2024 Columbia Training Camp

Day 9: MCMF

Christian Yongwhan Lim

Thursday, August 29, 2024

Overview

- Flow: Dinic + MCMF
- Practice Problems

Definitions

- A **residual network** G^R of network G is a network which contains two edges for each edge $(v, u) \in G$:
  - (v, u) with capacity $c^R_{vu} = c_{vu} - f_{vu}$
  - (u, v) with capacity $c^R_{uv} = f_{vu}$

Definitions

- A **blocking flow** of some network is such a flow that every path from s to t contains at least one edge which is saturated by this flow.
  - Note that a blocking flow is not necessarily maximal.

- A **layered network** of a network G is a network built in the following way:
  - For each vertex v we calculate level[v]: the shortest path (unweighted) from s to this vertex using only edges with positive capacity.
  - We keep only those edges (v, u) for which level[v]+1 = level[u].
  - Obviously, this network is acyclic.

Dinic's Algorithm

- The algorithm consists of several phases.
- On each phase:
  - Construct the layered network of the residual network of G.
  - Find an arbitrary blocking flow in the layered network and add it to the current flow.

Number of Phases

- The algorithm terminates in less than V phases.

Finding Blocking Flow

- In order to find the blocking flow on each iteration, we may simply try pushing flow with DFS (Depth-First Search) from s to t in the layered network while it can be pushed.
- In order to do it more efficiently, we must remove the edges which cannot be used to push anymore.
- We can keep a **pointer** in each vertex which points to the next edge which can be used.

Finding Blocking Flow (con't)

- A single DFS run takes O(k + V) time, where k is the number of pointer advances on this run.
- Over all runs, a number of pointer advances cannot exceed E.
- A total number of runs would not exceed E, as every run saturates at least one edge.
- So, a total running time of finding a blocking flow is **O(VE)**.

Dinic's Complexity

- Since there are less than V phases, the total time complexity is **$O(V^2E)$**.

Minimum Cost Maximum Flow (MCMF)

- Given a network G consisting of n vertices and m edges.

- For each edge, the capacity (a non-negative integer) and the cost per unit of flow along this edge (some integer) are given.

- Also the source s and the sink t are marked.

Minimum Cost Maximum Flow (MCMF)

- For a given value K, we have to find a flow of this quantity, and among all flows of this quantity we have to choose the flow with the lowest cost.
- This task is called **minimum-cost flow** problem.

- Sometimes the task is given a little differently: you want to find the **maximum flow**, and among all maximal flows we want to find the one with the **least cost**. This is called the **minimum-cost maximum-flow** problem.

Minimum Cost Maximum Flow (MCMF)

- Both these problems can be solved effectively with the algorithm of successive shortest paths.

- At each iteration of the algorithm, we find the shortest path in the residual graph from s to t.

Minimum Cost Maximum Flow (MCMF)

- In contrast to Edmonds-Karp, we look for the shortest path in terms of the **cost of the path** instead of the number of edges.
  - If there doesn't exist a path anymore, then the algorithm terminates.
  - If a path was found, we increase the flow along it as much as possible.
  - If at some point the flow reaches the value K, then we stop the algorithm.
  - It is not difficult to see, that if we set K to infinity, then the algorithm will find the minimum-cost maximum-flow.

Implementations

- **Dinic**: https://cp-algorithms.com/graph/dinic.html#implementation

- **MCMF**: https://github.com/yongwhan/library/blob/master/tourist_mincostmaxflow.cpp

Practice #1: 1913E: Matrix Problem

Practice #2: 976F: Minimal k-covering

Practice #3: 1430G: Yet Another DAG Problem

Practice #4: 1682F: MCMF?

Practice #5: 1684G: Euclid Guess

Practice #6: 724E: Goods transportation

Practice #7: 1615H: Reindeer Games

Practice #8: 925F: Parametric Circulation


THANK YOU