Flow

Ian

A Few things you need to know

---

1. Flow problem is like < 0.5% to appear in ioi contest(0% for full solution).
2. The most important component in flow problem is your model, without a effective model u better just go solve the subtask or work on other.
3. NOI flow problems is hard to model
4. There are many materials about flow in the Internet. If you want to have deeper understanding, google it(or ask me).

Maximum-Flow problem

Image you have a infinite water source S and sink T. There is some pump connecting them and each pump has capacity, find how many water could flow from S to T.

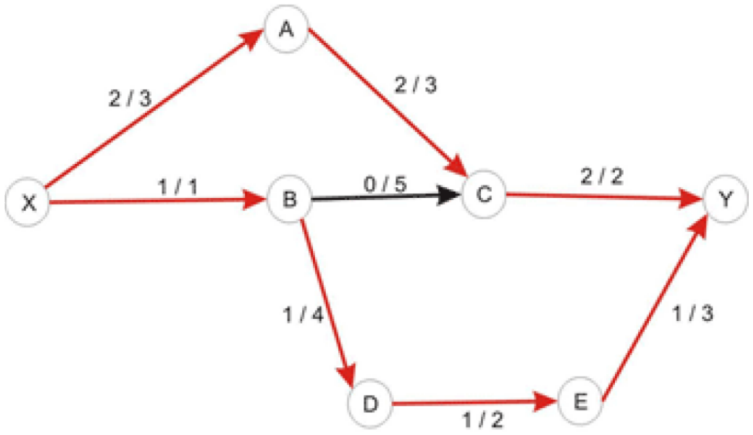


Figure 1a - Maximum Flow in a network

(picture from Topcoder)

Residual Network

For any edge u->v in the graph, there is a forward edge u->v with capacity equal to original capacity - flow, and a backward edge v->u with capacity equal to flow.

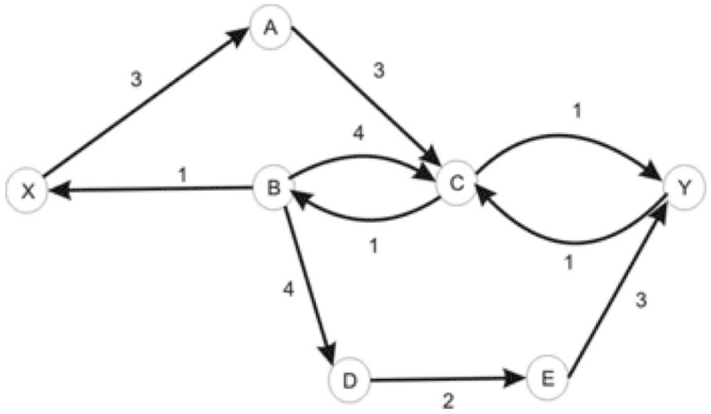


Figure 2b - The residual network of the network in 2a

(picture from Topcoder)

Augmenting Path

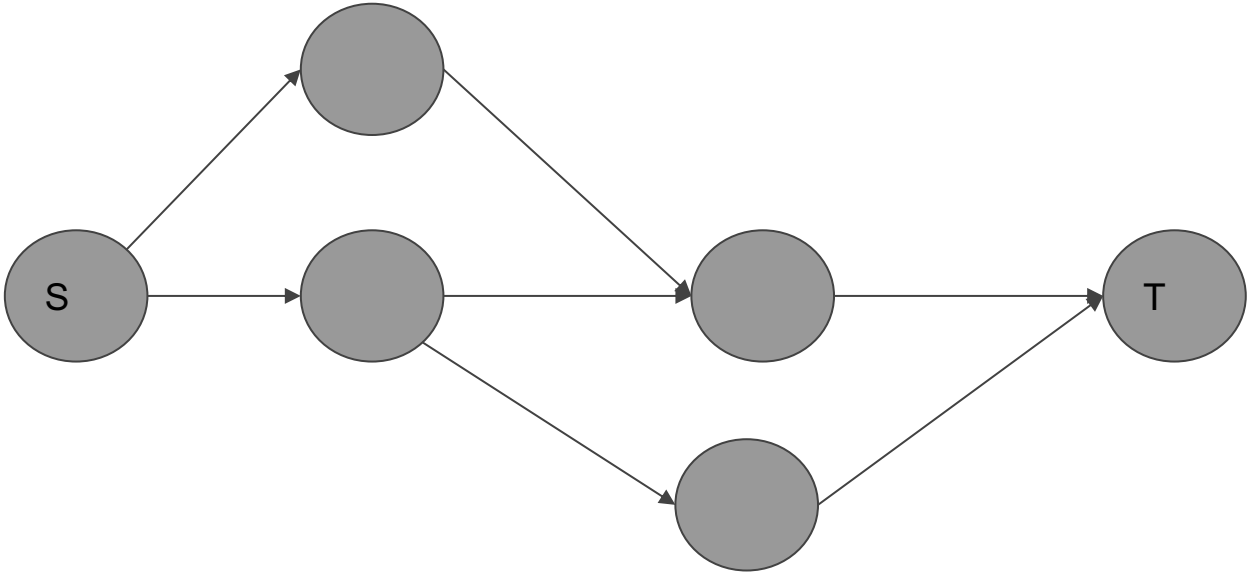
An augmenting path is a path from source to sink in the residual network and could increase the flow.

