CS4234
Optimiz(s)ation Algorithms

L8 – Max-(Cardinality/Weight) – (Bipartite) – 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
Competitive Programming

Dr. Steven Halim
Graph Matching

- Overview
- Unweighted MCBM: Max Flow review, Augmenting Path, Hopcroft Karp’s (nice theoretical result), Augmenting Path++
  - Relevant Applications: Bipartite Matching (with Capacity, use MF), Max Independent Set, Min Vertex Cover, Min Path Cover on DAG
  - Some will be discussed in tutorial 😊

- Weighted MCBM: (Min/Max) Cost (Max) Flow (overview)
- Unweighted MCM: Edmonds’s Matching (overview)
- Weighted MCM: DP with Bitmask (small graph only, review)
  - Sorry, still unable to make it work for Christofides’s 1.5-Approximation algorithm for M-R/NR-TSP
Outline for CS3233

- Graph Matching
  - Overview
  - Unweighted MCBM: Max Flow, Augmenting Path++, ("Dropped"): Hopcroft Karp’s
    - Relevant Applications: Bipartite Matching (with Capacity, use MF), Max Independent Set, Min Vertex Cover, Min Path Cover on DAG
  - Weighted MCBM: (Min/Max) Cost (Max) Flow (overview)
  - Unweighted MCM: Edmonds’s Matching (overview)
  - Weighted MCM: DP with Bitmask (small graph, review)
Graph Matching

A matching (marriage) in a graph G (real life) is a subset of edges in G (special relationships) such that no two of which meet at a common vertex (that is, no affair!)

Thus 1. and 2. are matchings (red thick edge),

But trying to match both edges in 3. is invalid since there is an overlapping vertex B
Max-Cardinality-Matching (MCM)

Usually, the problem asked in graph matching is the size (cardinality) of a maximum (not maximal) matching.

A maximum matching is a matching that contains the largest possible number of edges.

A possible variant: Perfect Matching

- MCM with no vertex left unmatched
  - PS: This is *impossible* on graph with odd number of vertices
Examples

A is a maximum matching (0 matching)
(no edge to be matched)

\[ \text{A} \rightarrow \text{B} \text{ is also a maximum matching (1 matching)} \]
(no other edge to be matched)

But \[ \text{A} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \text{D} \]
is not a maximum matching
(it is a maximal matching btw)

as we can change it to \[ \text{A} \rightarrow \text{B} \rightarrow \text{D} \text{ or } \text{A} \rightarrow \text{C} \rightarrow \text{D} \]
(2 matchings)
Types of Graph Matching

- Bipartite?
  - Yes: Weighted?
    - Yes: Weighted MCM
      - Unweighted MCM
        - Unweighted MCBM
          - Max Flow (2nd best)
          - Augmenting Path++
          - Hopcroft Karp's
      - Weighted MCM
        - Edmonds’s Matching
        - DP with Bitmask (small graph)
        - Edmonds’s (future w)
    - No: Unweighted MCM
      - Unweighted MCBM
      - Max Flow (2nd best)
        - Augmenting Path++
        - Hopcroft Karp’s
  - No: Weighted?
    - Yes: Weighted MCM
      - Weighted MCBM
        - Min Cost Max Flow
        - Kuhn Munkres’s (Hungarian, future w)
      - Weighted MCM
        - DP with Bitmask (small graph)
        - Edmonds’s (future w)
    - No: EASIER

CS3233/CS4234
Steven Halim, SoC, NUS
Types of Graph Matching

- **Bipartite?**
  - Yes
    - Weighted?
      - Yes
        - Weighted MCM
          - Min Cost Max Flow
          - Edmonds’s Matching
          - DP with Bitmask (small graph)
          - Edmonds’s (future w)
      - No
        - Unweighted MCM
          - Edmonds’s Matching
          - DP with Bitmask (small graph)
          - Edmonds’s (future w)
  - No
    - Weighted?
      - Yes
        - Weighted MCBM
          - Min Cost Max Flow
          - Kuhn Munkres’s (Hungarian, future w)
      - No
        - Unweighted MCBM
          - Max Flow (2nd best)
          - Augmenting Path++
          - Hopcroft Karp’s
Solutions:
Max Flow (2nd best solution for this variant)
Augmenting Path Algorithm++ (the ++ is very important :O)
Hopcroft Karp’s Algorithm (for theoretical interest only) ~= Dinic’s

