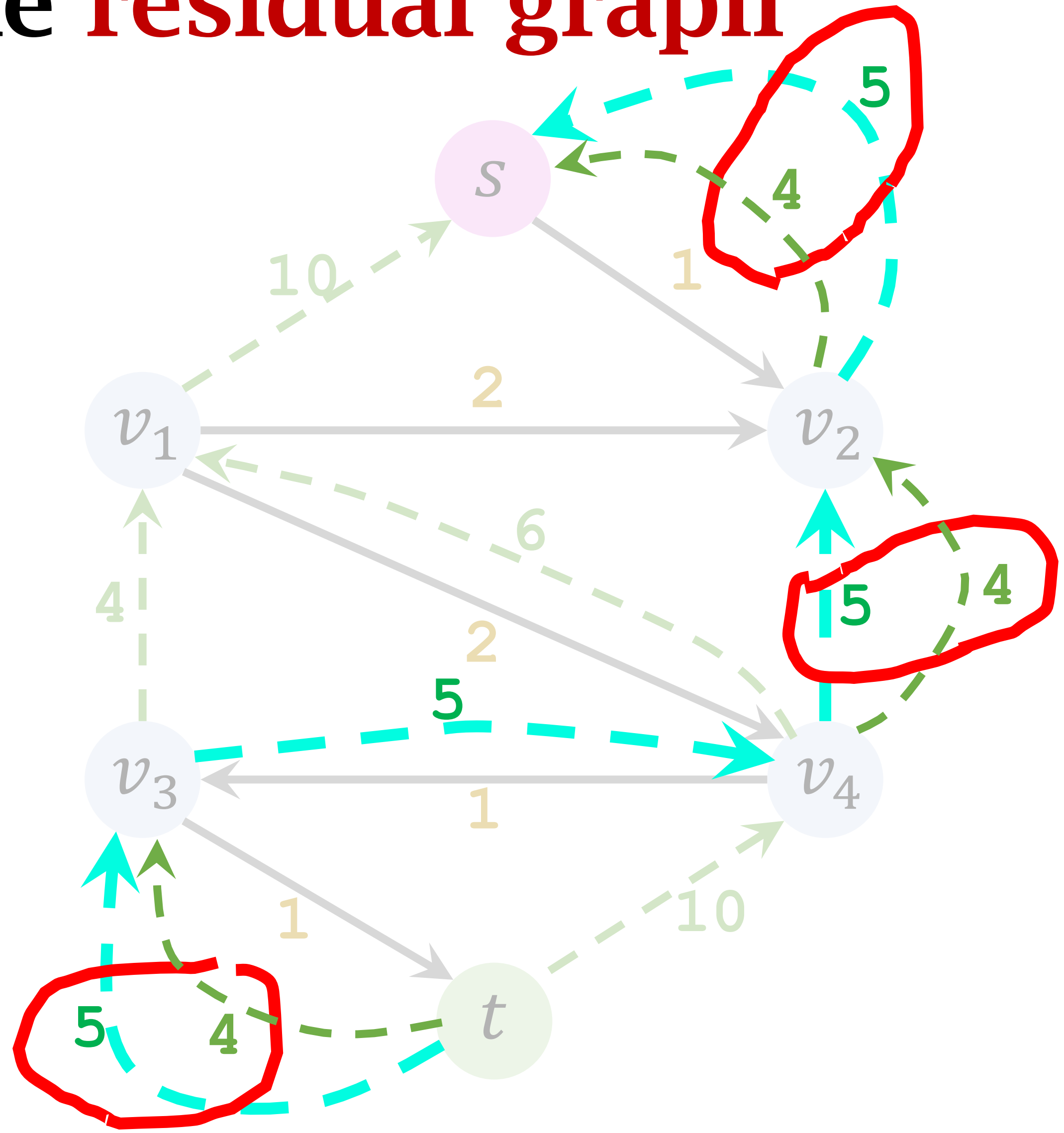




Add flows to the residual graph as backward paths.

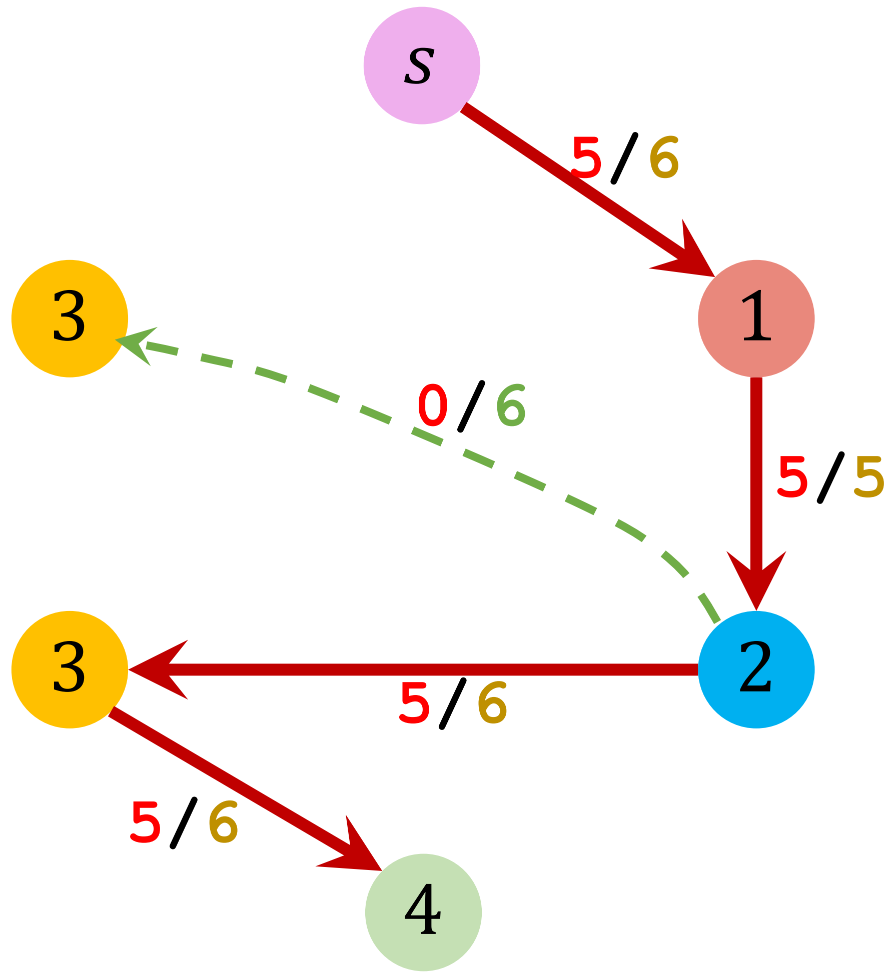
Iteration 2: Update the **residual graph**