Usage of Residual Network

Sometimes the path we found in the original network might not be optimal. Residual network could help us cancel the path and find a better path.

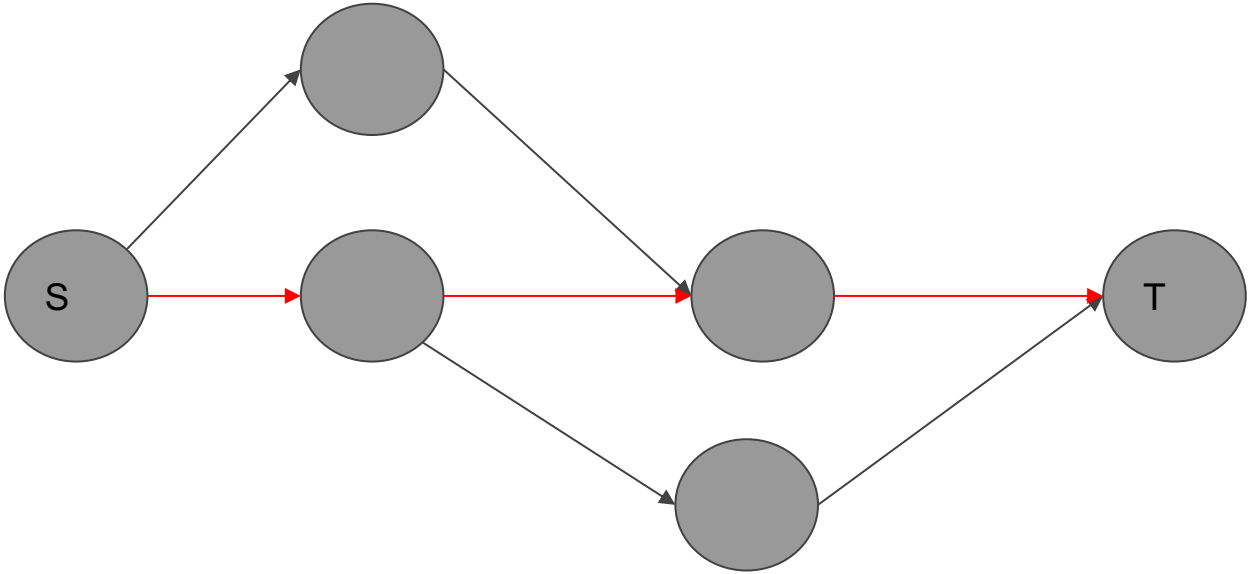
Usage of Residual Network

Assume all edge has capacity 1, red edge mean there is flow.