UNWEIGHTED MCBM
Max-Cardinality-Bipartite-Matching (MCBM)

A Bipartite graph is a graph whose vertices can be divided into two disjoint sets $X$ and $Y$ such that every edge can only connect a vertex in $X$ to one in $Y$.

Matching in this kind of graph is a lot easier than matching in general graphs.
MAX FLOW
CP3 SECTION 4.6 & 4.7.4
MCBM

Examples:
- Marriage (Man to Woman), Job to Machine, etc.

Set 2
- No edges within set 2

Set 1
- No edges within set 1

Edges can only go across the 2 sets!

Note: these slides are modified from an unknown source and has horizontal layout.
A Bipartite Matching

Any possible pairing from set 1 to set 2

Man

Woman

Job, etc

Machine, etc
Max Cardinality Bipartite Matching

Maximum possible matching

But the solution is not unique!
Max Cardinality Bipartite Matching

Maximum possible matching

But the solution is not unique!
Flow Graph Modeling for MCBM

NOTICE THE ARROWS!

All edge weight = 1
(can be > 1)

All edge weight = 1
(or set to INF)

All edge weight = 1
(can be > 1)
Max Flow Solution for MCBM

Time Complexity: Depends on the chosen Max Flow algorithm

http://visualgo.net/maxflow, select modeling, Bipartite Matching, all ones
Ex 4.7.4.1*:
Why the graph has to be directed?

• Try this counter example (there are many others) – $|MCBM| = 2$ (e.g. $0 \rightarrow 3$ and $1 \rightarrow 4$) – $|MF|$ from source to sink $= 3$ (all edges are undirected and have capacity 1) (e.g. $s \rightarrow 0 \rightarrow 3 \rightarrow t$, $s \rightarrow 1 \rightarrow 4 \rightarrow t$, AND $s \rightarrow 2 \rightarrow 4 \rightarrow 0 \rightarrow 5 \rightarrow t$) we don’t have back edge $4 \rightarrow 0… \rightarrow \infty$
When To Use Max Flow Solution?

Only if we are solving Bipartite Matching with Capacity (the “Assignment Problem”)

- e.g. UVa 259, page 166-167
- This variant will be much slower to solve if reduced to MCBM, imagine if the typical capacities of the edges are big, reducing the problem to MCBM will lead to a gigantic bipartite graph...

But if we are solving pure MCBM (capacity is all 1), just use the next algorithm (easier to code)
Finding MCBM via

AUGMENTING PATH ALGORITHM
CP3 SECTION 4.7.4++
Augmenting Path

In this graph, the path colored orange (unmatched) - red (matched) - orange: A-B-C-D is an augmenting path.

We can flip the edge status to red-orange-red: A-B-C-D and the number of edges in the matching set increases by 1.
Augmenting Path Algorithm

Lemma (Claude Berge 1957):

A matching M in G is maximum iff there is no more augmenting path in G

Augmenting Path Algorithm is a simple $O(V^2(V+E)) = O(V^2 + VE) \approx O(VE)$ implementation of that lemma

- Find and then eliminate augmenting paths in G
The Code (1) 😊

vi match, vis; // global variables

int Aug(int L) { // return 1 if ∃ an augmenting path from L
    if (vis[L]) return 0; // return 0 otherwise
    vis[L] = 1;
    for (int j = 0; j < (int)AdjList[L].size(); j++) {
        int R = AdjList[L][j].first;
        if (match[R] == -1 || Aug(match[R])) {
            match[R] = L;
            return 1; // found 1 matching
        }
    }
    return 0; // no matching
}
The Code (2) 😊