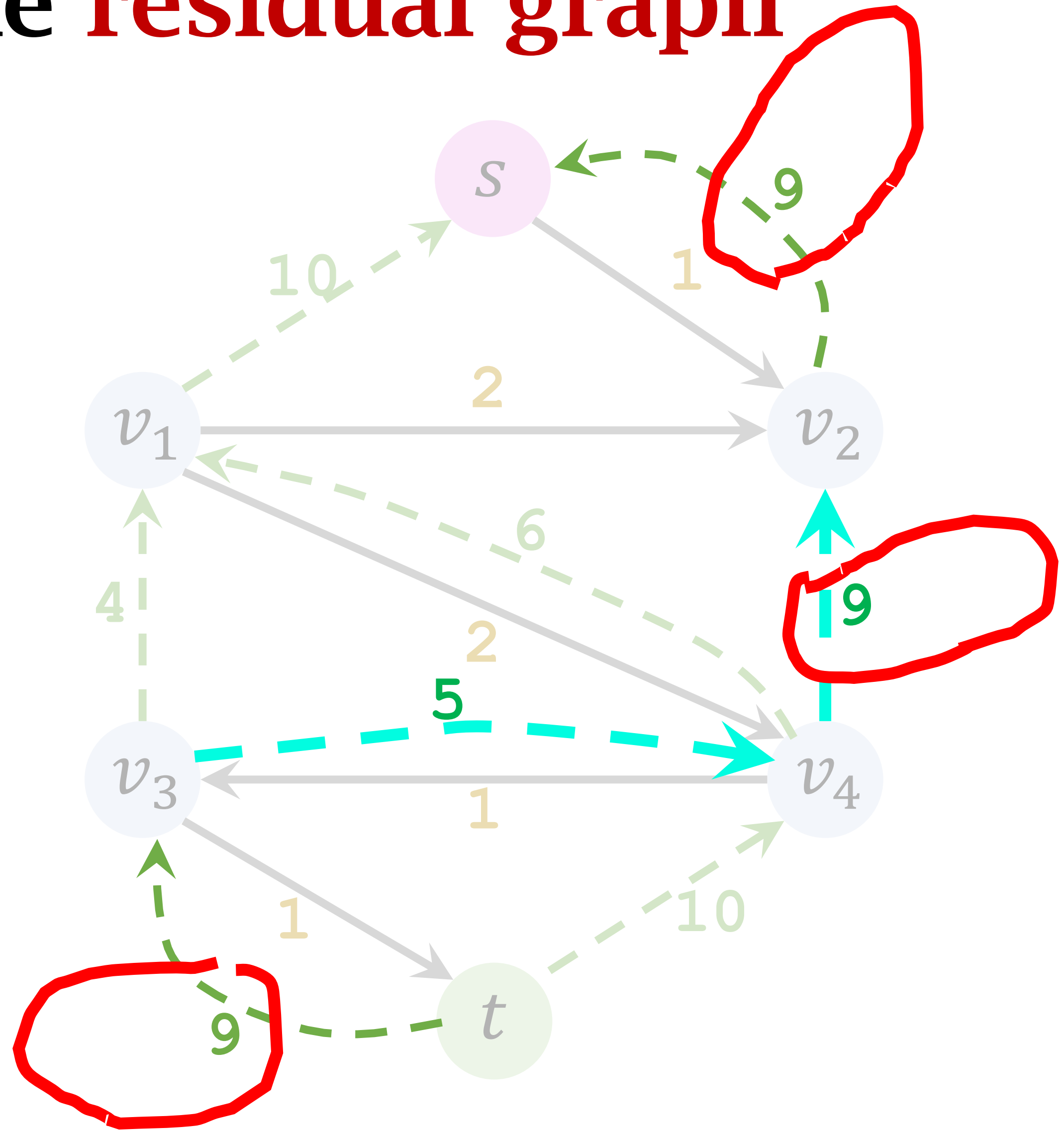




Merge Edges

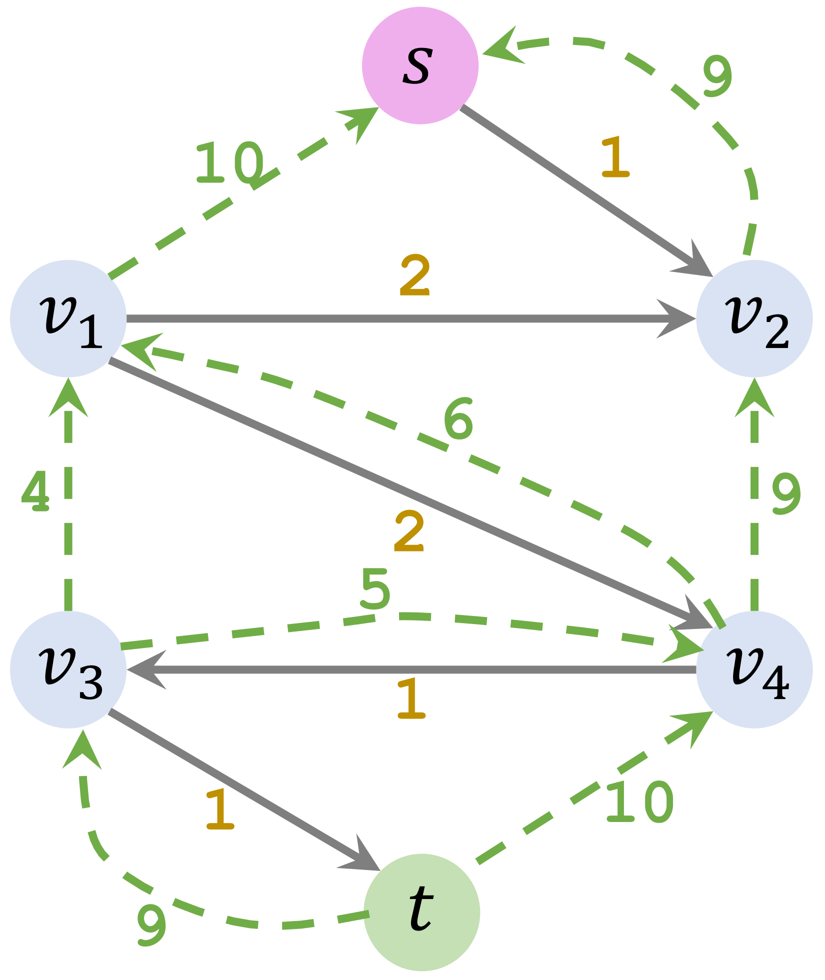
Iteration 2: Update the residual graph





Merge Edges

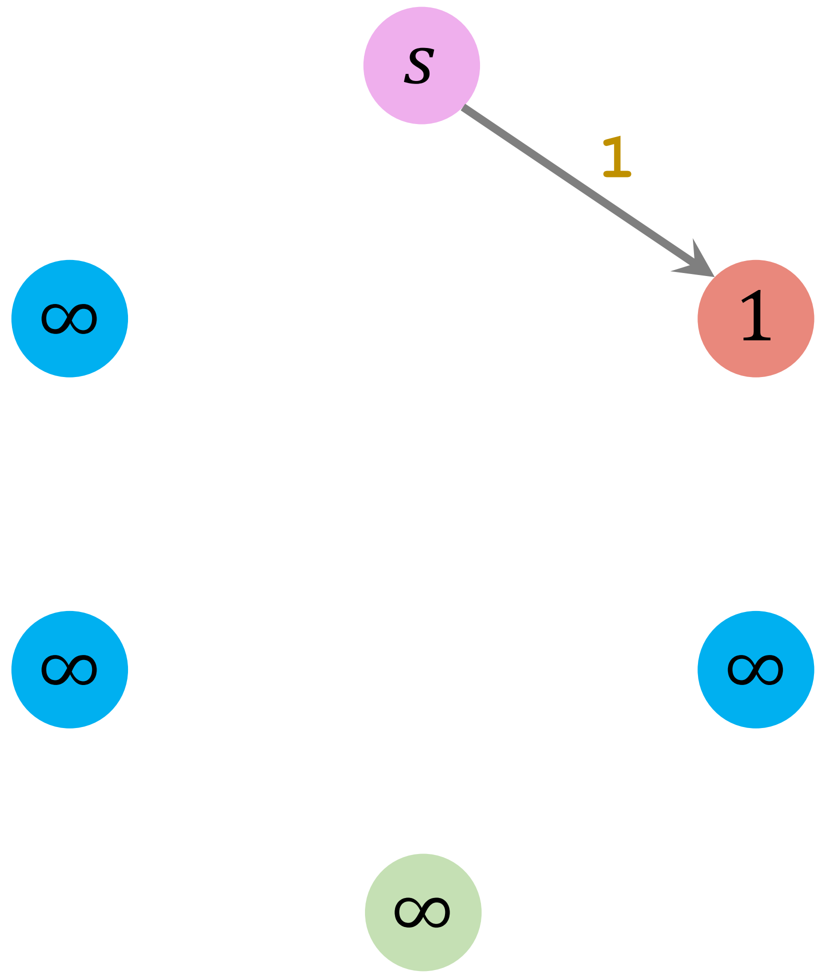