Usage of Residual Network

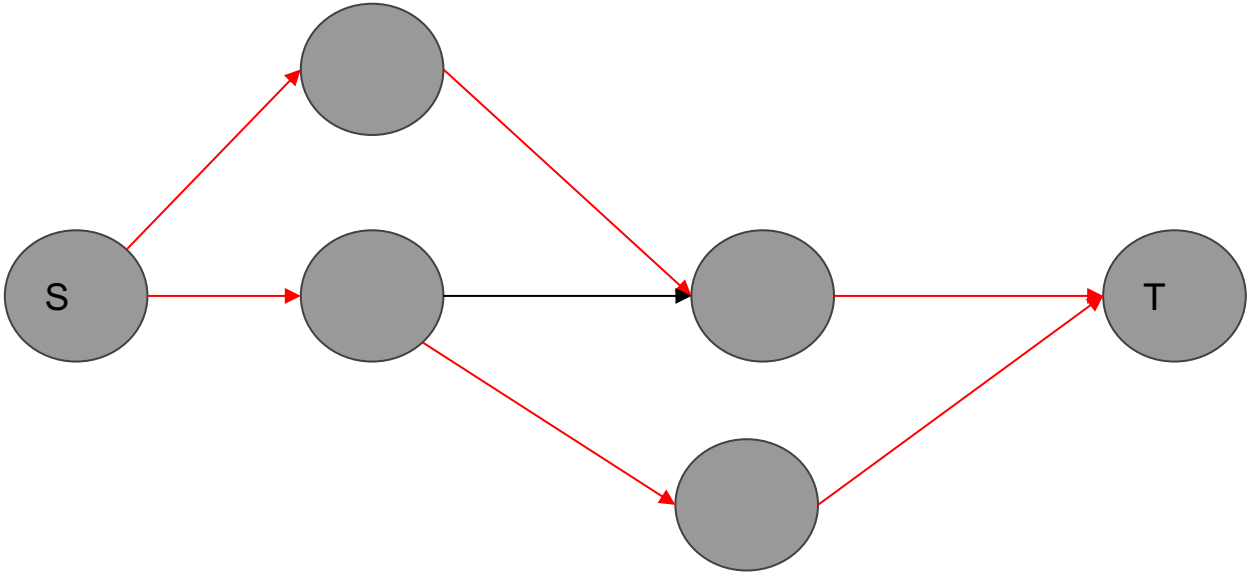
Assume all edge has capacity 1, red edge mean there is flow.



There is actually a better answer exist

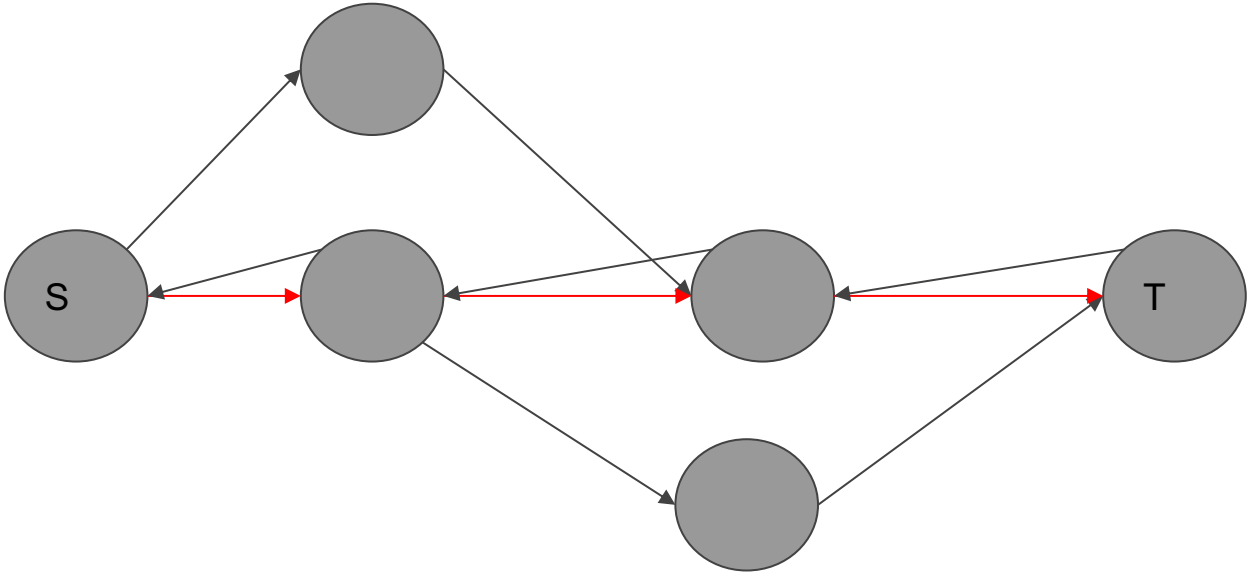
Usage of Residual Network

Assume all edge has capacity 1, red edge mean there is flow.



