#7주차 고급반

FLOW ALOGRITHM 1

T. 강건

수업의 목표

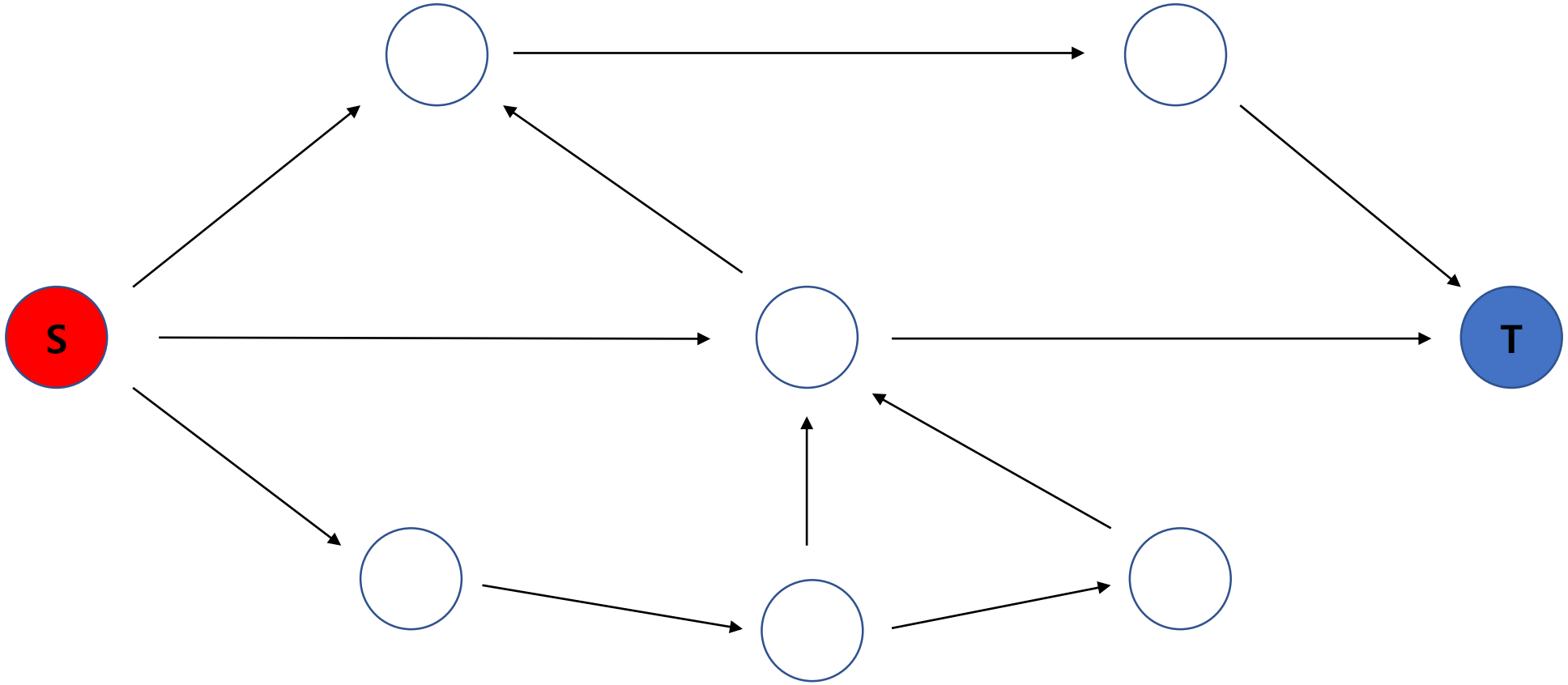
1. 강의 자료를 본 모두가 예제에서 “맞았습니다!!” 받기

Maximum Flow Problem

최대 유량 문제

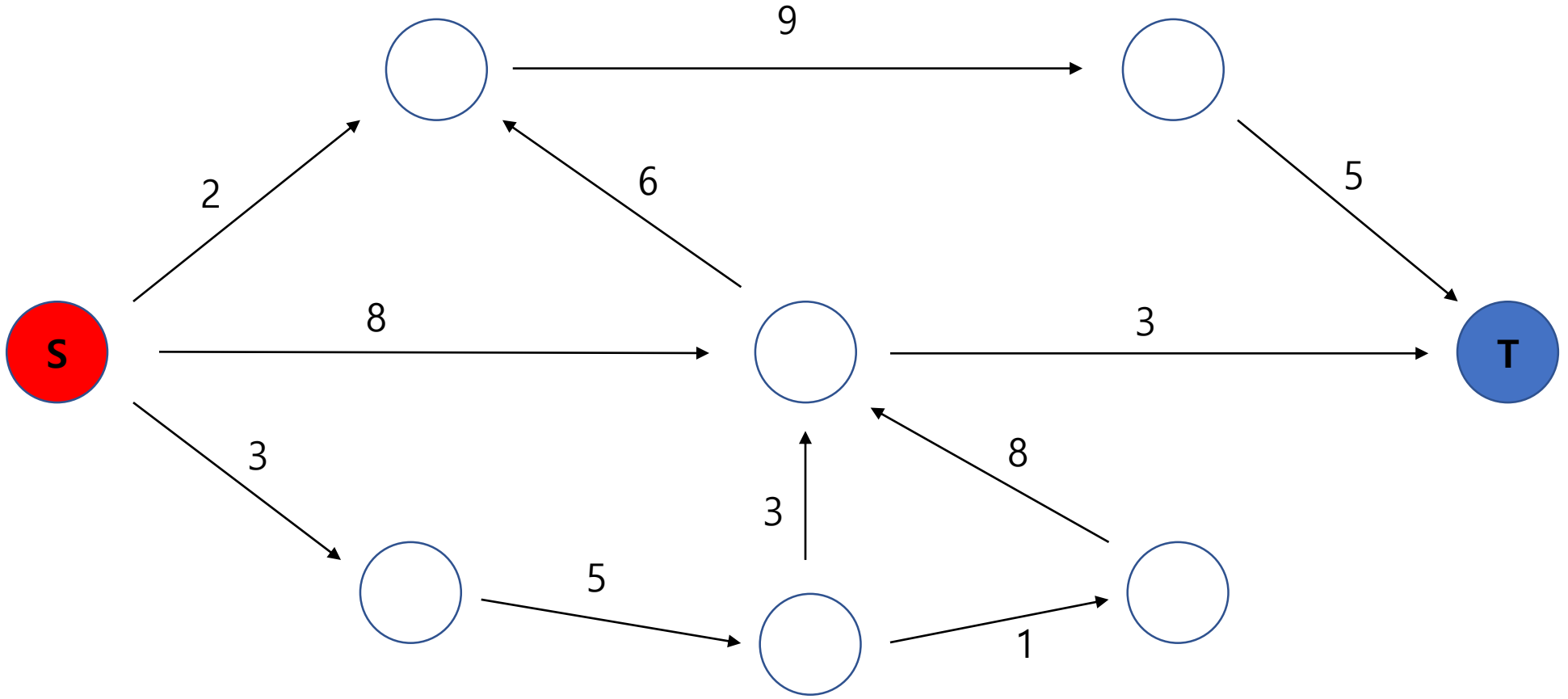
최대 유량 문제

S에서 T까지 파이프를 통해 물을 흘려 보낸다고 생각해보자



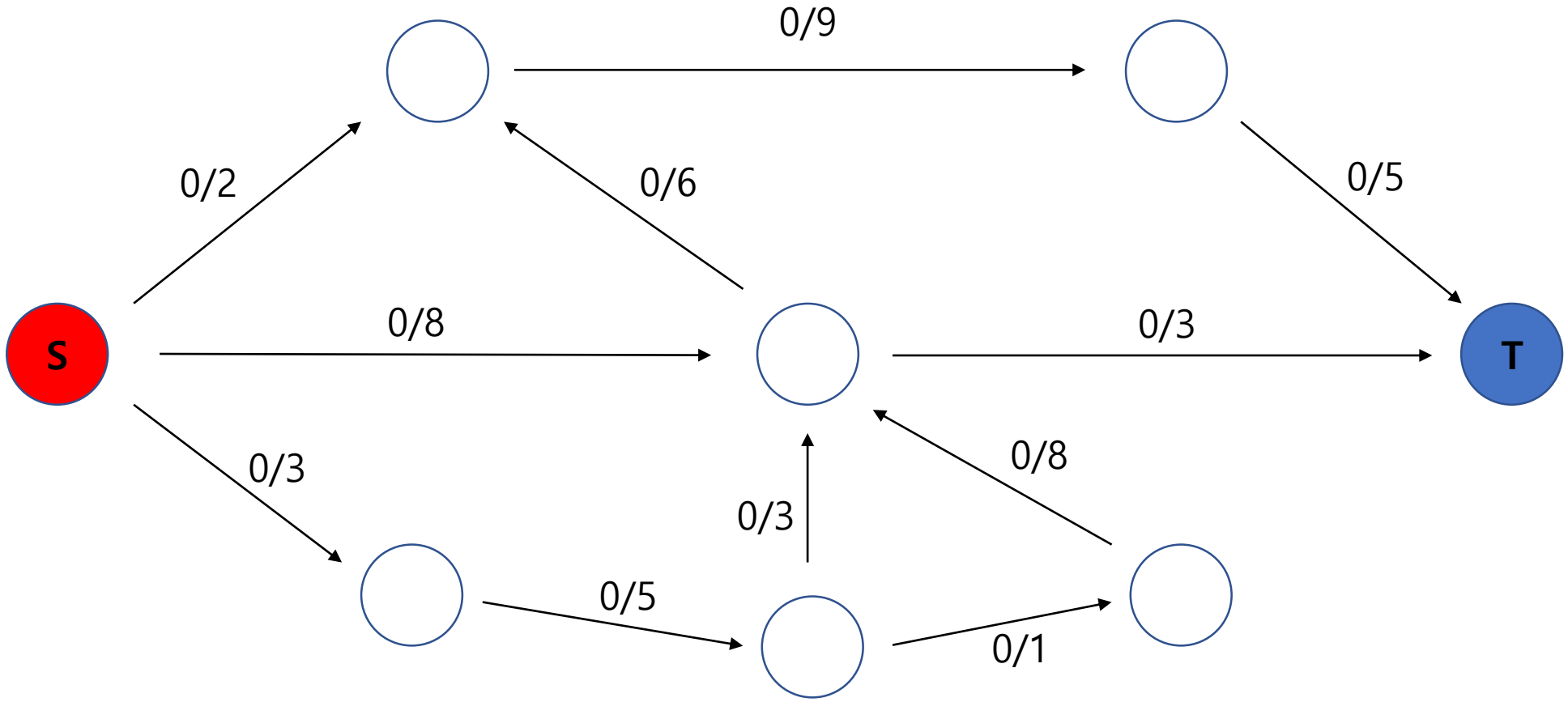