// in int main(), build the bipartite graph
// only directed edge from left set to right set is needed

int MCBM = 0;
match.assign(V, -1);

for (int L = 0; L < Vleft; L++) {
    vis.assign(Vleft, 0);
    MCBM += Aug(L); // find augmenting path starting from L
}

printf("Found %d matchings\n", MCBM);
Augmenting Path Algorithm

Easy Assignment
1 matching (dotted line)

Augmenting Path
2-3-1-4

After Flip
2 matchings (dotted lines)

An augmenting path
F=Free, M=Matched

Flip to increase matching from 1 to 2 matchings

http://visualgo.net/matching, try it there 😊
Finding MCBM via

HOPCROFT KARP’S ALGORITHM
(SECTION 9.12)
An Extreme Test Case...

A Complete Bipartite Graph $K_{N,M}$, $V = N+M$ & $E = N*M$

Augmenting Path algorithm $\rightarrow O((N+M)*(N*M))$

• If $M = N$, we have an $O(N^3)$ solution, only OK for $N \leq 200$

Example with $N = M = 5$

http://visualgo.net/matching, load (almost) complete Bipartite Graph, and see how slow the standard Augmenting Path algorithm can be...
Hopcroft Karp’s Algorithm (1973)

Key Idea (very similar to Dinic’s algorithm):

- Use shortest augmenting paths from all free vertices (BFS)
- Run similar algorithm as the Augmenting Path Algorithm earlier (DFS), but now using this BFS information,
- Can have ≥1 matching(s) per iteration
Hopcroft Karp’s Algorithm (1973)

Hopcroft Karp’s runs in $O(E\sqrt{V})$, proof omitted

- For the extreme test case in previous slide, this is $O(N^*M^*\sqrt{N+M})$
- With $M = N$, this is about $O(N^{5/2})$, OK for $N \leq 600$

Ex 4.7.4.3*: Is this algorithm must be learned in order to do well in practice/programming contest?

- CP Answer: NOT needed...
  - You can make the standard augmenting path algorithm avoids its worst case behavior by doing randomized greedy pre-processing (randomized to avoid adversary test case 😊), try this in visualgo: “Augmenting Path with Randomized Greedy Preprocessing”
Types of Graph Matching

- **Bipartite?**
  - **Yes**
    - Weighted?
      - **Yes**
        - Weighted MCM
          - Min Cost Max Flow
          - Kuhn Munkres’s (Hungarian, future w)
      - **No**
        - Unweighted MCM
          - Edmonds’s Matching
          - DP with Bitmask (small graph)
  - **No**
    - Weighted?
      - **Yes**
        - Weighted MCBM
          - Max Flow (2nd best)
          - Augmenting Path++
          - Hopcroft Karp’s
      - **No**
        - Unweighted MCBM
          - Max Flow (2nd best)
          - Augmenting Path++
          - Hopcroft Karp’s

CS3233/CS4234
Steven Halim, SoC, NUS
Solution:
Min (Max?) Cost (Max) Flow (Overview Only)
Kuhn Munkres’s (Hungarian, not discussed, haven’t learn 😞)

WEIGHTED MCBM
UVa 10746 (1)

Find Max Flow (=3 here) of Minimum Cost (still in P)

Complete Bipartite Graph $K_{n,m}$
Capacity = 1
Cost (as shown in edge labels)
Complete Bipartite Graph $K_{n,m}$
Capacity = 1
Cost (as shown in edge labels)

Min Cost so far = $0 + 5.0 + 0 = 5.0$
Complete Bipartite Graph $K_{n,m}$
Capacity = 1
Cost (as shown in edge labels)
UVa 10746 (Solution)

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*mn)-TBC if using Bellman Ford’s algorithm to find the cheapest augmenting path…
Types of Graph Matching

Bipartite?