Iteration 3: Construct level graph



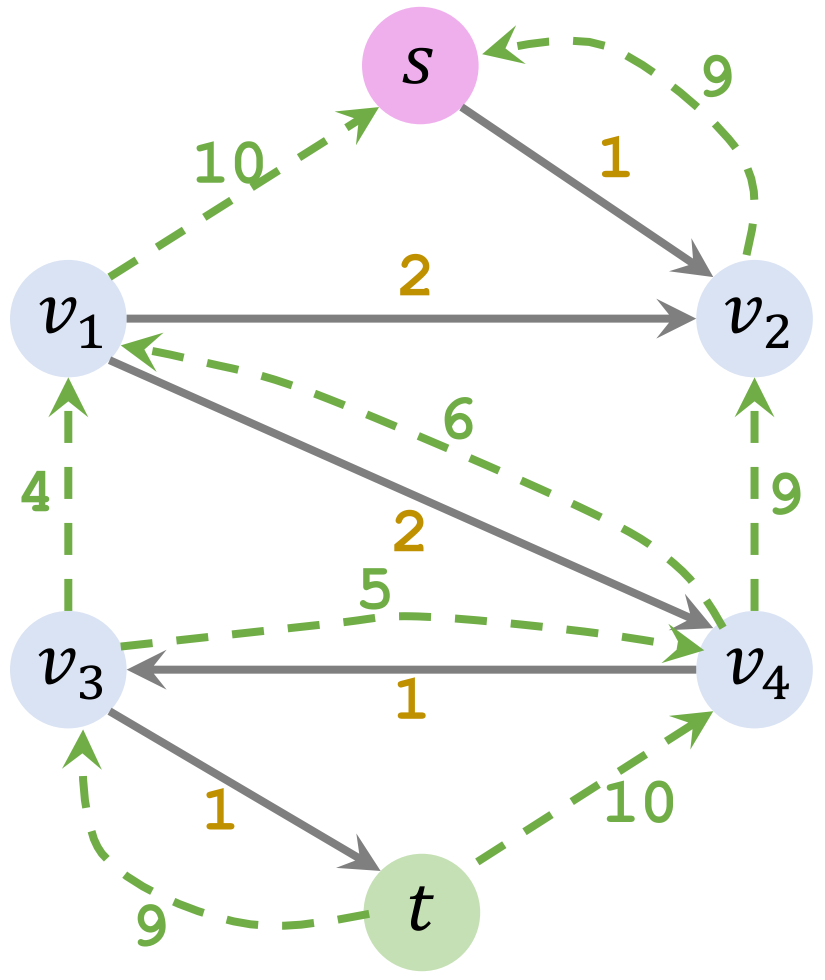
Level Graph

Residual Graph

Iteration 3: Construct level graph

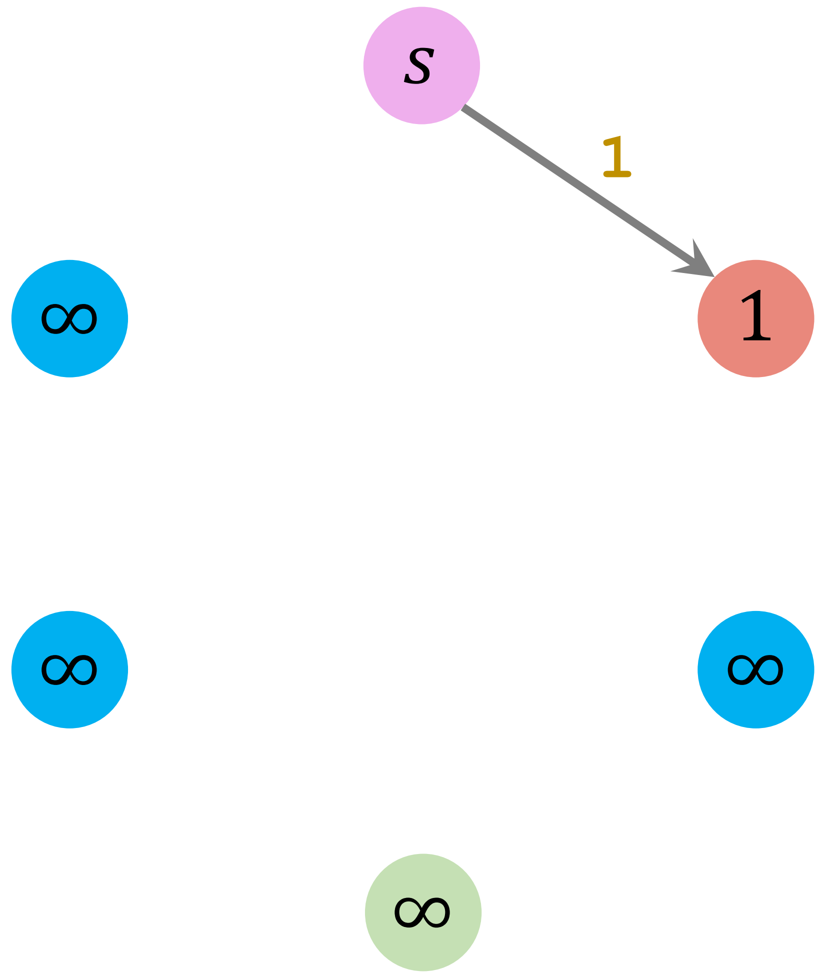