최대 유량 문제

가중치가 존재하는 방향 그래프를 생각해보자.
가중치는 파이프의 용량 이라고 생각하면 된다.



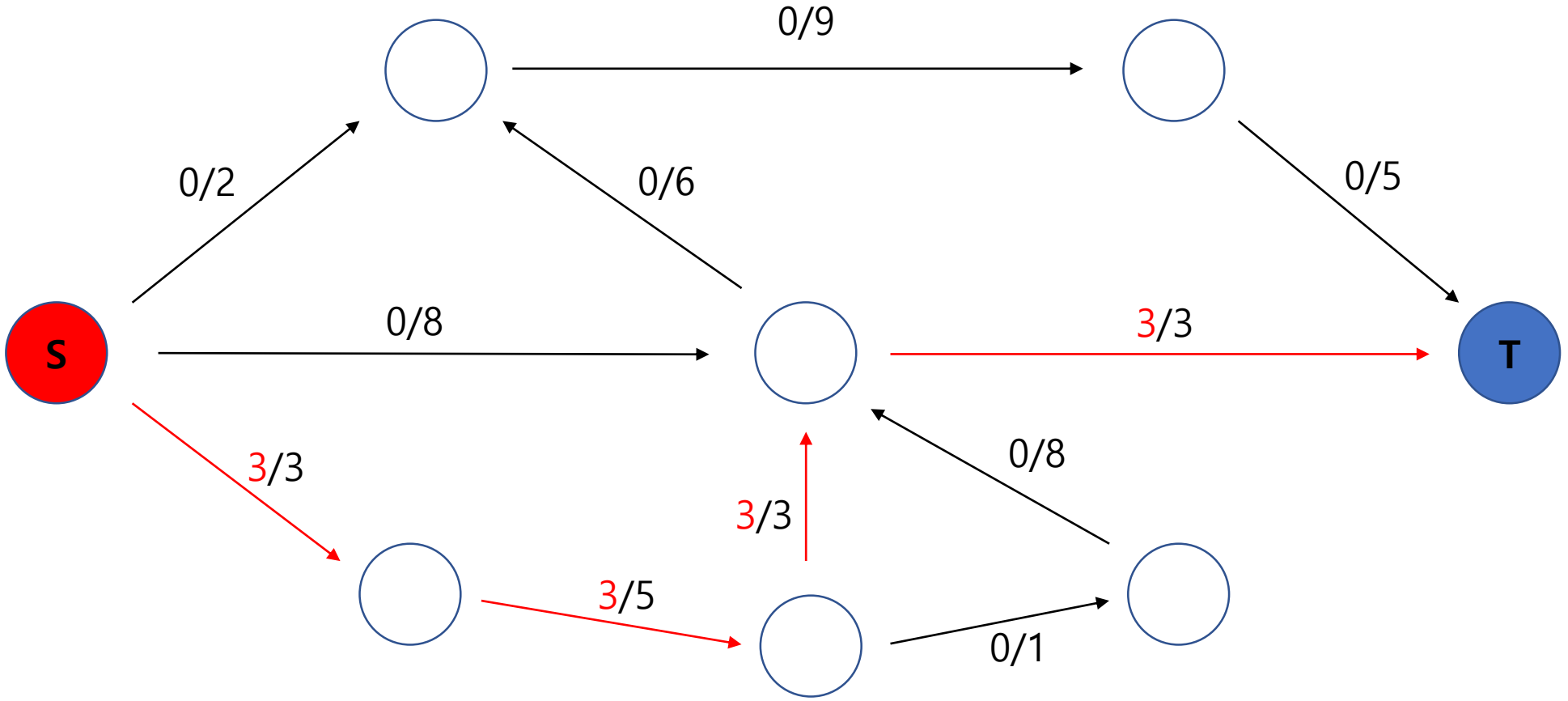
최대 유량 문제

물을 흘리기 전에는 이런 상태일 것이다.



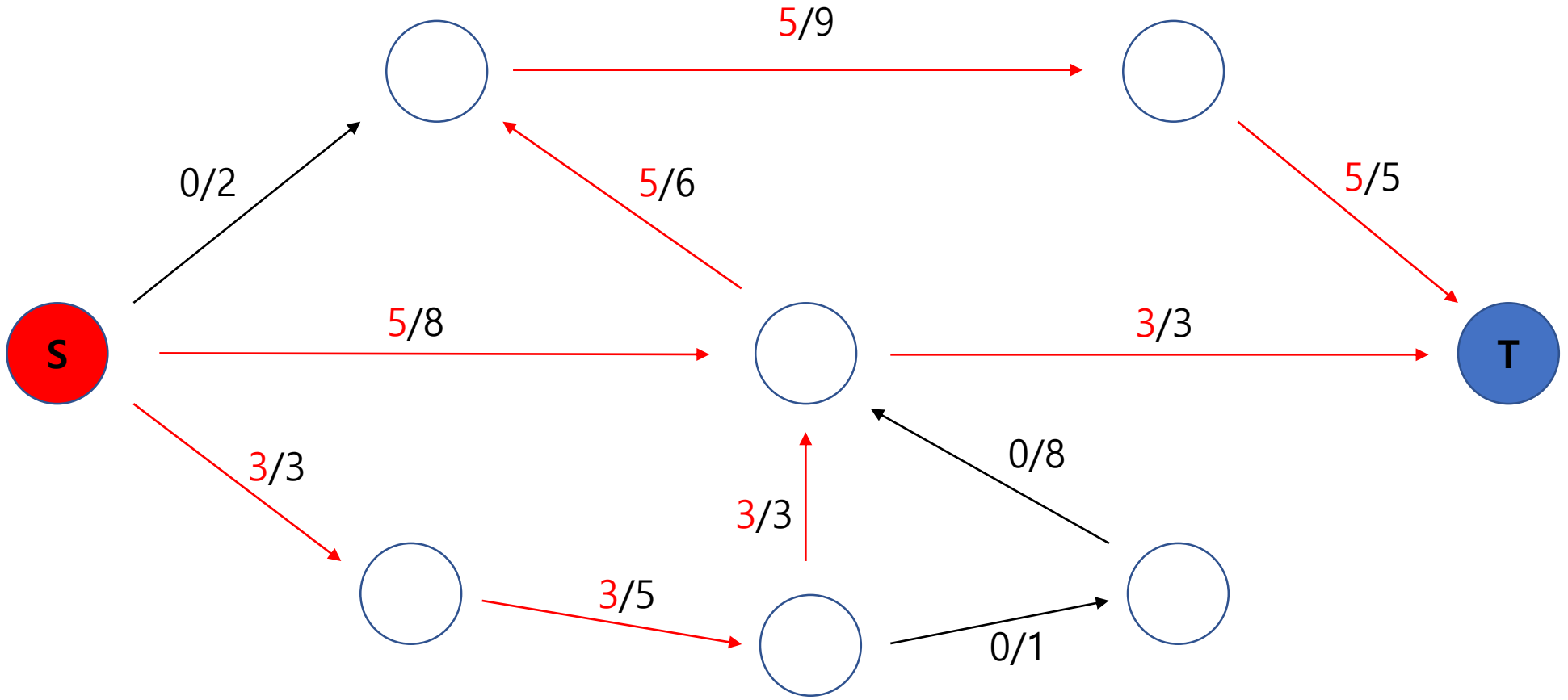
최대 유량 문제

이런 방식으로 흘려 보낼 수 있다.



최대 유량 문제

이렇게 흘려 보내면 최대로 흘려 보낼 수 있다.