Usage of Residual Network

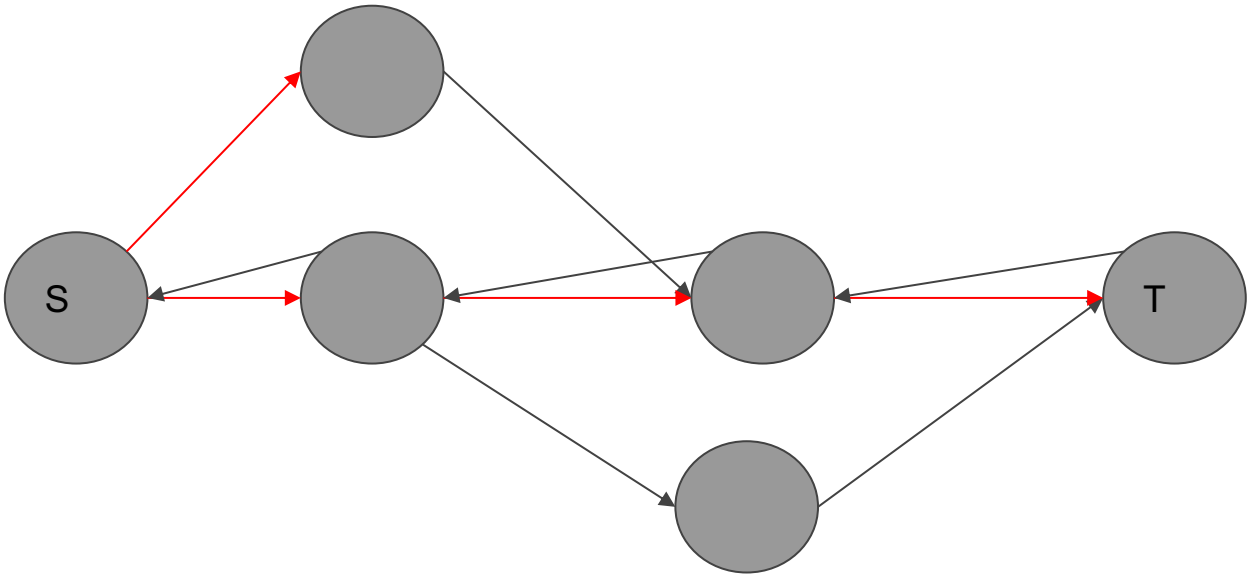
Assume all edge has capacity 1, red edge mean there is flow.



With residual network

Usage of Residual Network

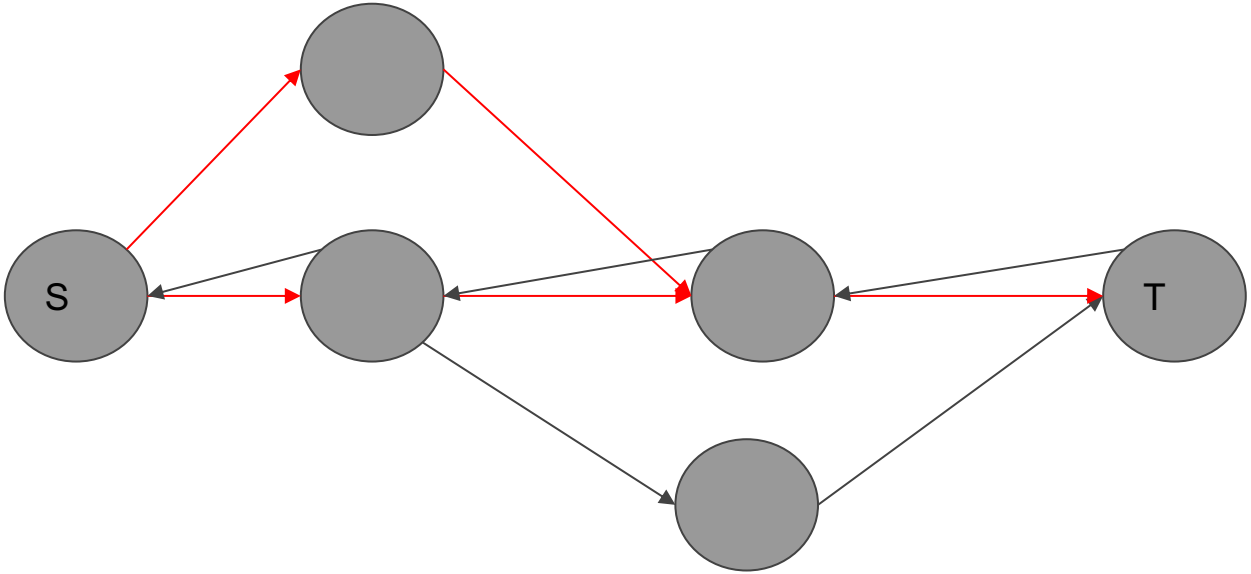
Assume all edge has capacity 1, red edge mean there is flow.