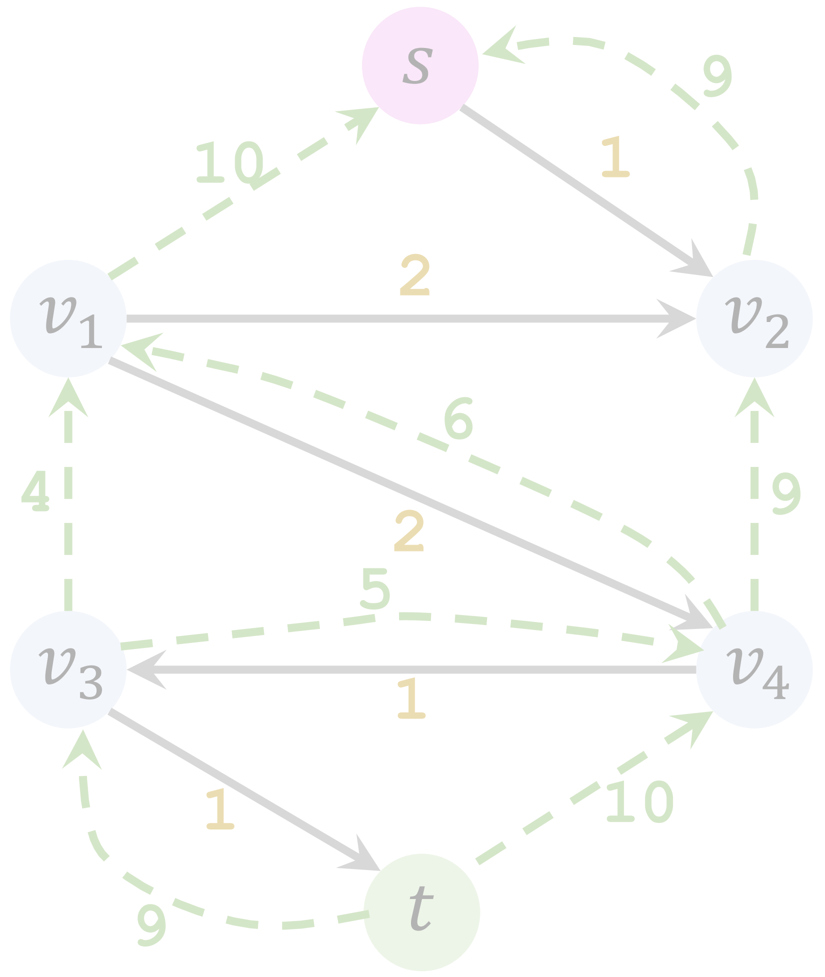
Level Graph



Residual Graph

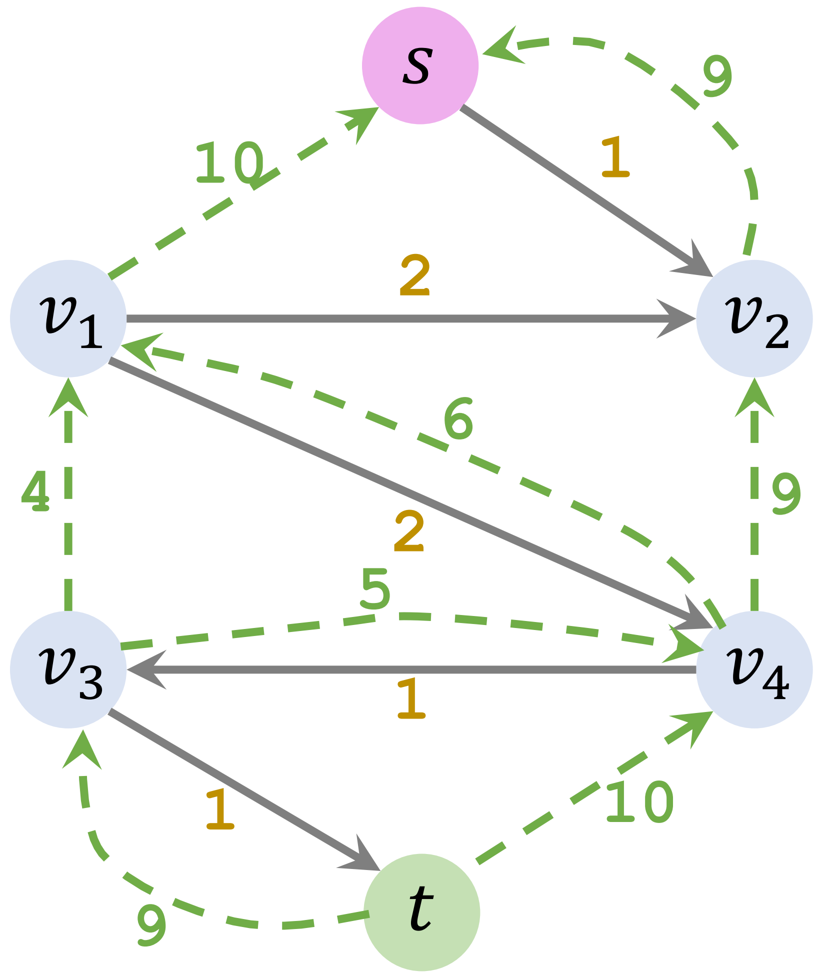
Iteration 3: Find blocking flow in level graph