최대 유량 문제_용어정리

유량 네트워크(flow network) : 간선의 가중치가 “용량”인 단방향 그래프

소스(source) : 유량이 시작하는 정점.

싱크(sink) : 유량이 도착하는 정점.

용량(capacity) : 각 간선에 대하여 흐를 수 있는 최대 유량(용량)

유량(flow) : 각 간선에 대하여 실제로 흐른 유량

최대 유량 문제_용량의 제한

유량(flow)는 그 간선의 용량을 초과할 수 없다.

flow <= capacity

$$f_{uv} \leq c_{uv}$$

최대 유량 문제_ 유량의 보존

소스와 싱크를 제외한 정점에는 들어오는 유량과 나가는 유량이 같다.

u에 들어오는 유량 = u에서 나가는 유량

$$\sum_{v:(u,v)\in E} f_{vu} = \sum_{v:(u,v)\in E} f_{uv}$$

(v는 u와 연결된 정점)

최대 유량 문제_유량의 대칭성

u->v로 f만큼의 유량이 흐르면 v->u로 −f만큼의 유량이 흐른다

$$f_{uv} = -f_{vu}$$

최대 유량 문제_최대유량

유량 네트워크의 최대유량은 소스에서 싱크로 흘러가는 유량의 크기이다.

소스에서 나오는 유량 = 싱크로 흘러 들어가는 유량

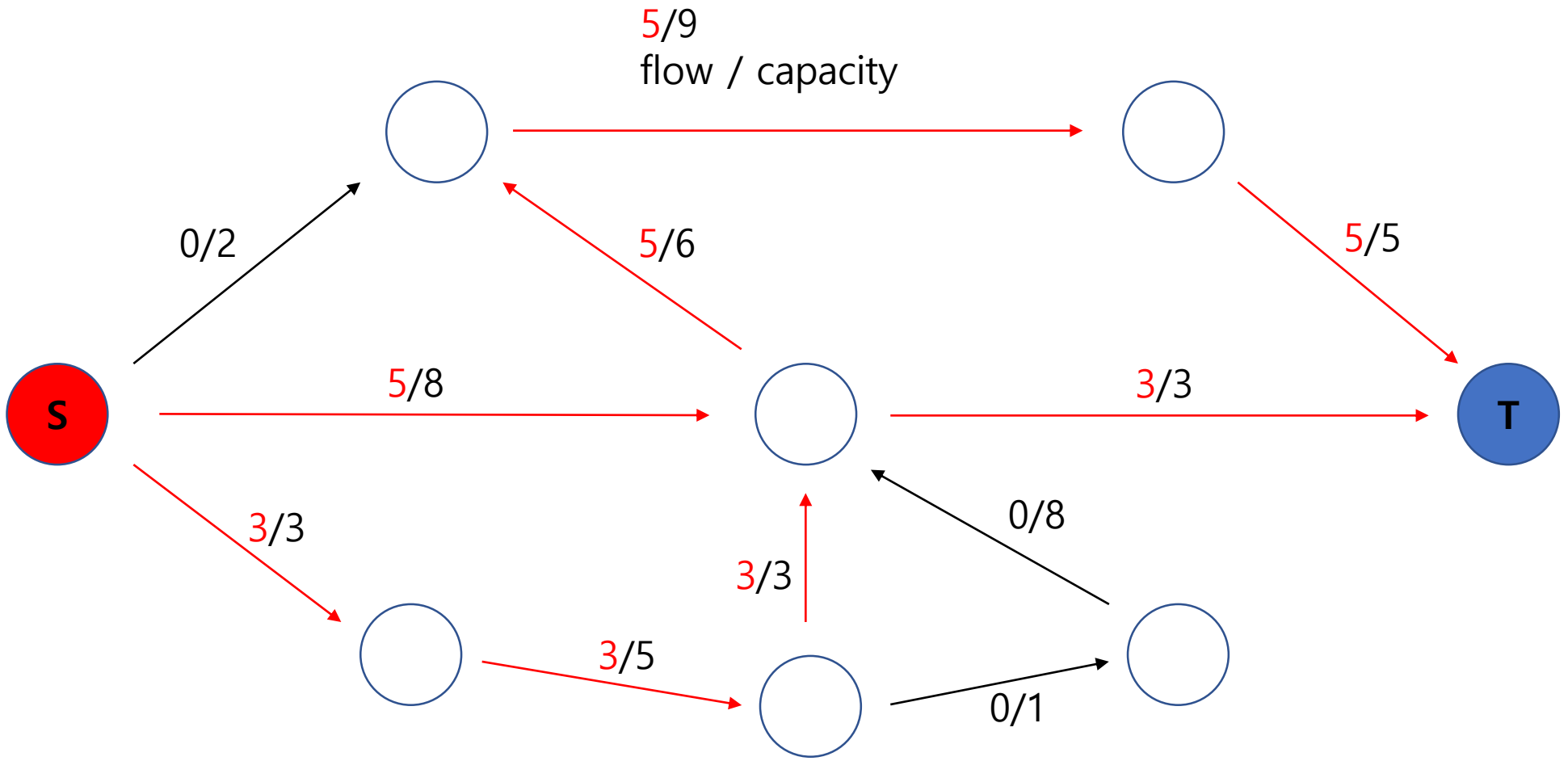
$$F = \sum_{u:(s,u)\in E} f_{su} = \sum_{v:(v,t)\in E} f_{vt}$$

s:source, t:sink

이걸 구하는게 문제의 답입니다.

최대 유량 문제

이걸 보면서 한번 이해해보세요.



예제

6086 – 최대 유량

이 문제를 풀거에요!
지금은 아니고 천천히 같이 풀건데
혹시 문제를 읽어 보면 더 이해가 쉬워질수도 있으니까…

Ford Fulkerson Algorithm

가장 기본적인 유량 알고리즘

Ford Fulkerson Algorithm_개념

1. 증가 경로(계속 흐를 수 있는 경로)를 찾는다.(DFS 이용)

2. 증가 경로로 흘릴 수 있는 만큼의 유량을 흘려준다
   -> 경로 상의 모든 f(u,v)를 증가시킨다

3. f(u,v)를 증가시킨 만큼 **역방향 간선**에 감소시킨다.

4. 더 이상 흐를 수 없을 때까지(증가 경로가 없을 때가지) 위의 과정을 반복한다.

Ford Fulkerson Algorithm_역방향 간선

역방향 간선이란?

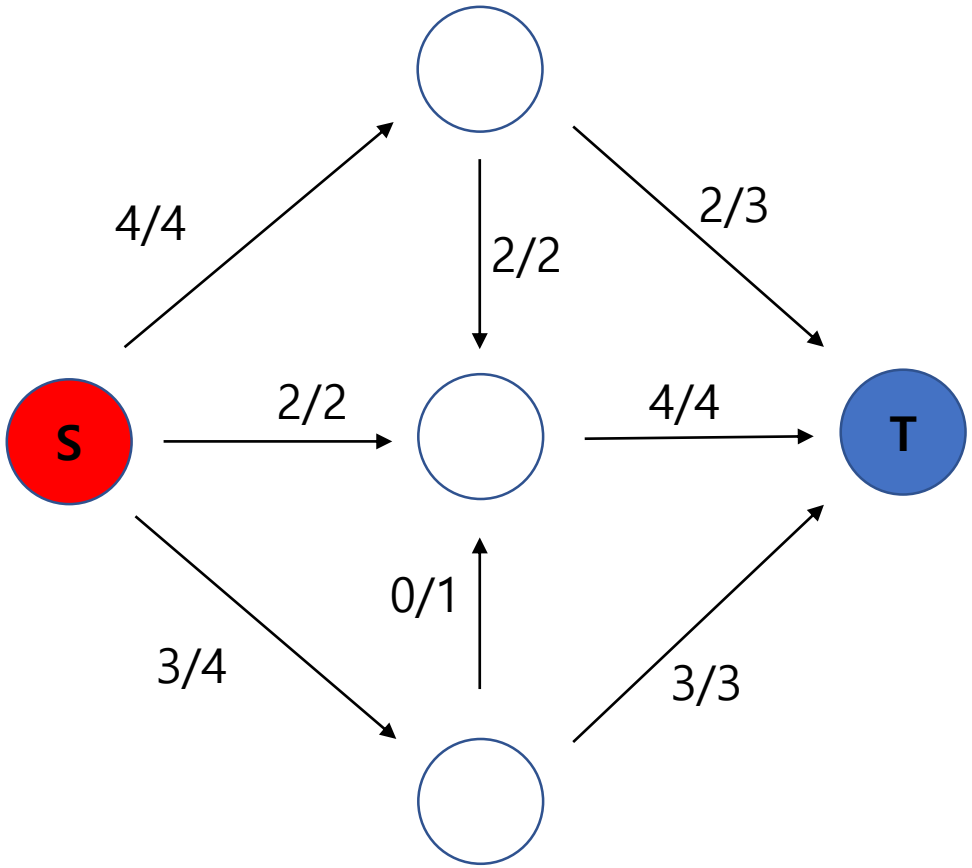
간선의 용량이 c, 유량이 f인 간선 A가 있다면,

A의 역방향 간선은
용량은 0(무조건)
흐르는 양은 -f이다.

