CS4234
Optimiz(s)ation Algorithms

L7 – (Weighted) Max-Cardinality -(Bipartite)-Matching, round 2

https://visualgo.net/en/matching

This course material is now made available for public usage.
Special acknowledgement to School of Computing, National University of Singapore
for allowing Steven to prepare and distribute these teaching materials.

CS3233/CS4234
Dual Slides ☺

Dr. Steven Halim

Social
Development
Network

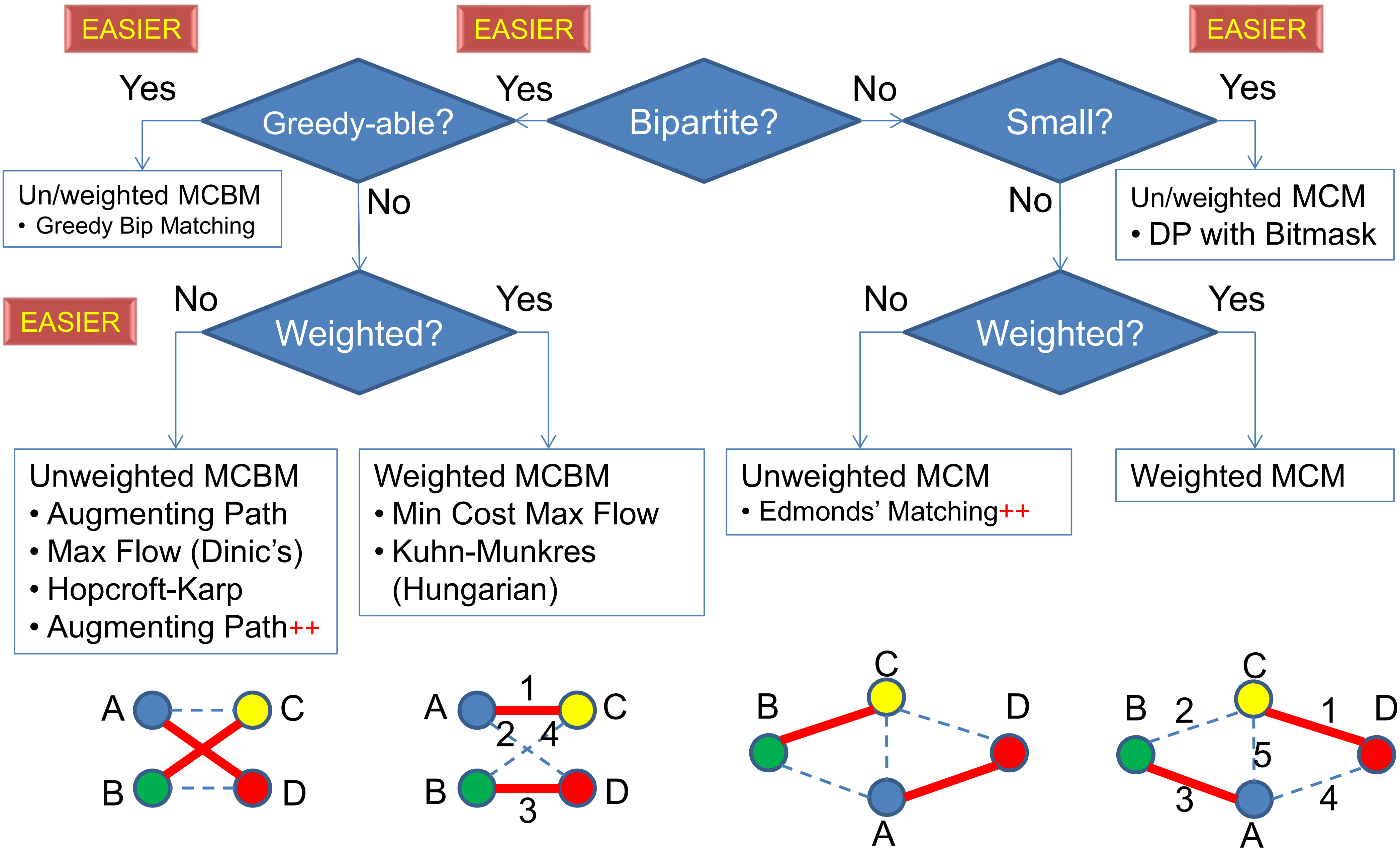
Roadmap (for CS4234) – Week 07

Graph Matching (after recess, **not part of Midterm test**)

- Weighted MCBM: (Min/Max) Cost (Max) Flow, Hungarian (Kuhn-Munkres algorithm)
- Unweighted MCM: Edmonds' Matching (**overview**)
- Weighted MCM: DP with Bitmask **(small graph only, review…)**
  - This DP with Bitmask solution will also solve other variants, but only if they are posed on small ($V \leq 20$) graphs…
  - Still unable to make it work for applying the Christofides's 1.5-Approximation algorithm for large instances of M-R/NR-TSP as of year 2021 ☹
    - PS: Now 1.5-e, see https://www.quantamagazine.org/computer-scientists-break-traveling-salesperson-record-20201008/

https://www.youtube.com/watch?v=GhjwOiJ4SqU&feature=youtu.be&t=2m54s cites CP3

Types of Graph Matching



Solutions (slightly deeper this AY):