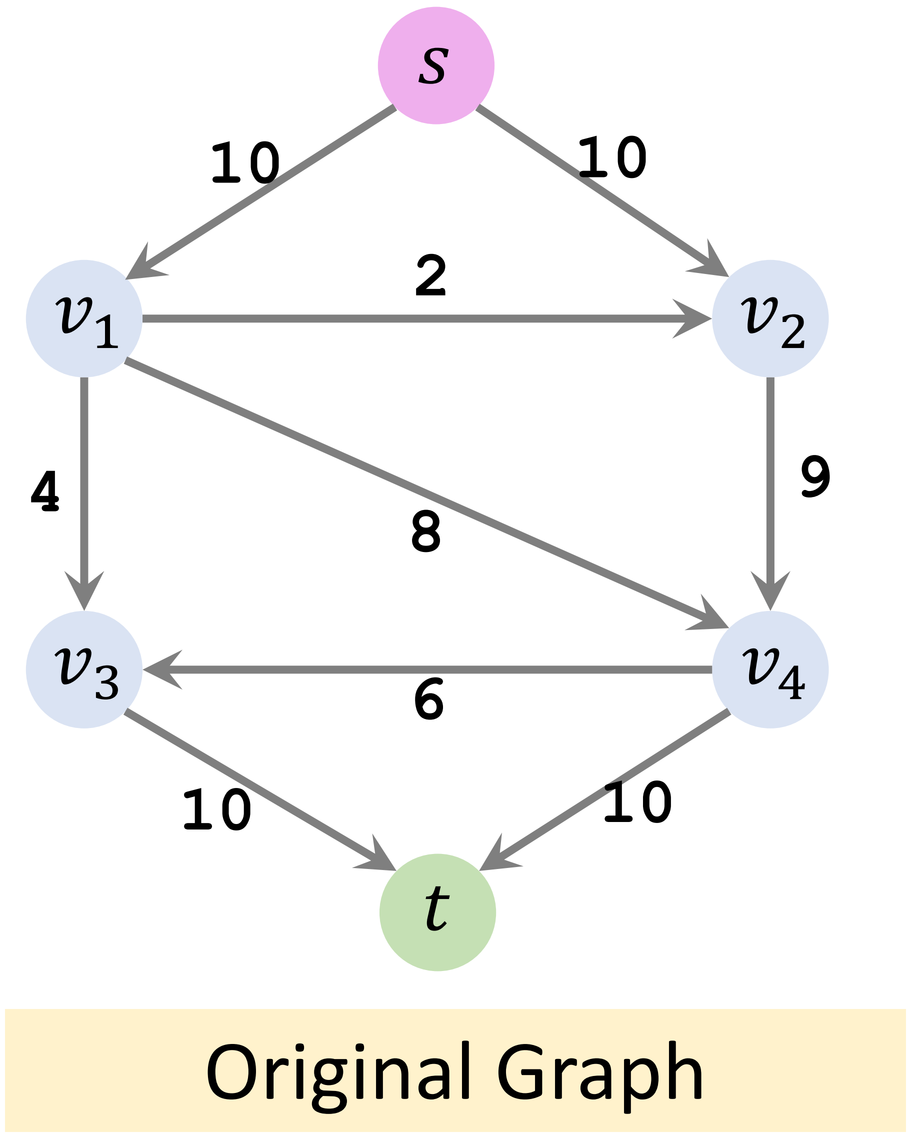
No flow can be found in the level graph.