With residual network

Usage of Residual Network

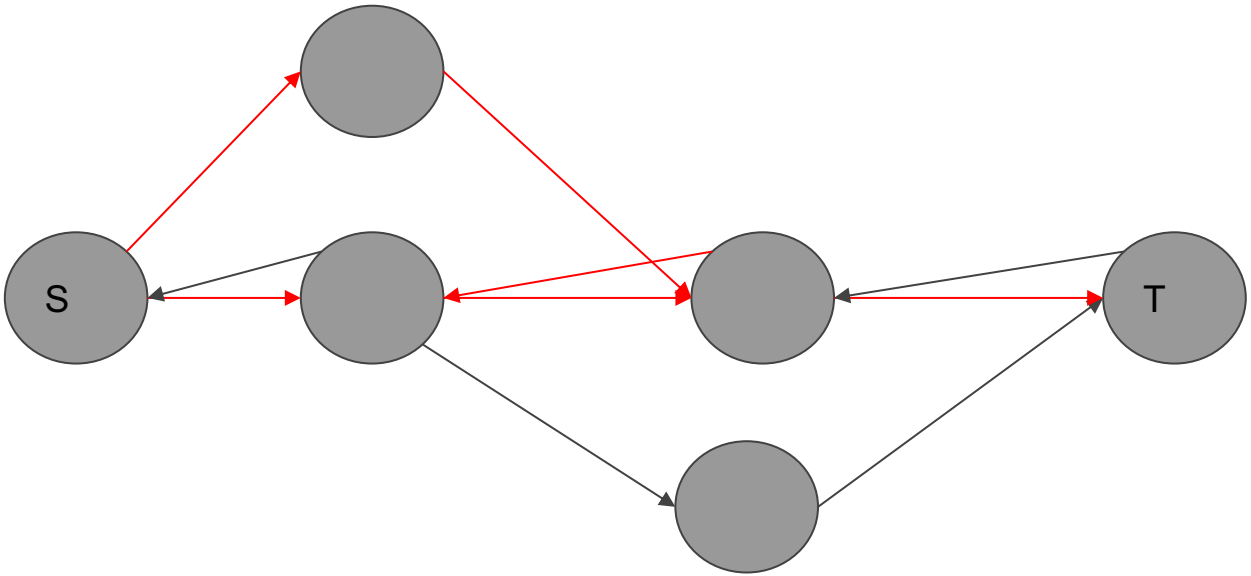
Assume all edge has capacity 1, red edge mean there is flow.