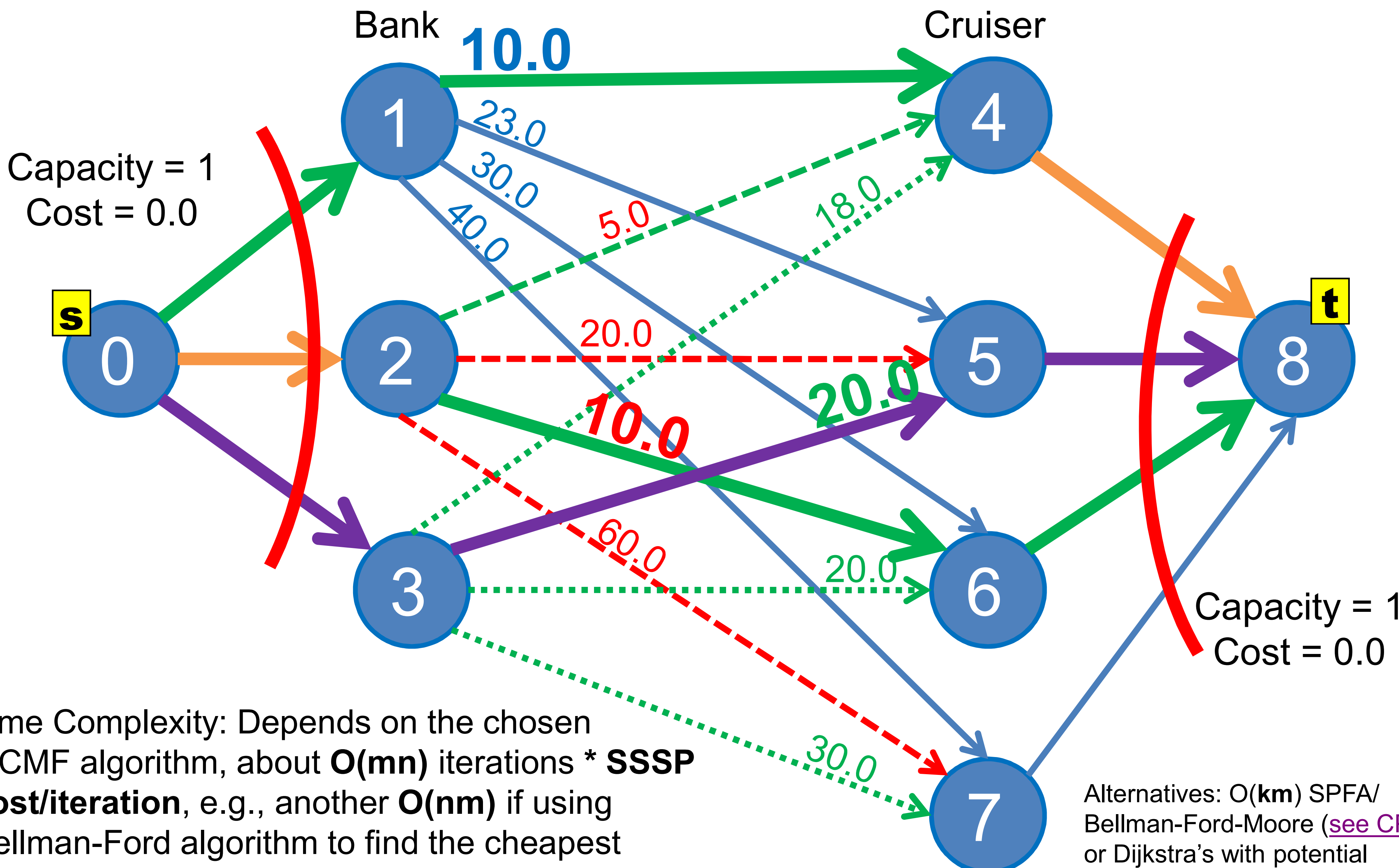
Min/Max Cost (Max) Flow, CP4 Book 2 Section 9.25

Kuhn-Munkres (Hungarian), CP4 Book 2 Section 9.27

WEIGHTED MCBM

UVa 10746 (last state)

Min Cost so far =
0 + **5.0** + 0 +
0 + 10.0 **- 5.0** + 10.0 + 0 +
0 + 20.0 + 0 = 40.0



Time Complexity: Depends on the chosen MCMF algorithm, about **O(mn)** iterations * **SSSP cost/iteration**, e.g., another **O(nm)** if using Bellman-Ford algorithm to find the cheapest augmenting path…, so **O(n^2m^2)** (quite bad…)

Alternatives: O(**km**) SPFA/ Bellman-Ford-Moore (see CP4) or Dijkstra's with potential (I am not familiar yet)

Kuhn-Munkres (Hungarian) Algorithm

Harold Kuhn (1955) and James Munkres (1957) name their (joint) algorithm based on the work of two other **Hungarian** mathematicians (Denes Konig + Jano Egervary)

- There is a graph version and matrix version (we discuss the graph version)