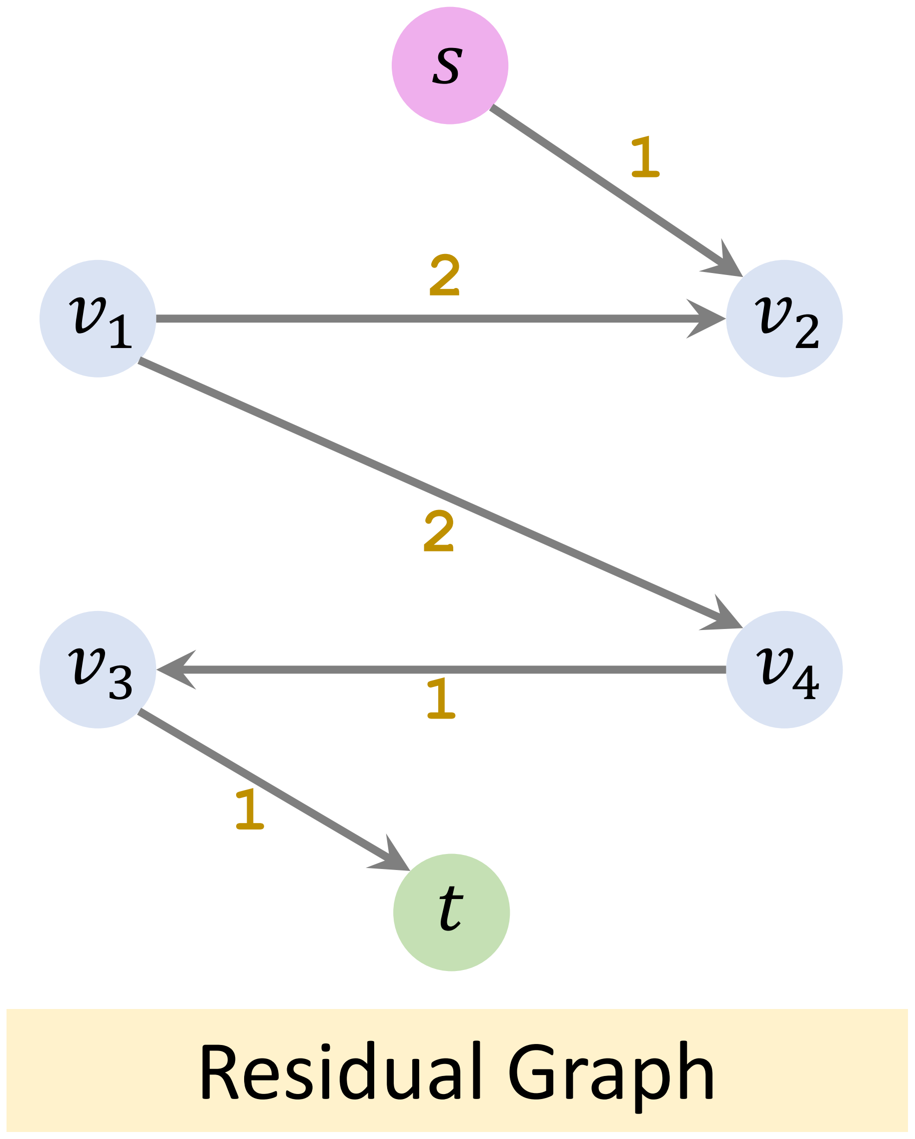
End of Procedure



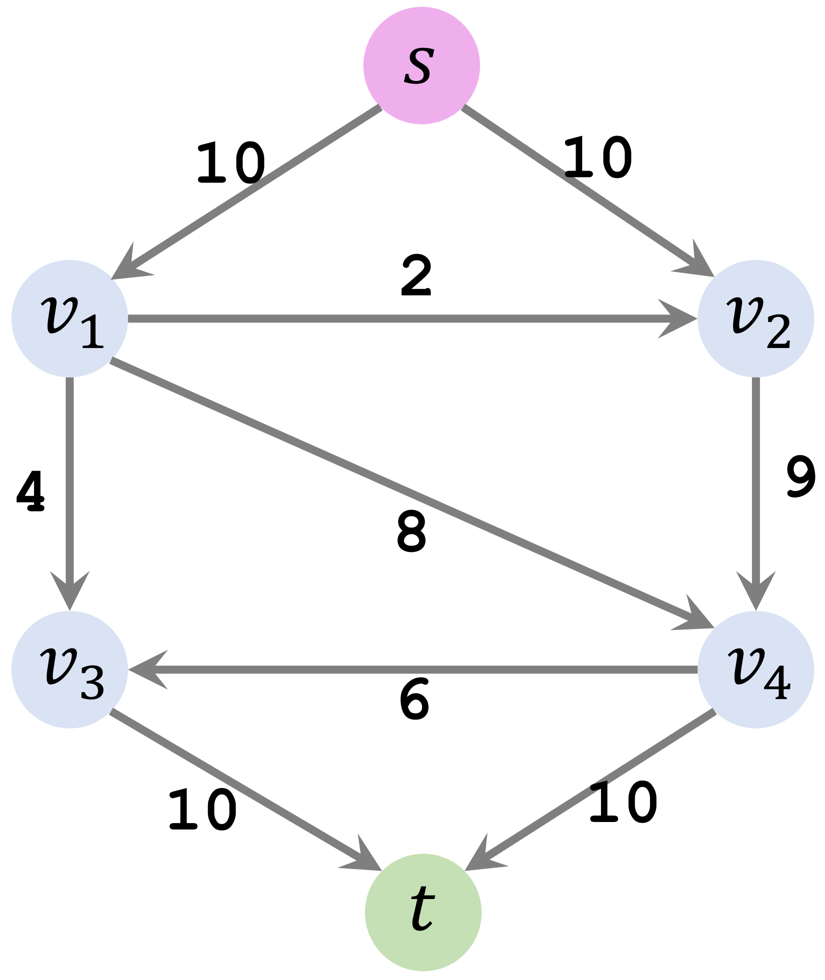
Residual Graph

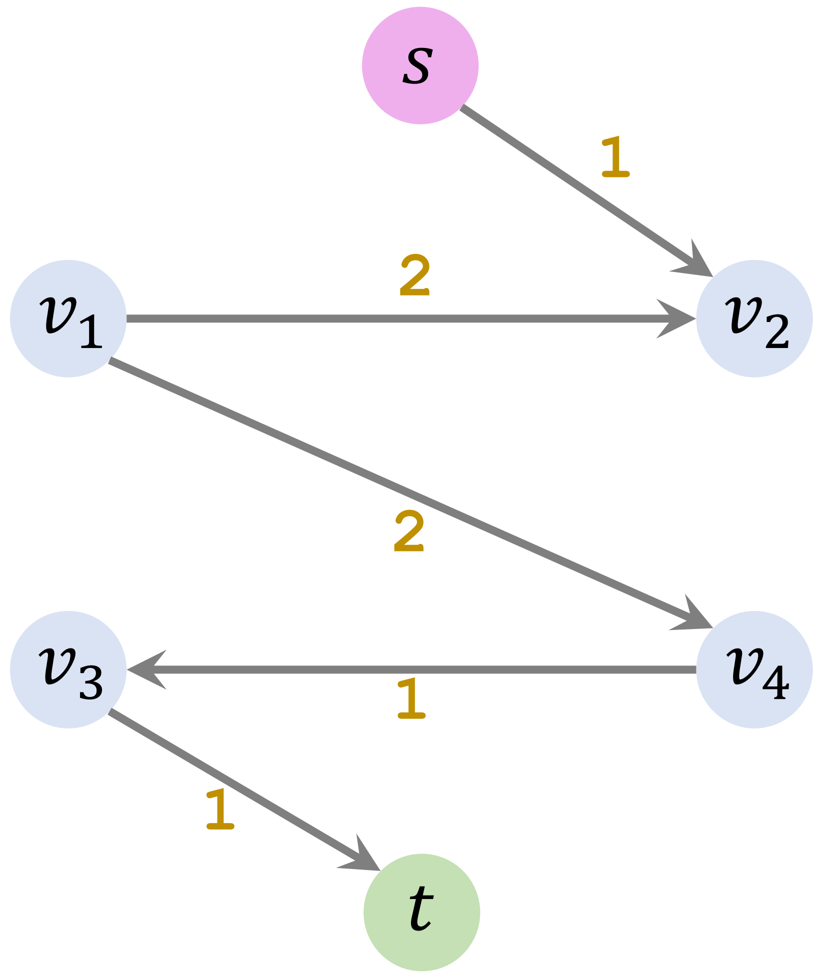