With residual network

Usage of Residual Network

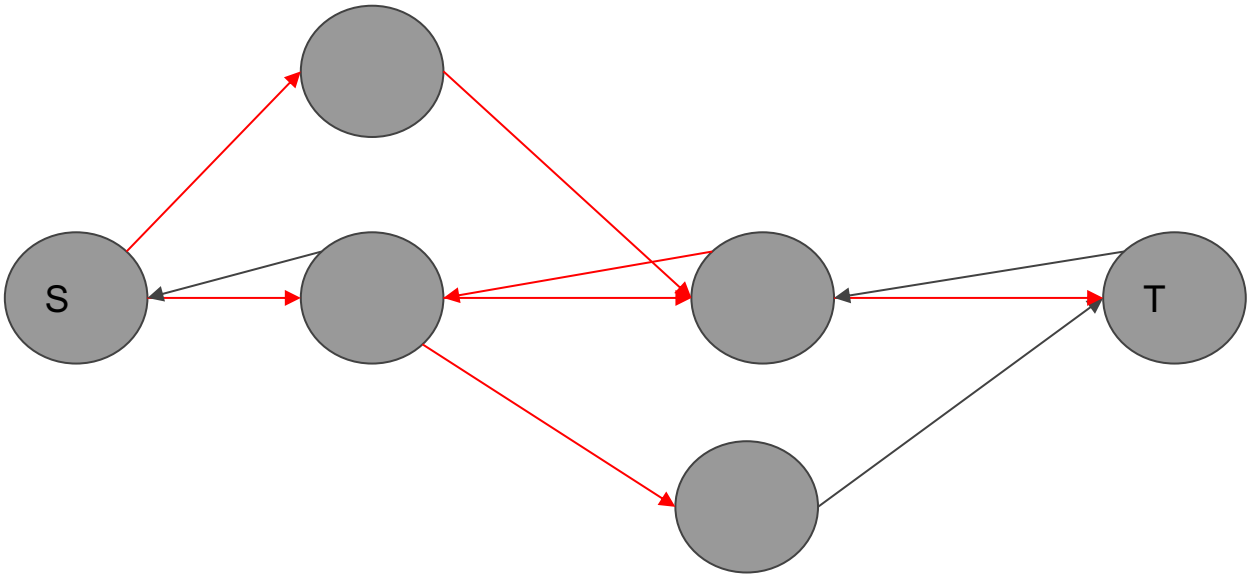
Assume all edge has capacity 1, red edge mean there is flow.



With residual network

Usage of Residual Network

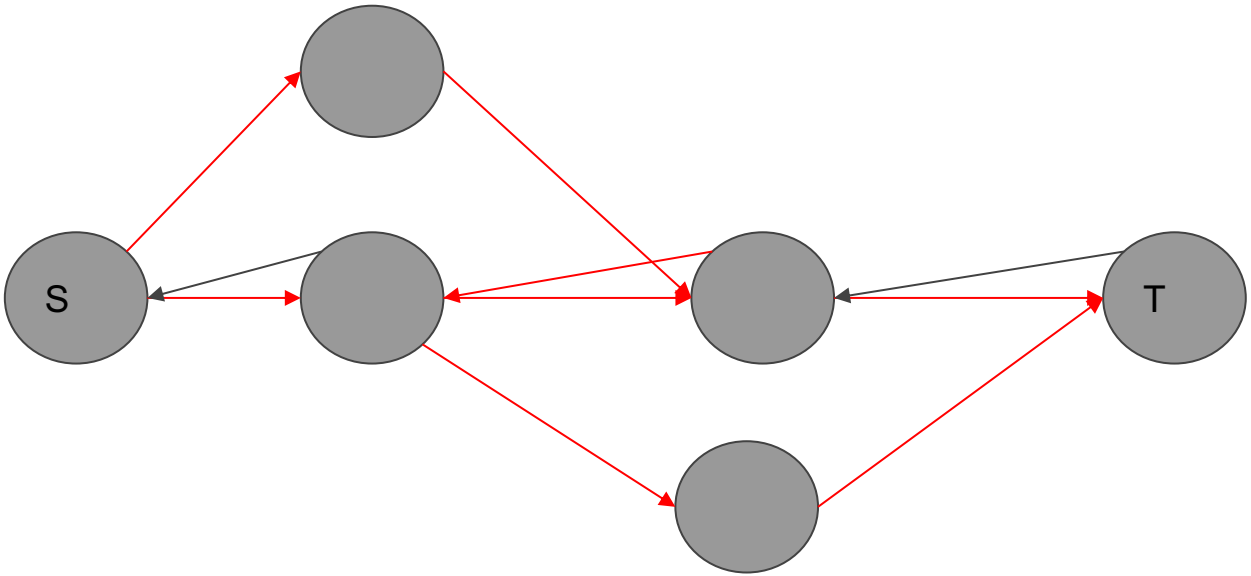
Assume all edge has capacity 1, red edge mean there is flow.