Initial implementation $O(n^4)$; today's best version $O(n^3)$

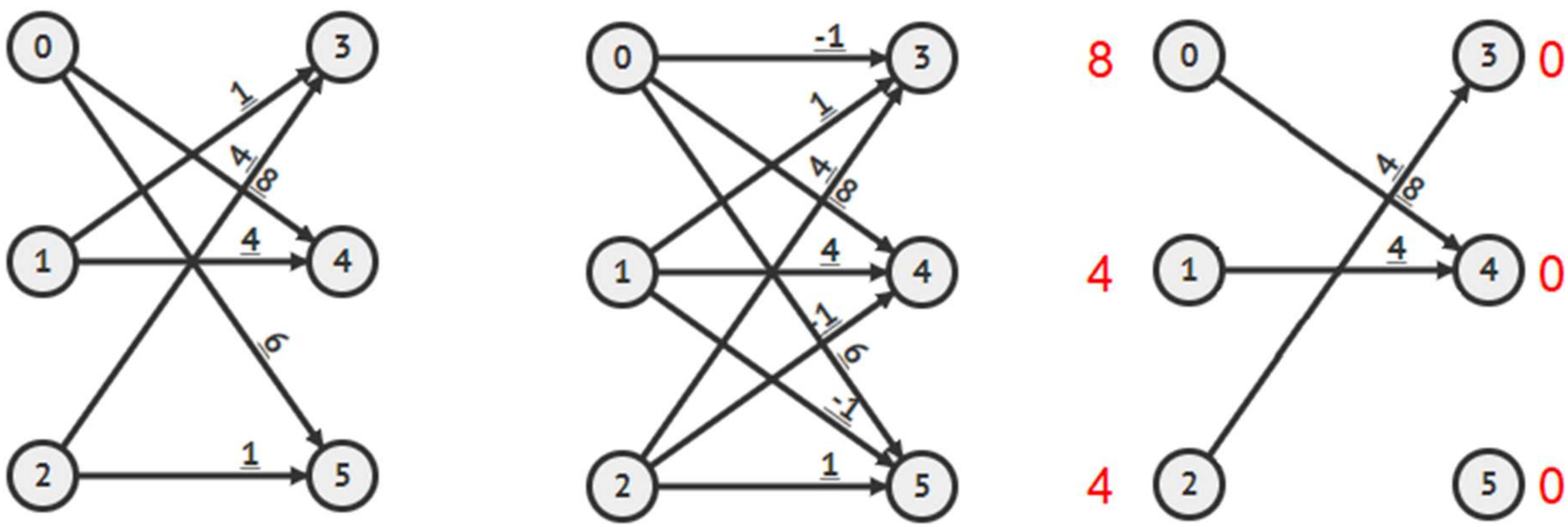
Default version: For <u>Max Weighted Perfect</u> Bipartite Matching

- But can easily be modified to support Min Weighted (negate edge weights) and for Bipartite Graph where Perfect Matching is impossible (add dummy vertices/edges with irrelevant weights)

An explanation using CP4 Book 2 drawings

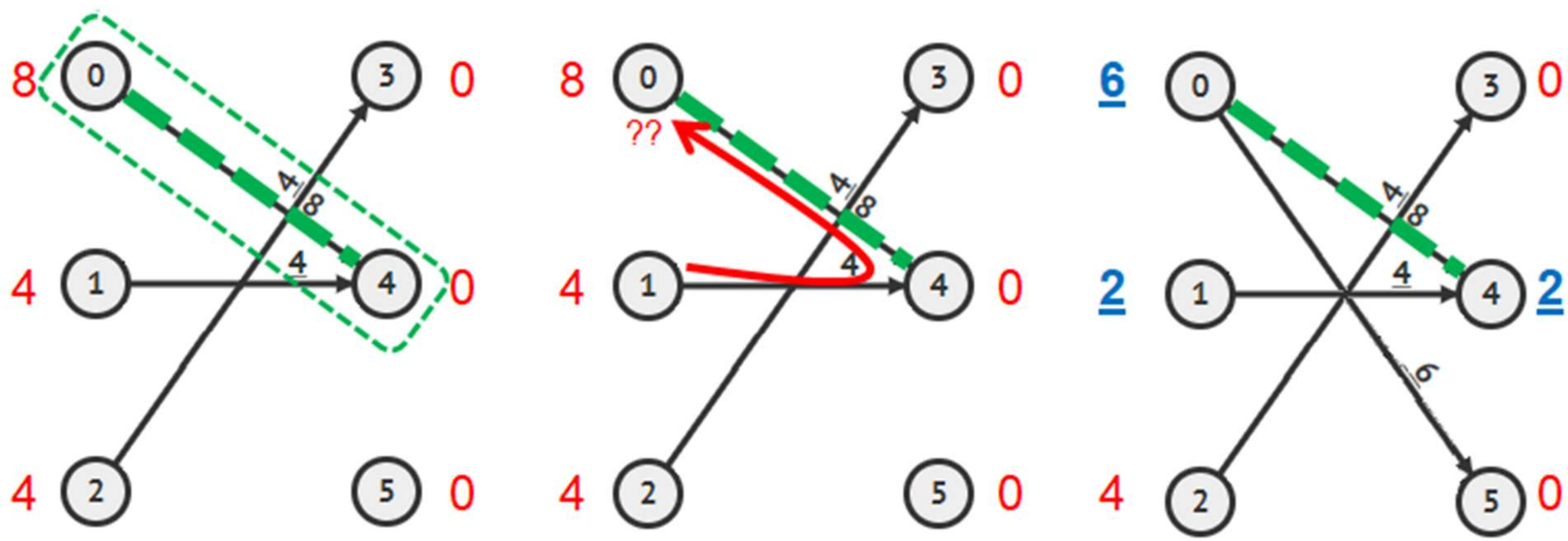
L: Initial Graph
M: Complete Weighted Bipartite Graph
R: Equality Subgraph