End of Procedure





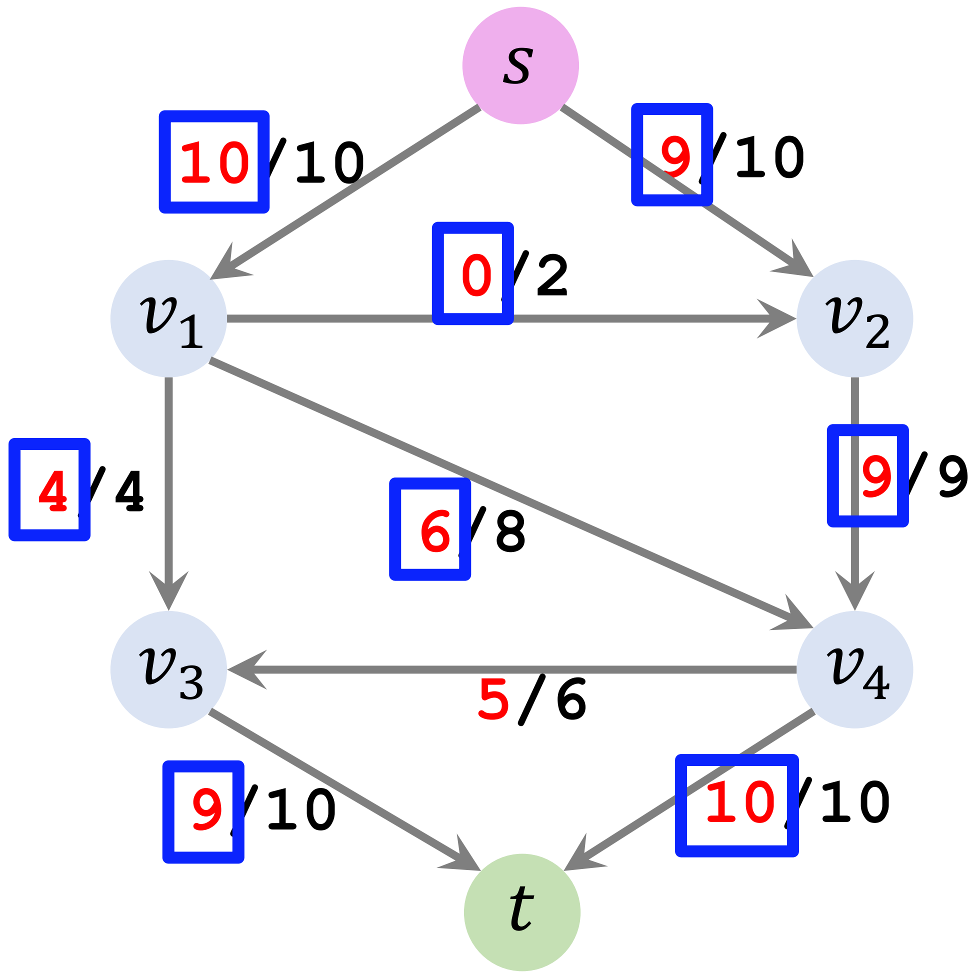
End of Procedure

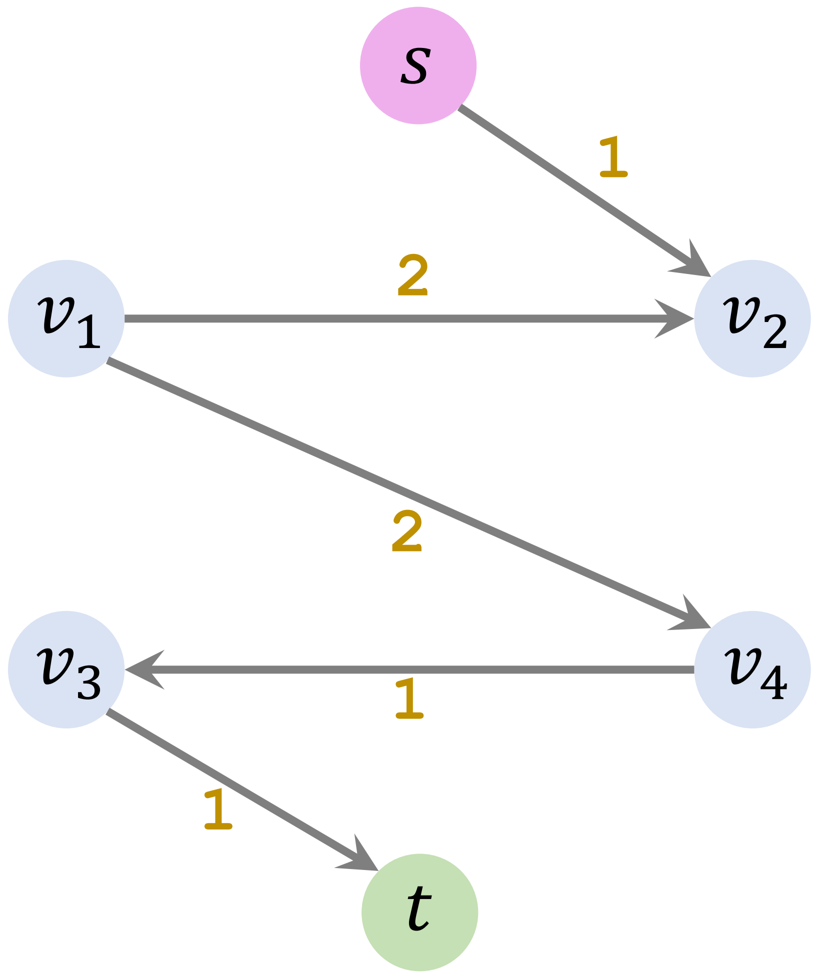




Flow = Capacity − Residual.