With residual network

Usage of Residual Network

Assume all edge has capacity 1, red edge mean there is flow.



With residual network

Solving Maximum-Flow Problem

Using residual network and augmenting path could help us solve the problem. We keep finding augmenting path in the residual graph until there is none, each time adding the path capacity to the answer. The final answer would be the maximum flow.

This is the Ford-Fulkerson algorithm which works in O(EF), where E is the no. of edges and F is the maximum flow.

Dinic's Alogrithm

Dinic algorithm is a better max-flow algorithm.

It has polynomial time complexity, O(V^2E).

The main concept is to divide the network into multiple layer. Each time the augmenting path would only travel from previous layer to the current layer.

Dinic's Algorithm

---

1. Do a bfs starting at the source. Only edge with remaining capacity > 0 would be used. If we could not reach to sink, terminate and return the answer.
2. We keep finding augmenting path in the residual network. We would only travel edge (v, u) which remaining capacity > 0 and dist(v) < dist(u).
3. Add the capacity of all augmenting path to answer and repeat step 1.

Special Case with Dinic’s Algorithm

In bipartite matching problem, it works in O(sqrt(V) * E).

Max-Flow Min-Cut Theorem

A s-t-cut is a partition of the vertices of a flow network into two sets, such that a set includes the source s and the other one includes the sink t.

The capacity of a s-t-cut is defined as the sum of capacities of the edges from the source side to the sink side.

Max-Flow Min-Cut Theorem

Obviously, we could not send more flow than the minimum-cut. So max-flow is bounded by min-cut.

Furthermore, by max-flow min-cut theorem, max-flow is actually equals to min-cut.

Classic Model in Max-Flow problem

- N0921 植物大戰殭屍
- N0621 最大獲利

N0921 植物大戰殭屍

Add source S and sink T.

Slightly rephase the problem, we have N plants, after killing plant i we might gain f(i) and f(i) might be negative.

If we have to kill plant j before i, add edge (i, j) with capacity INF.

Obviously, if there is a cycle in the graph, all vertex that could reach the cycle is not going to be killed. So we remove them first.

N0921 植物大戰殭屍

If f(i) is positive, add edge (s, i) with capacity f(i).

If f(i) is negative, add edge (i, t) with capacity -f(i).

Sum of positive f(i) - Max-flow would be the answer!

N0921 植物大戰殭屍

Why ???

We first assume the best situation, gaining all the positive f(i). But most of the time things won’t work in this way. We have to try not gain f(i) or pay some cost.

If edge (s, i) is in the min-cut, it means we better try not gain f(i).

If edge (i, t) is in the min-cut, it means paying f(i) is actually better.

N0921 植物大戰殭屍

Other edges got INF capacity and would not be in the min-cut. They also states the relying relationship.

N0621 最大獲利

It is actually very similar to the previous problem. Now try to think of it!

Min-Cost Flow

Now we got a new problem, what if the edges actually have cost. E.g. if we sent 1 capacity from the i-th edge, we need to pay cost(i). 2 capacity -> 2 * cost(i) etc.

We want to know what is the minimum cost to send f flows from s to t.

Min-Cost Flow

It is actually very similar to solving max-flow problem.

Instead of randomly choosing the augmenting path. We would like to choose the one that pay the least cost.

So we could apply shortest path algorithm. In practice, SPFA(shortest path faster algorithm) would be a nice choice(not 100%) as it could deal with negative cost without modifying.

M1641 Insepctor

Let’s try to solve this problem.

M1641 Insepctor

Main observation: optimal solution would be form some simple cycles.

Let’s slightly transform the problem. Find to(i) for every vertex where there would be an edge(i, to(i)) in the optimal solution and to(i) is distinct(as optimal solution must be simple cycles).

Clearly we want to minimize cost(i, to(i)) so just simple min-cost flow!

L-R Flow

If we still have much time we would discuss this topic.

If no time then you could visit https://cp-algorithms.com/graph/flow_with_demands.html

It is a extra materials and it possibly would affect 0% of your performance in IOI(don’t know about NOI).