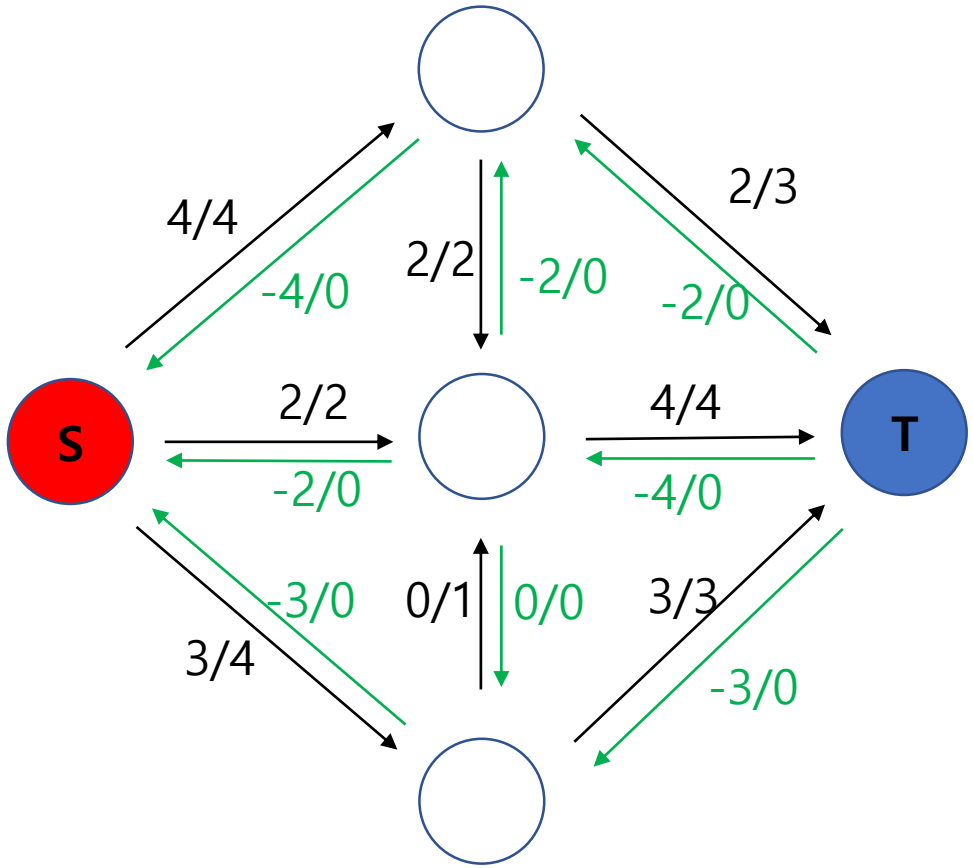
실제론 존재하지 않지만 이렇게 생각하면 문제 풀 때 도움이 많이 된다.
실제로도 코드로 구현하게 되는 간선이다.

Ford Fulkerson Algorithm_역방향 간선

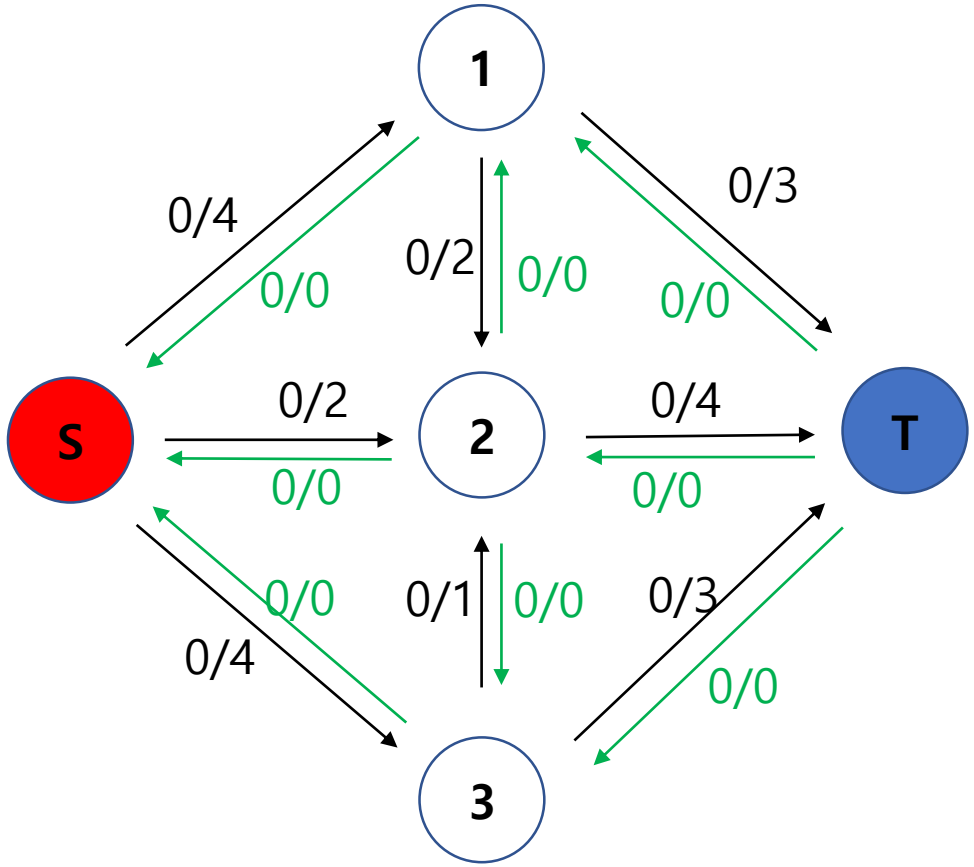
실제 그래프



역방향 까지 포함한 그래프



Ford Fulkerson Algorithm_개념



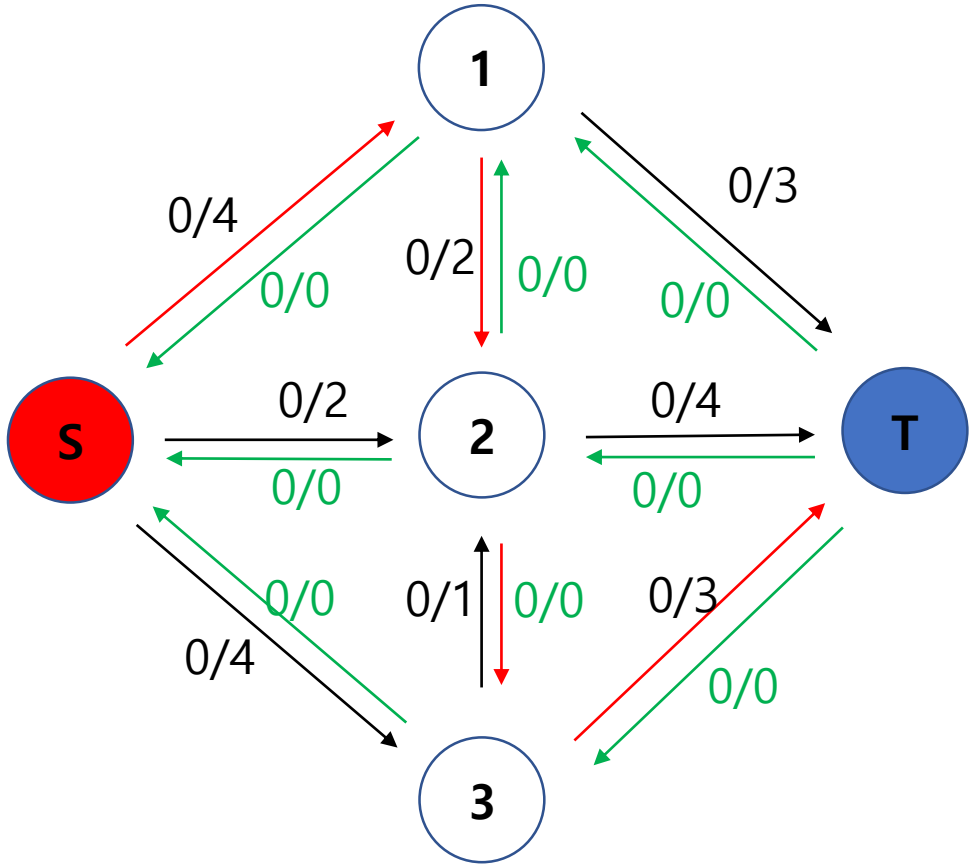
초기 상태입니다.

지금부터 잘 보세요.

역방향 간선도 똑같은 간선이라 생각하고 보세요.

탐색 순서는 source,1,2,3,sink 순서 입니다.