Review the recording or re-read CP4 Book 2 for the explanation



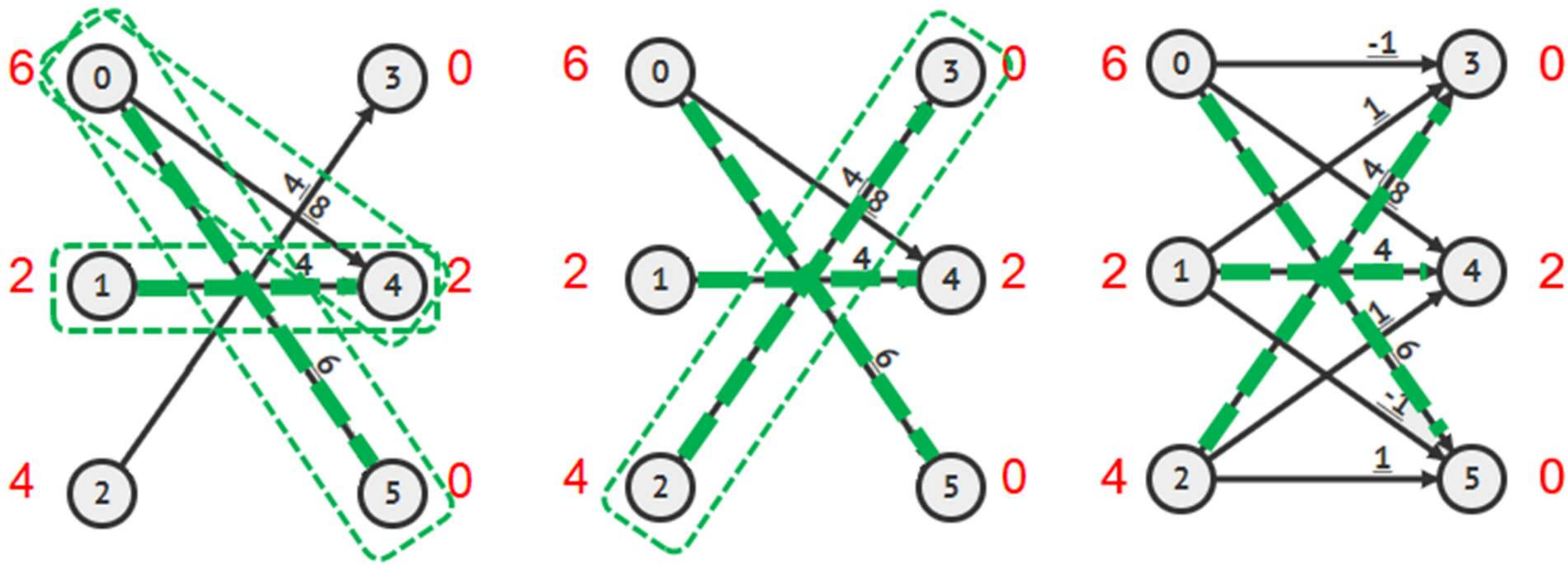
L: 1st Augmenting Path
M: Stuck
R: Relabel the Equality Subgraph

Review the recording or re-read CP4 Book 2 for the explanation

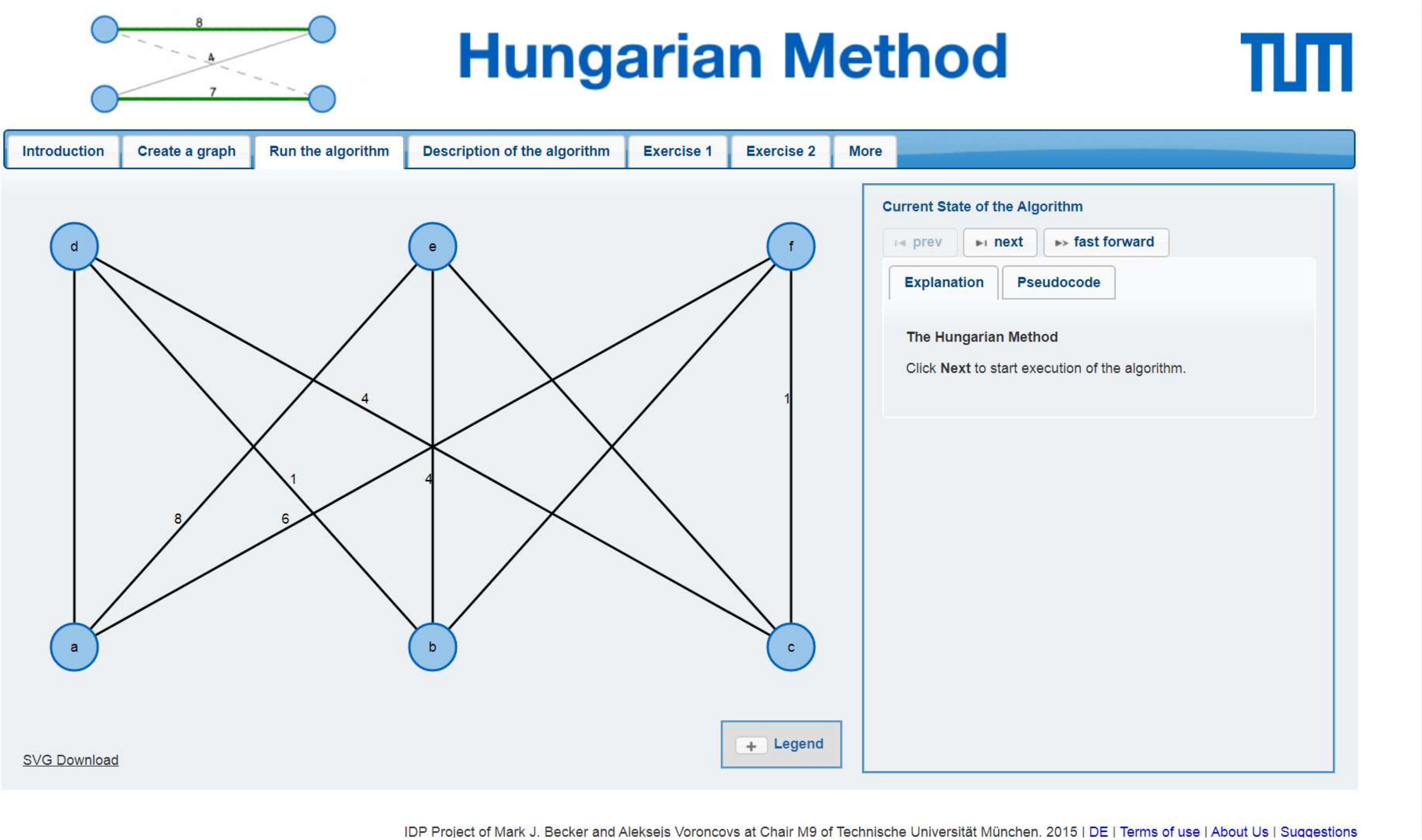


L: 2nd Augmenting Path
M: 3rd Augmenting Path
R: Max Weighted Perfect Matching

Review the recording or re-read CP4 Book 2 for the explanation



Let’s See



This tool https://algorithms.discrete.ma.tum.de/graph-algorithms/matchings-hungarian-method/index_en.html is on maximization problem on a **complete bipartite** graph (so that Perfect Matching exists)