- Yes
  - Weighted?
    - Yes
      - Unweighted MCM
        - Edmonds’s Matching
        - DP with Bitmask (small graph)
        - Edmonds’s (future w)
    - No
      - Weighted MCM
        - DP with Bitmask (small graph)
        - Kuhn Munkres’s (Hungarian, future w)
  - No
    - Weighted?
      - Yes
        - Weighted MCM
          - Min Cost Max Flow
          - Kuhn Munkres’s (Hungarian, future w)
      - No
        - Unweighted MCM
          - Edmonds’s Matching
          - DP with Bitmask (small graph)
          - Hopcroft Karp’s

- No
  - Weighted?
    - Yes
      - Max Flow (2nd best)
      - Augmenting Path++
      - Hopcroft Karp’s
    - No
      - Unweighted MCBM
Solution:
Edmonds’s Matching Algorithm (Overview only)
DP with Bitmask (Small Graph, discussed later)

UNWEIGHTED MCM
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 in such graph

- Alternating Cycle: 3-1-2-3 is called a blossom
Blossom Shrinking/Expansion

Shrinking these blossoms (recursively) will make this problem “easy” again (PS: I will show some live animation locally)
When To Use Edmonds’s Matching?

This algorithm is a bit hard to implement...

\(O(V^4)\) or faster* library code is preferred

- Only used for unweighted MCM with \(V > 21\) and \(V \leq 50\)
Types of Graph Matching

- **Bipartite?**
  - Yes
  - Weighted?
    - Yes
      - Weighted MCBM
        - Max Flow (2\textsuperscript{nd} best)
        - Augmenting Path++
        - Hopcroft Karp’s
      - Weighted MCM
        - DP with Bitmask (small graph)
        - Edmonds’s (future w)
    - No
      - Unweighted MCBM
        - Max Flow (2\textsuperscript{nd} best)
        - Augmenting Path++
        - Hopcroft Karp’s
      - Unweighted MCM
        - Edmonds’s Matching
        - DP with Bitmask (small graph)
  - No
    - Weighted?
      - Yes
        - Weighted MCM
          - DP with Bitmask (small graph)
          - Edmonds’s (future w)
      - No
        - Unweighted MCM
          - Edmonds’s Matching
          - DP with Bitmask (small graph)

CS3233/CS4234
Steven Halim, SoC, NUS
Solution(s):
DP with Bitmask (only for small graph)
Edmonds’ Matching (future work...)

WEIGHTED MCM
No? Choice... (only for $V \leq [19..21]$)

```cpp
ii wMCM(int mask) {
    if (mask == (1<<N)-1) return ii(0, 0);
    if (memo[mask] != ii(-1, -1)) return memo[mask];
    int p1, p2;
    for (p1 = 0; p1 < N; p1++) if (!(mask & (1<<p1))) break;
    ii ans = wMCM(mask | (1<<p1)); // p1 unmatched
    for (p2 = p1+1; p2 < N; p2++)
        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];
            if ((nxt.first > ans.first) || // equal # matching
                ((nxt.first == ans.first) &&
                 (nxt.second < ans.second))) // or smaller cost
                ans = nxt;
        }
    return memo[mask] = ans;
}
```
http://visualgo.net/recursion, select the “Matching” example…

Also for Small Unweighted MCM Too

Just set all weights = 1 in the previous code
Summary

- Bipartite?
  - Yes
    - Weighted?
      - Yes
        - Weighted MCM
          - DP with Bitmask (small graph)
          - Edmonds’s (future w)
      - No
        - Unweighted MCM
          - Edmonds’s Matching
          - DP with Bitmask (small graph)
  - No
    - Weighted?
      - Yes
        - Weighted MCBM
          - Min Cost Max Flow
          - Kuhn Munkres’s (Hungarian, future w)
      - No
        - Unweighted MCBM
          - Max Flow (2nd best)
          - Augmenting Path++
          - Hopcroft Karp’s
References

• Mostly CP3, Section 4.7.4, 9.10, 9.12, and 9.24 😊
• TopCoder PrimePairs, RookAttack solution