Ford Fulkerson Algorithm_개념



첫번째 DFS결과입니다.(visit을 이용해 들린 곳은 pass)

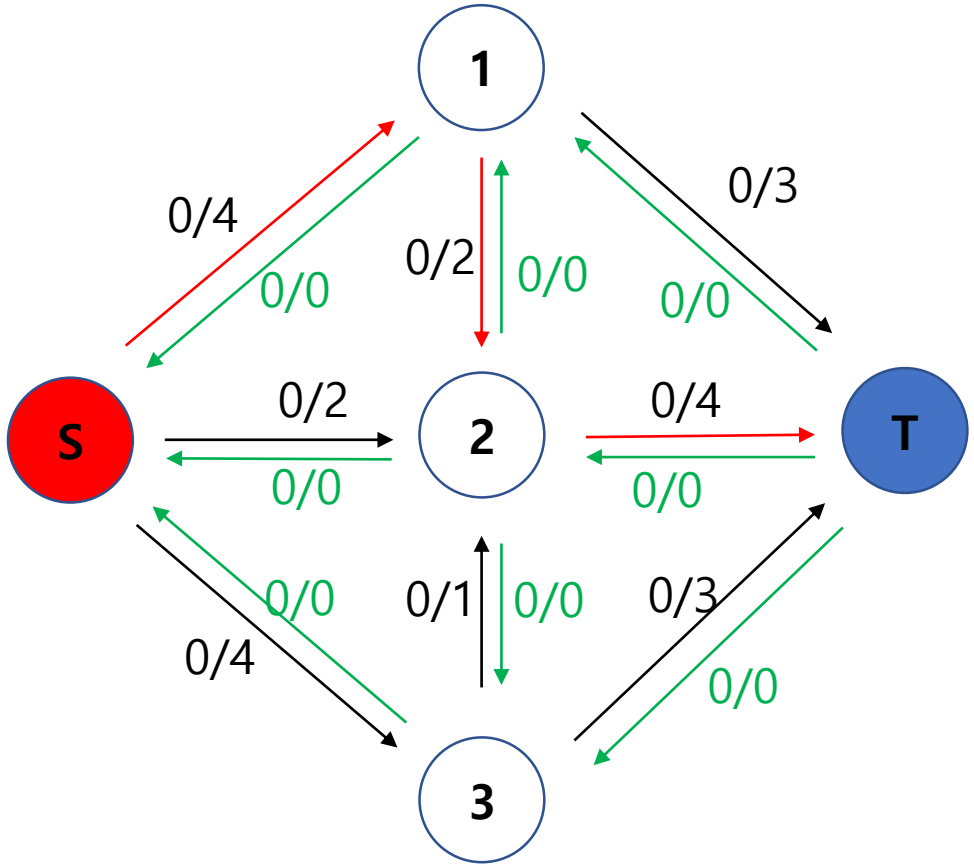
S->1->2->3->T
인 경로가 있습니다.

2->3 에서 flow < capacitiy인 상태가 아니기 때문에
물을 더 흘려 보낼 수 없죠? (0 < 0)

그러므로 이 경로는 증가 경로가 아닙니다.

(실제 구현에서는 2->3을 가려고 시도조차 하지 않을거에요)

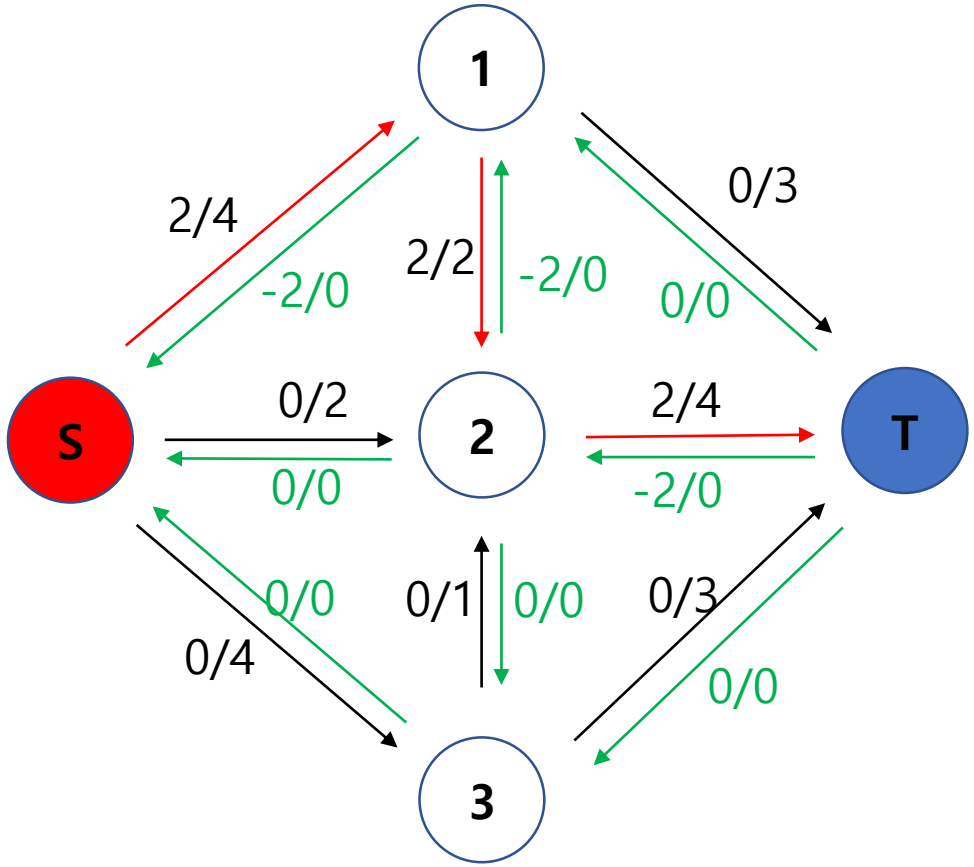
Ford Fulkerson Algorithm_개념



S->1->2->T

이 경로에서는 최대 2만큼 흘려 보낼 수 있습니다.
1->2에서 2까지 밖에 못 흘려보내니까.

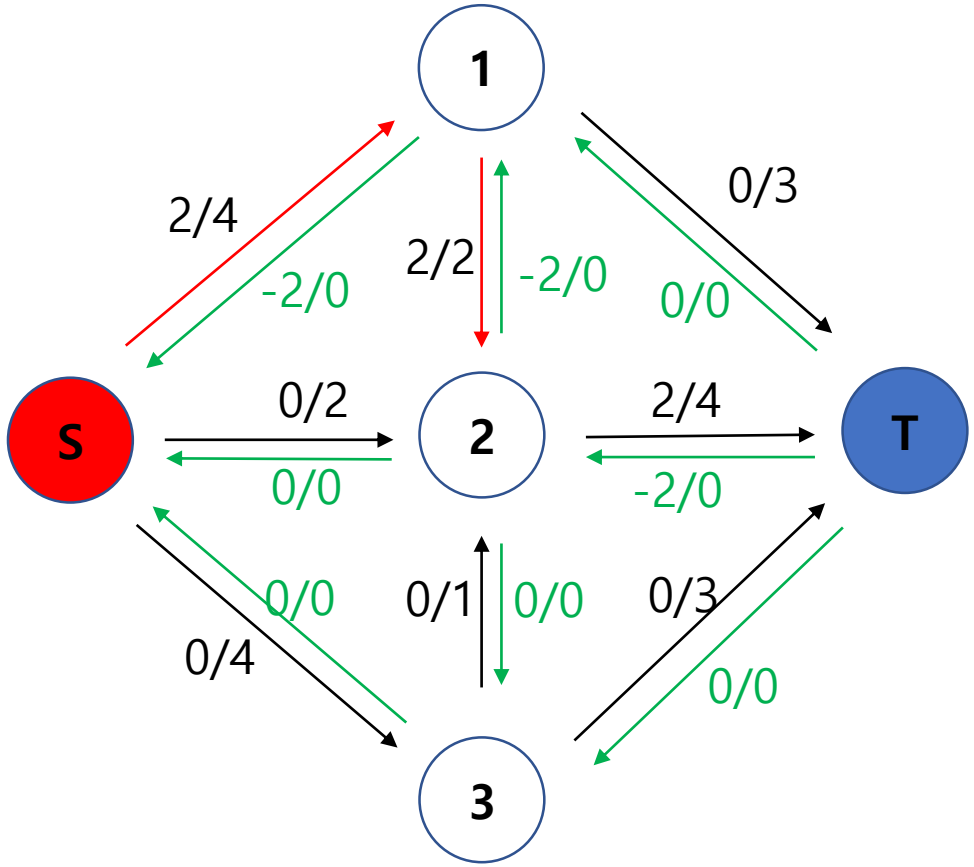
Ford Fulkerson Algorithm_개념



S->1->2->T

역방향 간선까지 갱신한다면 이런 모양이 되겠죠?

Ford Fulkerson Algorithm_개념



여기서 중요합니다!!!