The layout is top row = right set and bottom row = left set

Kuhn-Munkres (Hungarian) Algorithm

A good Hungarian algorithm implementation runs in O(**V^3**), thus it is a much better algorithm for **Weighted MCBM** problem compared to (the more general) MCMF

- You are allowed to quote this info verbatim in PS4/exam
  - Focus on the modeling of the *complete* weighted bipartite graph
  - And on whether it is a maximization or a minimization problem
- References:
  - https://github.com/jaehyunp/stanfordacm/blob/master/code/MinCostMatching.cc
  - https://e-maxx.ru/algo/assignment_hungary
    - https://translate.google.com/translate?hl=ru&sl=ru&tl=en&u=https%3A%2F%2Fe-maxx.ru%2Falgo%2Fassignment_hungary
  - https://brilliant.org/wiki/hungarian-matching/

Solutions (High Level Tour Only – but slightly deeper after each AY):

Edmonds’ Matching Algorithm, CP4 Book 2, Section 9.28

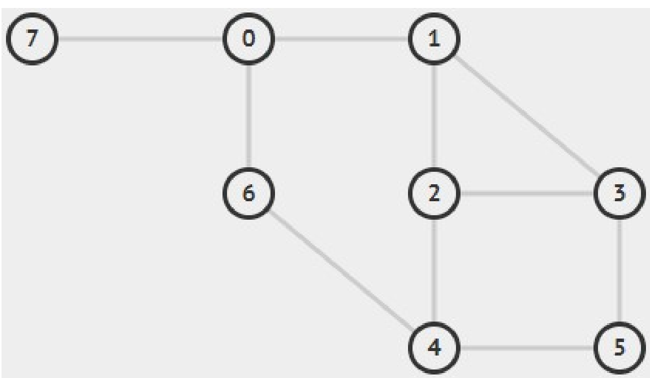
UNWEIGHTED MCM

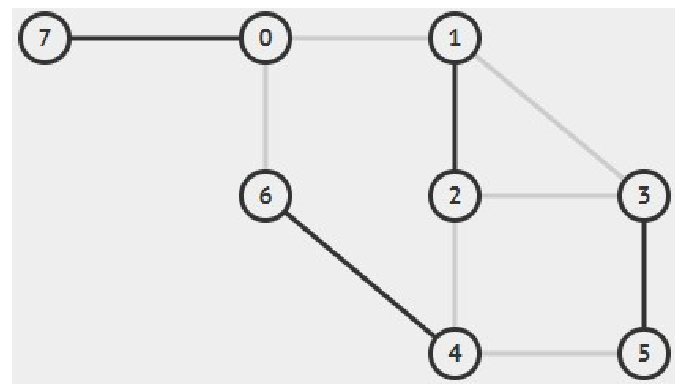
https://visualgo.net/en/matching, select "Unweighted General" tab

Non-Bipartite Graph and Blossom

A graph is not bipartite if it has at least one odd-length cycle

What is the MCM of this non-bipartite graph?





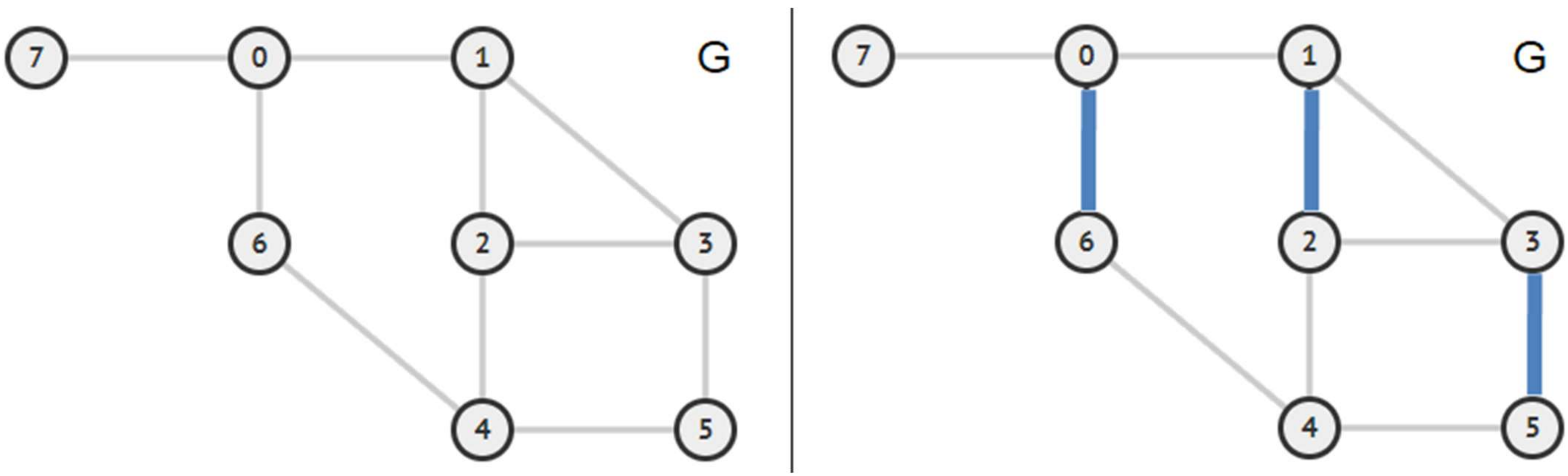
It is harder to find augmenting path (Berge's Lemma) in such graph due to alternating cycles called **blossoms** https://visualgo.net/en/matching, select “Unweighted General” tab