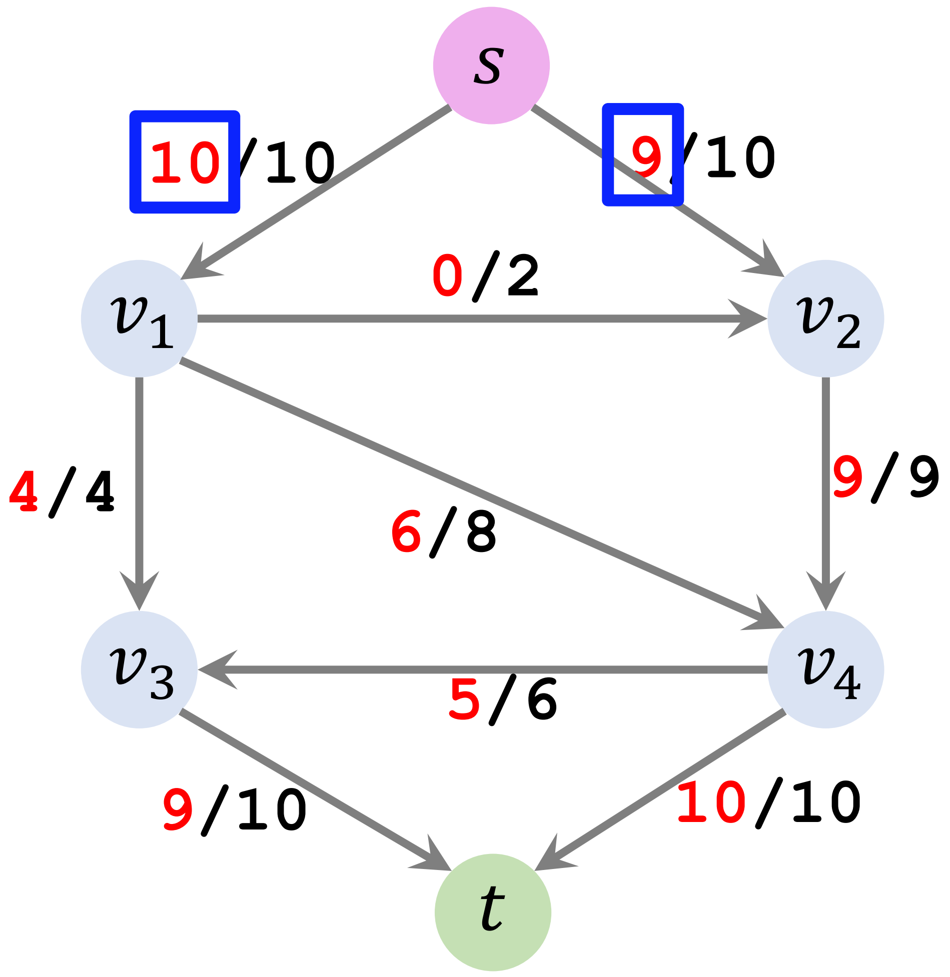
End of Procedure

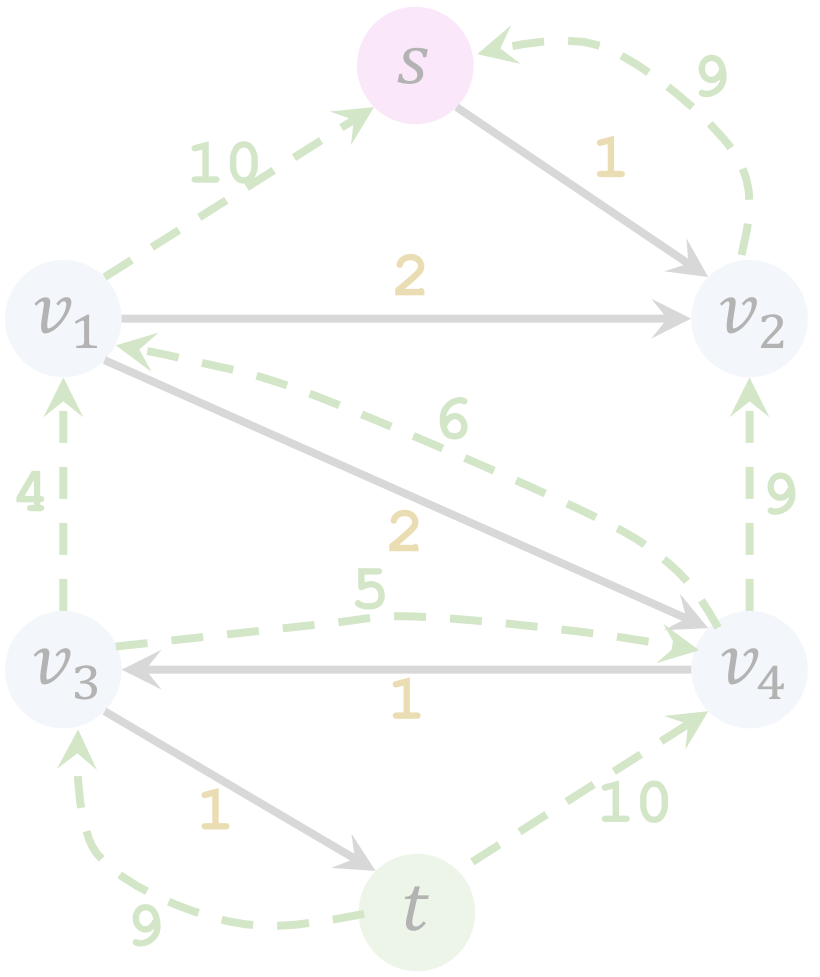




Flow = Capacity − Residual.

End of Procedure





Max Flow = 19. (Why? The flow leaving the source sum to 19.)

Dinic’s Algorithm

1. Build the residual graph which is a copy of the original graph.

Dinic's Algorithm

1. Build the residual graph which is a copy of the original graph.
2. Repeat:
    a. Construct the level graph of the residual graph.
    b. Break if the level graph has no path from source to sink.
    c. Find a blocking flow on the level graph.
    d. Update the residual graph (update the weights, remove saturated edges, and add backward edges.)

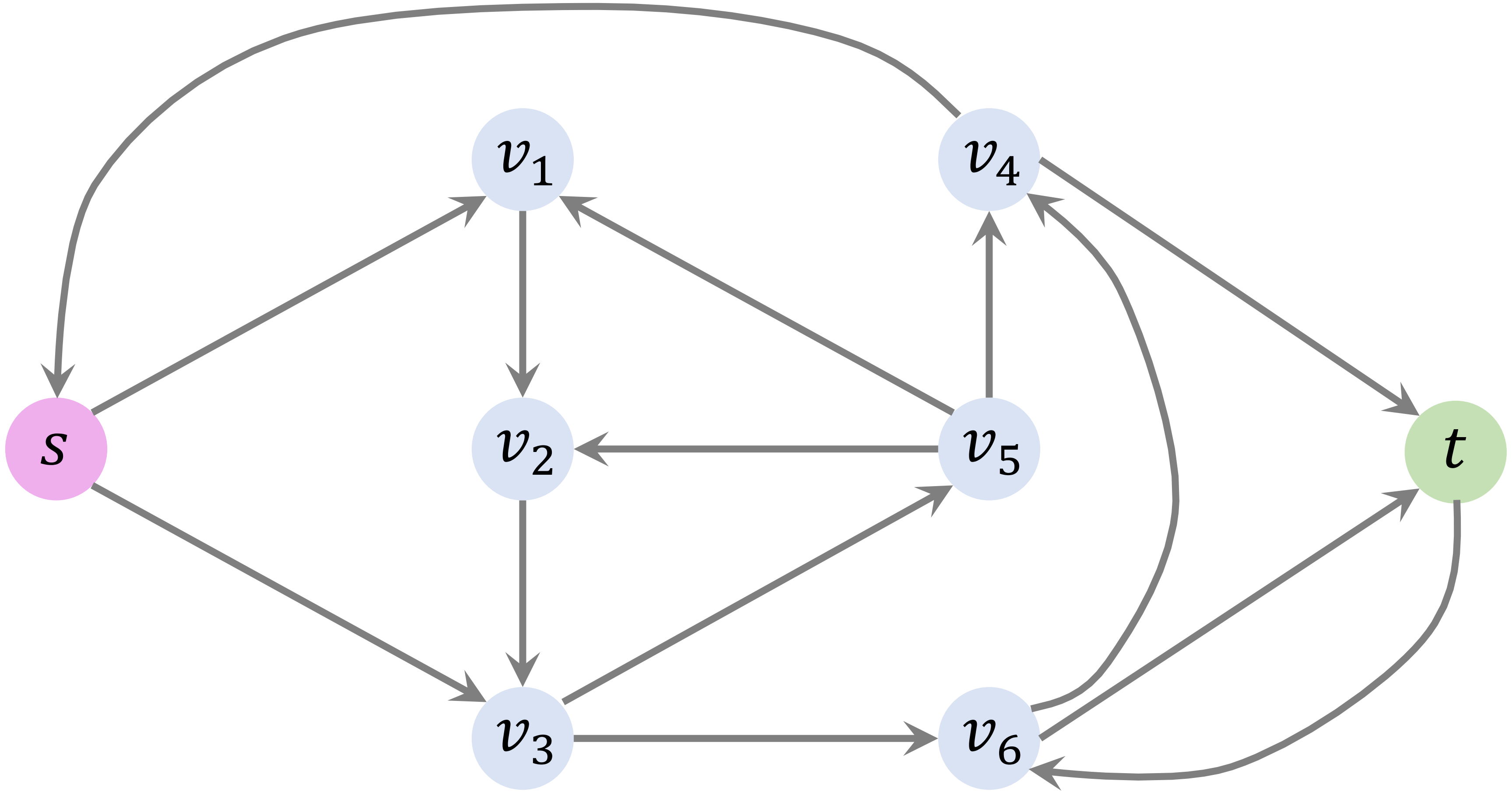
Time Complexity

Time complexity: $O(m \cdot n^2)$. (m is #edges; n is #vertices.)

- Dinic's algorithm has at most $n - 1$ iterations.
- Per-iteration time complexity is $O(mn)$.

Questions

Q1: What is the level graph?



Thank You!