DFS가 한번 끝나면(증가 경로를 찾아 물을 한번 흘려 주면)

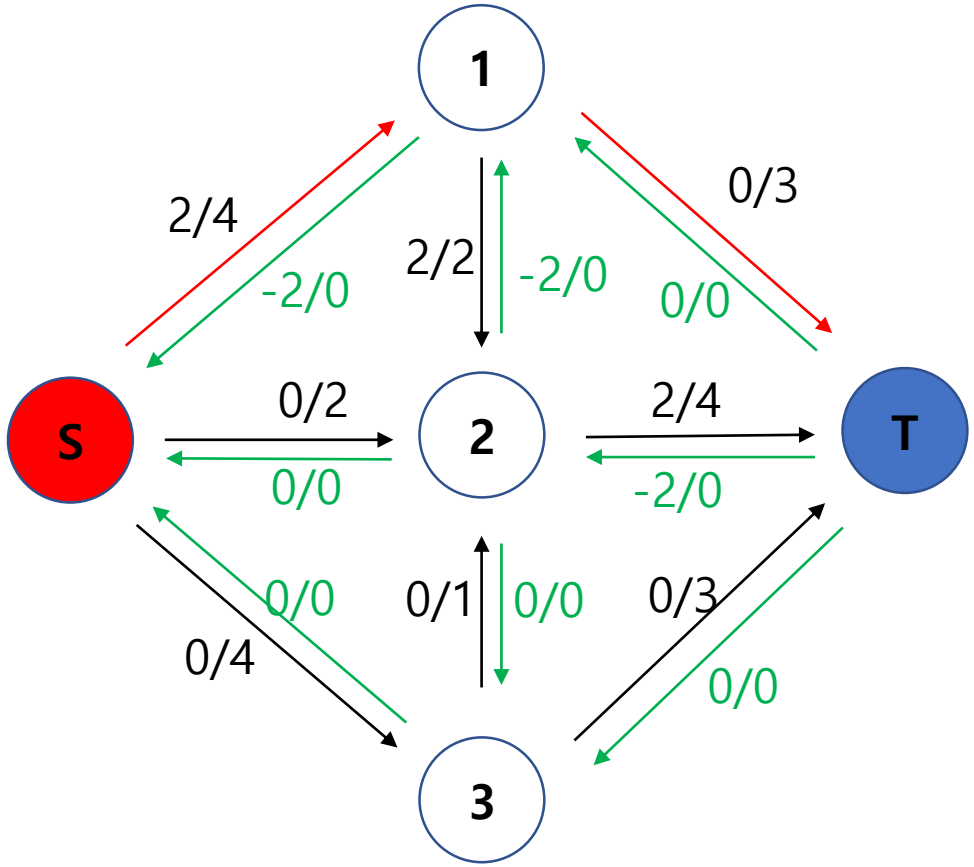
Source 부터 새로 DFS를 할거에요.(visit 초기화)

아니 그럼… 똑같은 경로로 또 가는거 아닌가요?
이미 한 번 물을 흘려 보내주었기 때문에
1->2 로 가는 경로가 막혔죠?

방문하지 않았다 or 흐를 수 있다(capacity > flow)
일 때 탐색을 합니다.

1에서 2로 가지 못하니 새로운 길을 찾아 탐색을 할거에요.

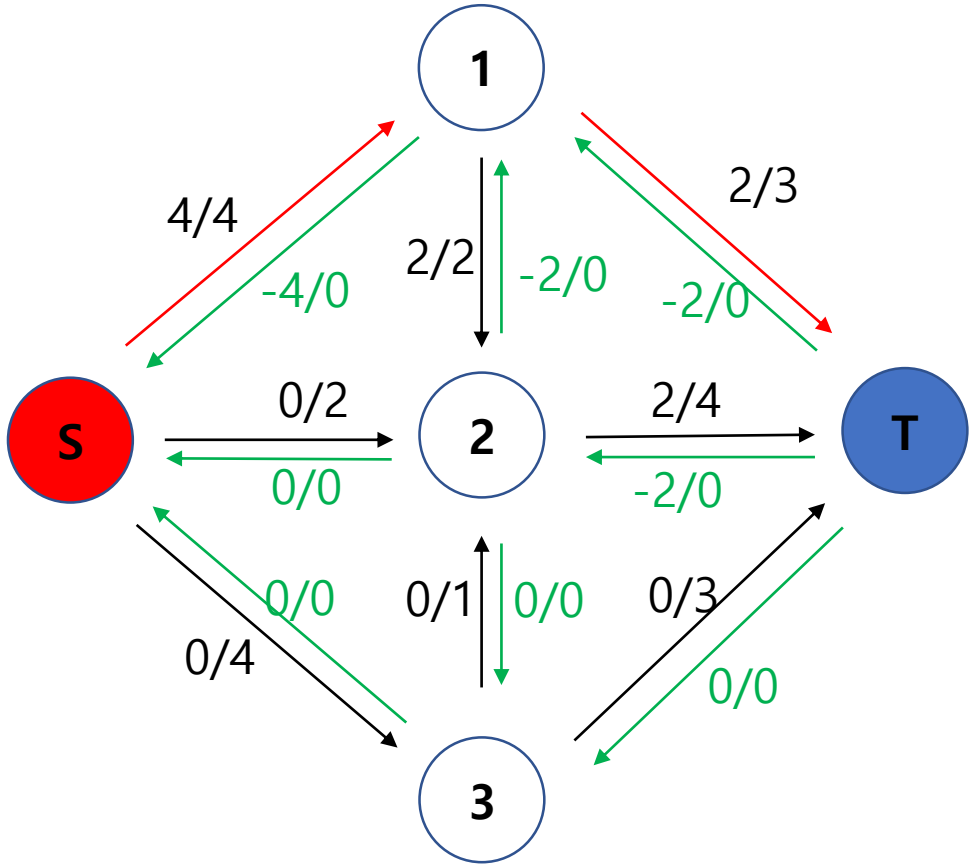
Ford Fulkerson Algorithm_개념



S->1->T

2만큼 흐를 수 있어요.

Ford Fulkerson Algorithm_개념

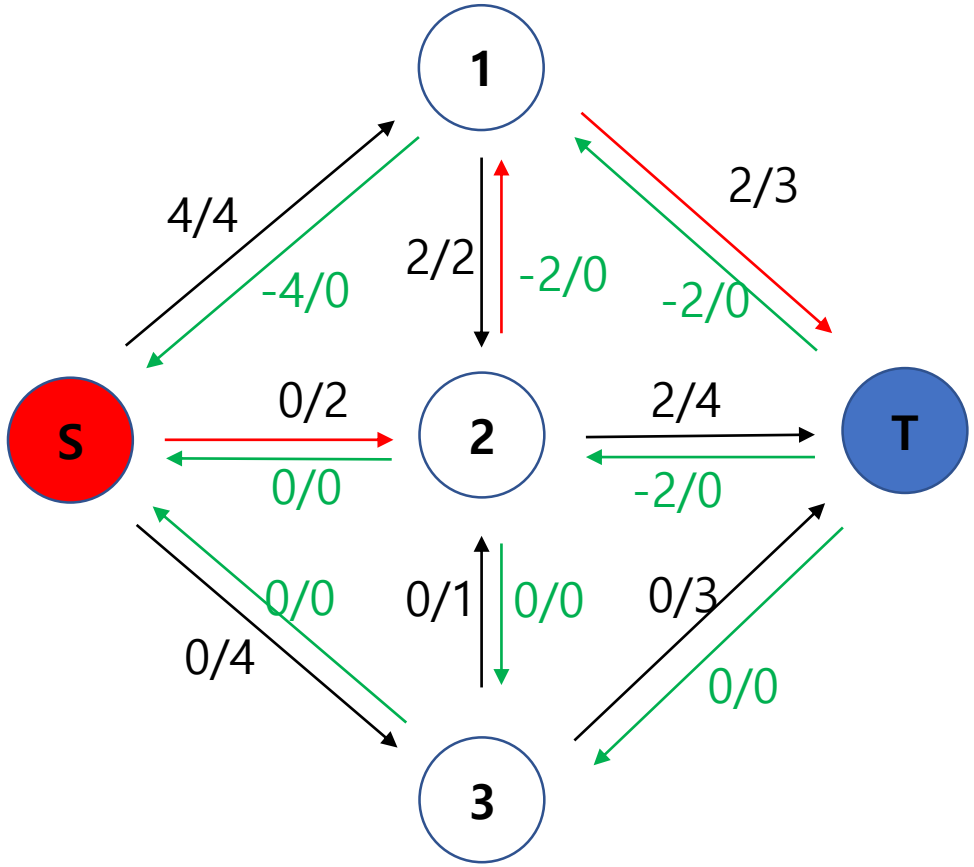


S->1->T

갱신

계속 DFS해볼까요?