Blossom Shrinking/Expansion (1)

Review the recording or re-read CP4 Book 2 for the explanation

However, shrinking these blossoms (recursively) will make this problem “easy” again (with proper post-processing when the recursion unwinds)

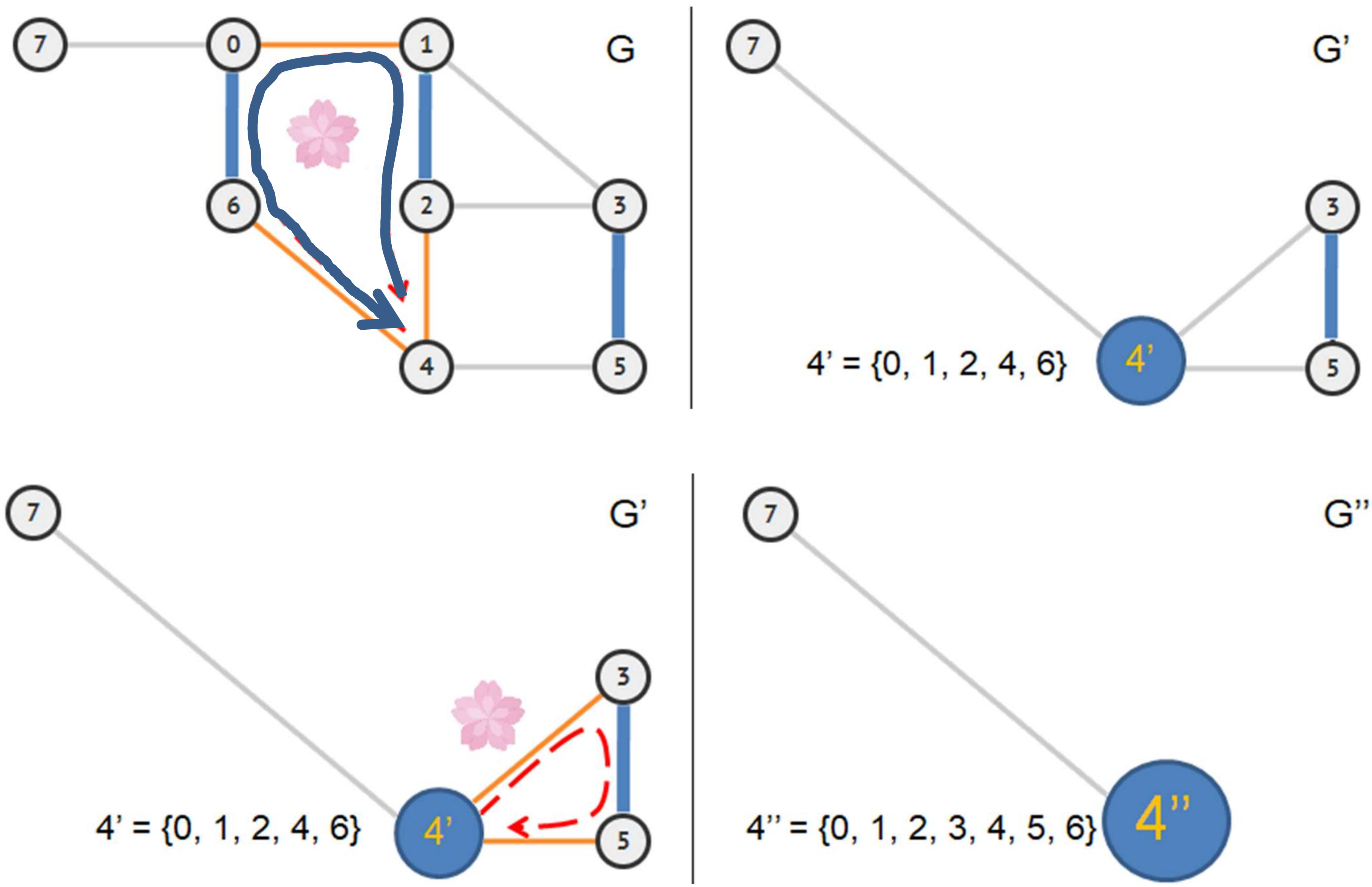
(Switch to still-draft visualization @ VisuAlgo for live explanation)



https://visualgo.net/en/matching, select "Unweighted General" tab

Blossom Shrinking/Expansion (2)

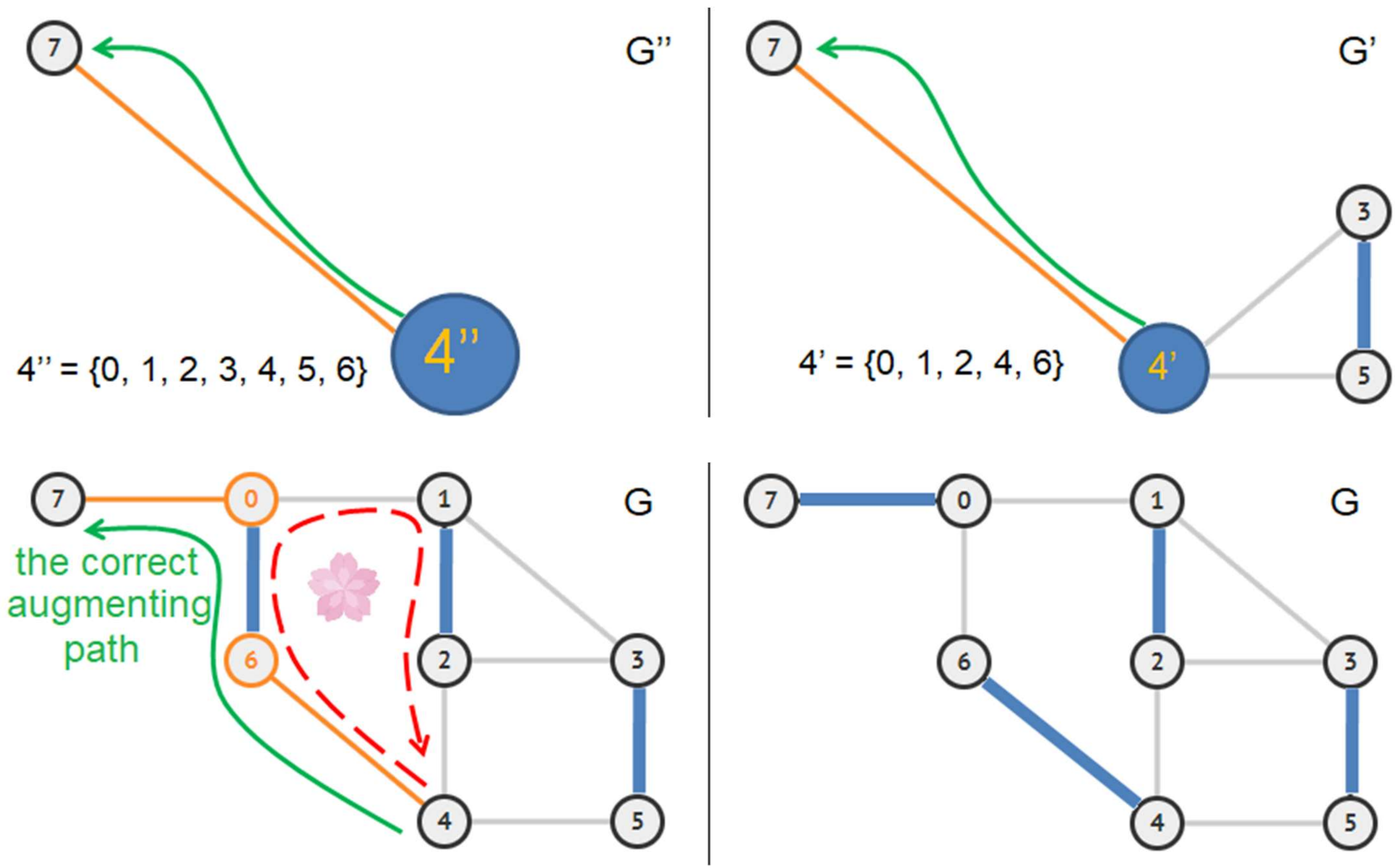
Review the recording or re-read CP4 Book 2 for the explanation



https://visualgo.net/en/matching, select “Unweighted General” tab

Blossom Shrinking/Expansion (3)

Review the recording or re-read CP4 Book 2 for the explanation



When To Use Edmonds' Matching?

This algorithm is a *hard* to implement…

O($\mathbf{V}^3$) library code is preferred

- Only used for unweighted MCM with $\mathbf{V} \in$ [22..200*; complex code]
  - If $\mathbf{V} \leq$ [19..21], see the next section
- Randomized greedy pre-processing is ALSO APPLICABLE here!!
  - Again, you can quote this verbatim for final assessment
  - Focus on the unweighted non-bipartite graph modeling

For code, just use external source

- https://codeforces.com/blog/entry/49402 (C++)
- https://sites.google.com/site/indy256/algo/edmonds_matching (Java)

Solution(s):

DP with Bitmask (only for small graph)

Or Modified? Edmonds' Matching (future work…)

- https://arxiv.org/abs/1703.03998 or at https://home.cs.colorado.edu/~hal/MCM.pdf

WEIGHTED MCM

No? Choice… (only for **V** ≤ [19..21])

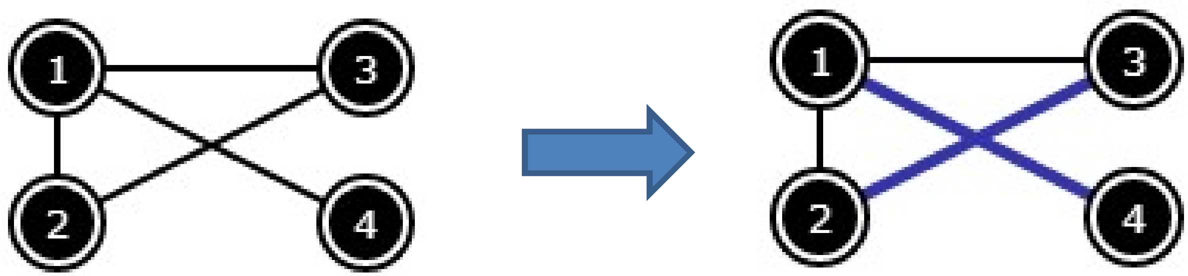
Old code, you can optimize a bit using Least Significant One technique

```
ii wMCM(int mask) { // returns (|MCM|, min-weight-of-the-MCM)
  if (mask == (1<<N)-1) return ii(0, 0); // no more matching
  if (memo[mask] != ii(-1, -1)) return memo[mask];
  int p1; // find the first free vertex
  for (p1 = 0; p1 < N; ++p1) if (!(mask & (1<<p1))) break;
  ii ans = wMCM(mask | (1<<p1)); // p1 unmatched
  for (int p2 = p1+1; p2 < N; ++p2) // find the second free vertex
    if (!(mask & (1<<p2)) && cost[p1][p2]) {
      ii nxt = wMCM(mask | (1<<p1) | (1<<p2)); // match p1-p2
      nxt.first += 2; nxt.second += cost[p1][p2]; // add MCM+weight
      if ((nxt.first > ans.first) || // better MCM
          ((nxt.first == ans.first) && // or equal MCM
           (nxt.second < ans.second))) // but with smaller weight
        ans = nxt;
    }
  return memo[mask] = ans;
}
```