Ford Fulkerson Algorithm_개념



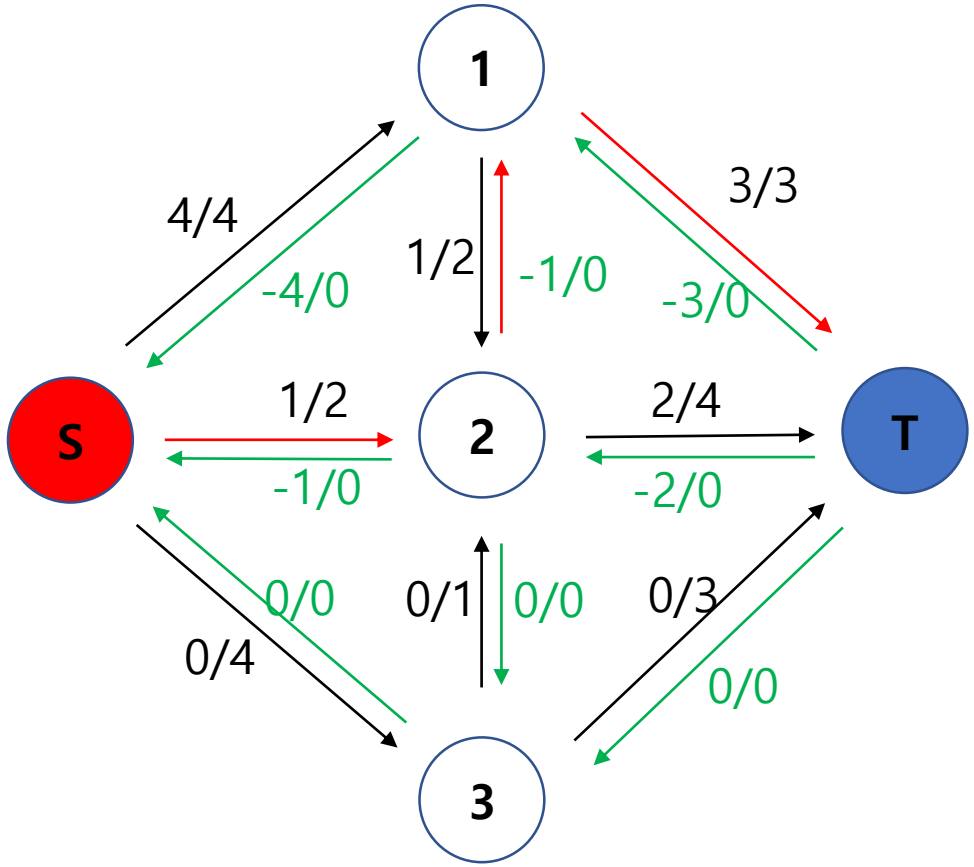
S->2->1->T

역방향 간선도 똑같은 간선으로 생각합니다.

0/20이면 2-0=2 니까 2만큼 더 흐를 수 있죠.
-2/00이면 0-(-2)=2 니까 2만큼 흐를 수 있죠?

이 경로는 1만큼 더 흐를 수 있네요.

Ford Fulkerson Algorithm_개념



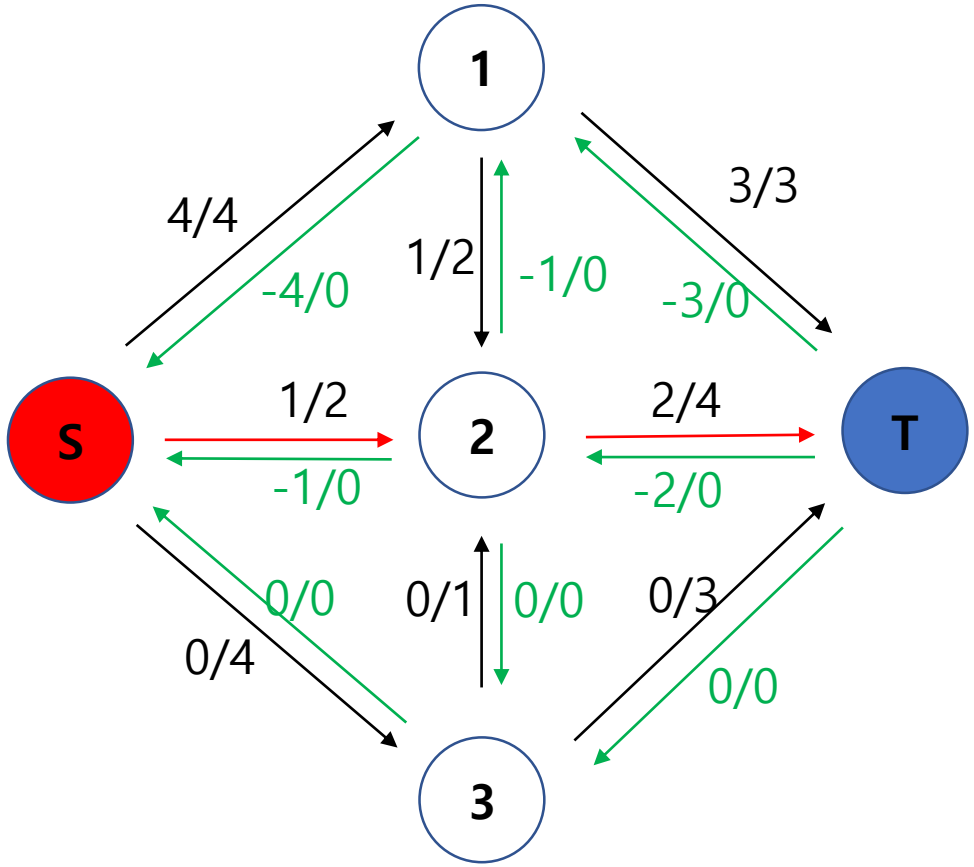
S->2->1->T

어떻게 갱신 되는지 이해가 되나요???

실제로는 1->2로 2만큼 왔던게 1만큼 오고
나머지 1이 1->T로 갔음을 알 수 있어요.

역방향 간선과 함께 생각한다면 더욱 이해하기 쉽죠?

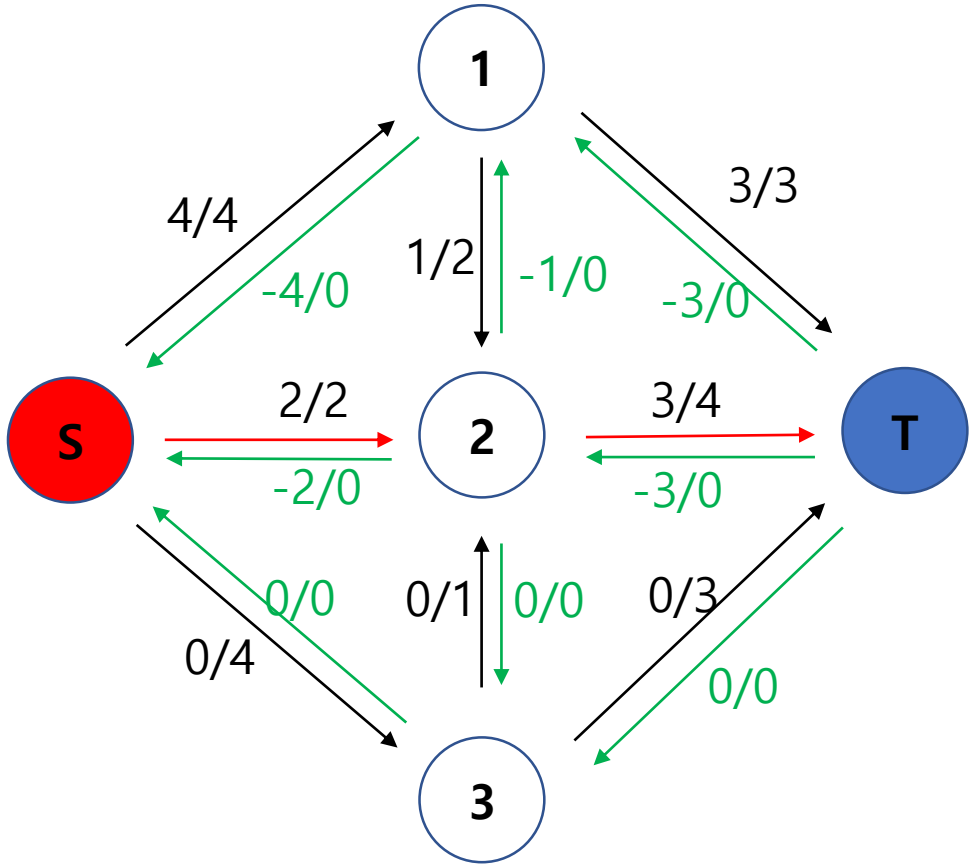
Ford Fulkerson Algorithm_개념



S->2->T

1만큼 더 흐를 수 있네요.

Ford Fulkerson Algorithm_개념

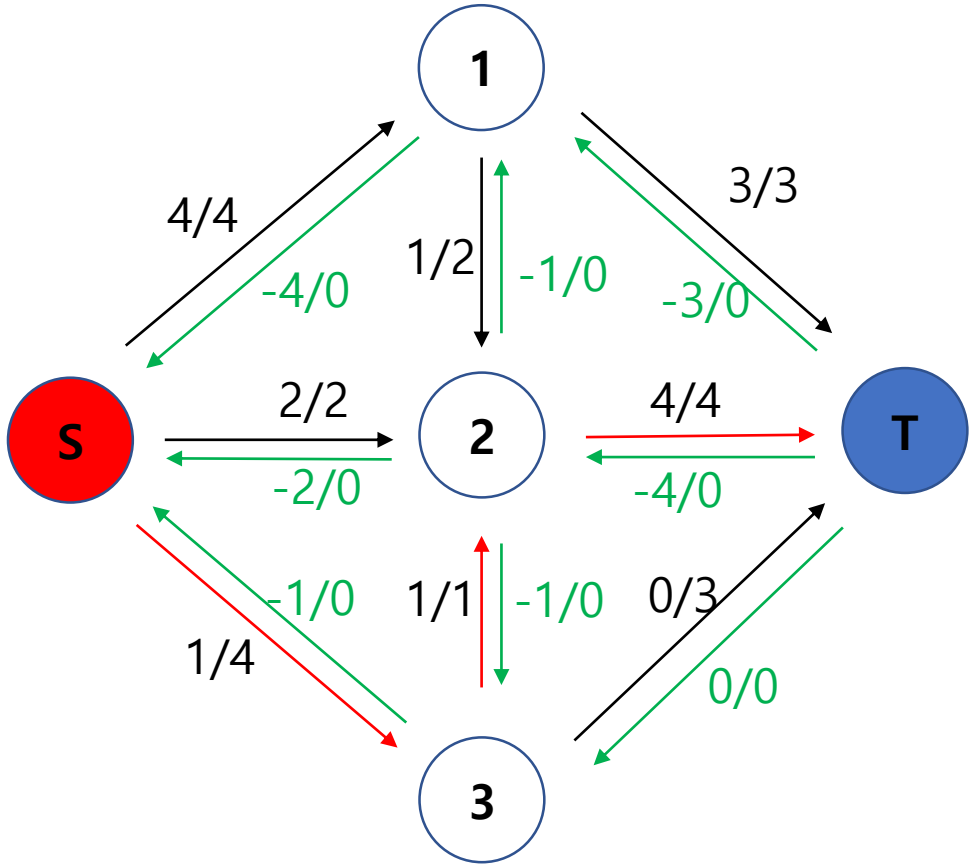


S->2->T

갱신 완료.