This is slightly more general than the intro problem in Chapter 1 of CP book series (UVa 10911) – min weight **perfect** matching on a weighted complete graph https://visualgo.net/en/recursion, select the “Matching” example…

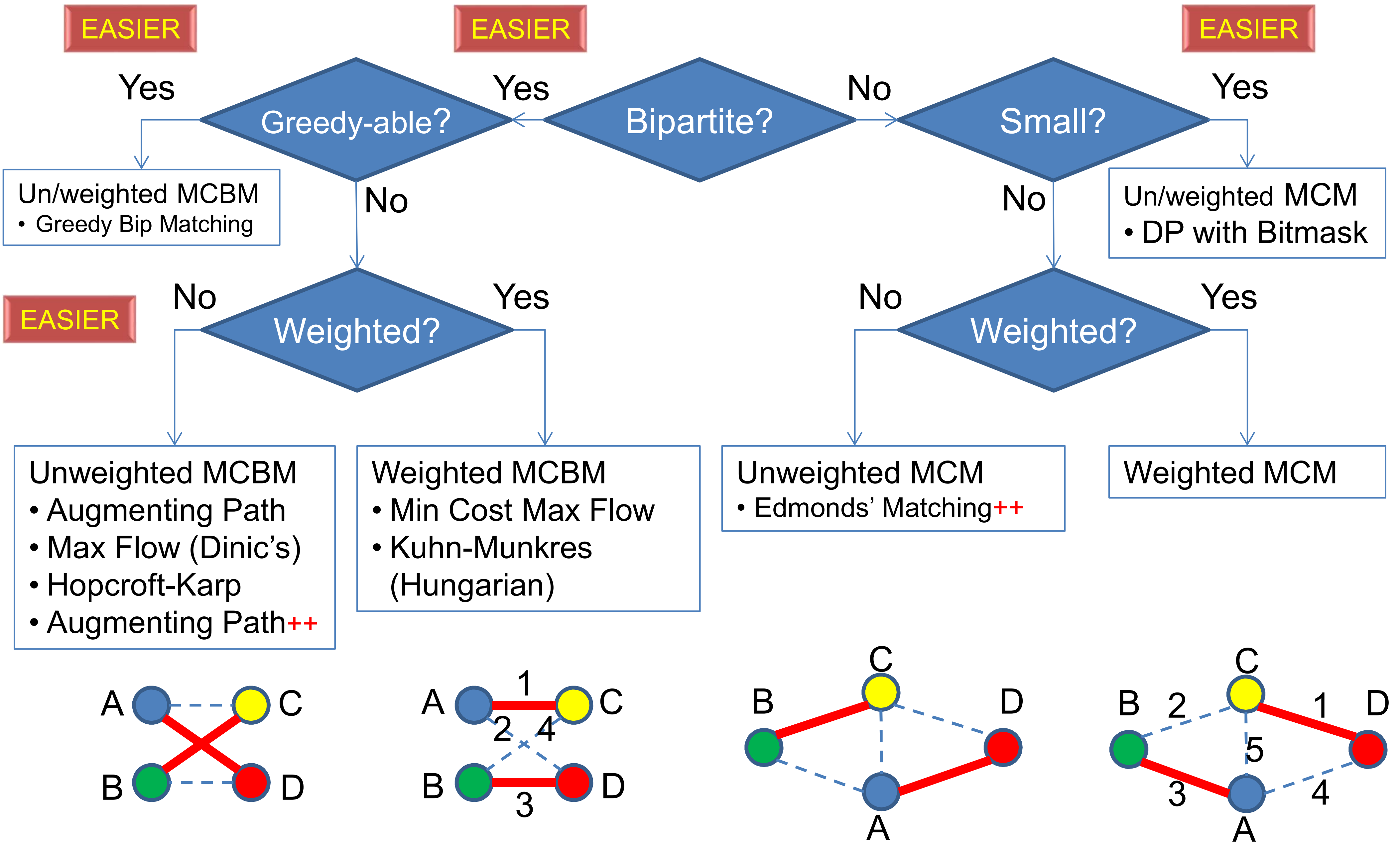
Also for Small Unweighted MCM Too

Just set all weights = 1 in the previous code
But most likely TLE for 22 < V < 200 – so just use Edmonds’ Matching algorithm for that



https://www.youtube.com/watch?v=GhjwOiJ4SqU&feature=youtu.be&t=2m54s cites CP3

Types of Graph Matching



References

- Mostly CP4, Book 1 Section 4.6, then Book 2 Section 8.5, and a few sections in Chapter 9 (9.25-9.29)
- TopCoder PrimePairs, RookAttack solution
- http://www.comp.nus.edu.sg/~cs6234/2009/Lectures/Match-sl-PC.pdf (Prof LeongHW's/P Karras slides)
- There are much bigger topics outside these two lectures and two tutorials and these two lecture notes will keep be improved over the years (twice per year, in S1/CS4234 and S2/CS3233)…

All the best for your Midterm Test

- The true class ranking will emerge after this
- At the moment too many are tied at 15/15 (and soon at 20/20)
  - I will grade PS4 early birds soon