S에서는 이제 3으로 밖에 못가네요.

Ford Fulkerson Algorithm_개념



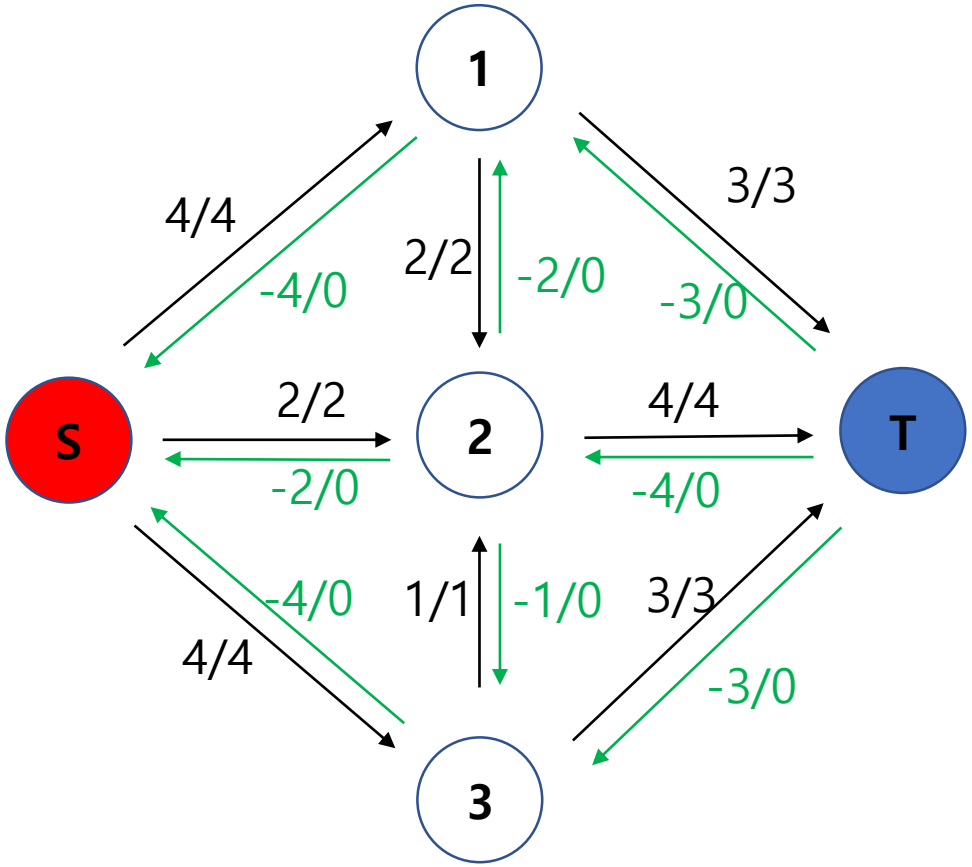
S->3->2->T

갱신 완료.

제가 갱신까지 해버렸어요.
이젠 느낌이 오시죠?

마지막S->3->T를 끝내면

Ford Fulkerson Algorithm_개념



완성!!!

답은 얼마일까요?

~~4+2+4 = 3+4+3 = 10~~

예제

6086 – 최대 유량

6086_최대유량

- 다음 장의 코드를 보면서 풀어보아요.

~~자신이 있다면 넘기지 마시오.~~

6086_최대유량_자료구조

Class를 이용하면 편리합니다.

```cpp
#include <climits>

#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class MaximumFlow
{
public:
    class Edge
    {
    public:
        int from, to, capacity, flow; // 간선을 이루는 요소들
        Edge *reverse;                // 역방향 간선
        Edge(void) :
        from(0), to(0), capacity(0), flow(0), reverse(nullptr)
        {
        }
        Edge(const Edge &edge) :
        from(edge.from), to(edge.to), capacity(edge.capacity), flow(edge.flow), reverse(edge.reverse)
        {
        }
        Edge(int from, int to, int capacity) :
        from(from), to(to), capacity(capacity), flow(0), reverse(nullptr)
        {
            // class말고 struct로 하면 이 생성자만 있으면 됩니다.
        }
    };
```

6086_최대유량_자료구조

일단 따라해보세요.

```
30      int N, source, sink;          // 정점의 갯수, 소스, 싱크
31      vector<vector<Edge *>> Graph;
32      // 주어진 그래프; Graph[i] : i에서 출발하는 Edge들, 인접리스트 형식이라고 생각하면 됨.
33      vector<bool> visit;           // dfs를 위해 필요한 visit
34
35      MaximumFlow(void) :
36      N(0), source(0), sink(0), Graph(), visit()
37      {
38      }
39      MaximumFlow(const MaximumFlow &mf) :
40      N(mf.N), source(mf.source), sink(mf.sink), Graph(mf.Graph), visit(mf.visit)
41      {
42      }
43      MaximumFlow(int N, int source, int sink) :
44      N(N), source(source), sink(sink), Graph(N), visit(N)
45      {
46          // class말고 struct로 하면 이 생성자만 있으면 됩니다.
47      }
```

6086_최대유량_자료구조

자 여기까지 기본 세팅입니다.

아직 MaximumFlow class 안에 있습니다.

```
48
49      void add_edge(int from, int to, int capacity)
50      {
51          // 간선을 추가 하는 함수
52          Edge *e = new Edge(from, to, capacity);
53          Edge *e_reverse = new Edge(to, from, 0);
54          Graph[from].push_back(e);
55          Graph[to].push_back(e_reverse);
56          e->reverse = e_reverse;
57          e_reverse->reverse = e;
58      }
```

6086_최대유량_자료구조

Ford Fulkerson 알고리즘의 dfs 부분입니다.

```
60      int dfs(int now, int mf)
61      {
62          // now : 현재 정점
63          // mf : 현재까지 흐를 수 있는 최대 유량
64          if (now == sink)
65          { // sink까지 도달하면 흐를 수 있는 가장 큰 값을 리턴한다.
66              return mf;
67          }
68          if (visit[now])
69          { // 방문한 적 있다면 back
70              return 0;
71          }
72          visit[now] = true;
73
74          for (auto e : Graph[now]) // 현재에서 출발하는 모든 간선에 대해
75          {
76              if (e->capacity - e->flow > 0) // 흐를 수 있다면
77              {
78                  int flow = dfs(e->to, min(mf, e->capacity - e->flow));
79                  // dfs를 통해  흐를 수 있는 유량을 찾는다.
80                  if (flow > 0)                                       // 찾았다면
81                  {
82                      e->flow += flow;          // 정방향으로 흘려준다
83                      e->reverse->flow -= flow; // 역방향으로 흘려준다
84                      return flow;              // 얼마나 흘렀는지 리턴해준다.
85                  }
86              }
87          }
88
89          return 0;
90      }
```

6086_최대유량_자료구조

최대유량을 구하는 Ford Fulkerson 알고리즘입니다.

class도 끝났네요.

```
91
92      int maximum_flow(void) // 진짜 답을 구하는 함수
93      {
94          int ret = 0;
95          while (true)
96          {
97              int flow = dfs(source, INT_MAX);
98              fill(visit.begin(), visit.end(), false);
99              if (flow == 0) // 증가 경로를 못찾으면 break
100             {
101                 break;
102             }
103             ret += flow;
104         }
105         return ret;
106     }
107 };
```

6086_최대유량

코드를 보고도 이해가 어려울 수 있어요.

코드를 띄워 놓고 pdf의 앞부분으로 돌아가서 차근차근 시뮬레이션 해보면 감이 오실 거에요.

이게 진짜 멘토링 하면서 서로 마주 보고 설명해도 어려운 부분인데…

Pdf에는 너무 많은 양을 담아서도 안되고, 어떻게 해도 설명이 빈약할 수 밖에 없어요…

강사에게 직접 연락하거나 알로하 질문방에 질문하면 잘 답변해 드릴게요ㅠ

이제 main함수로 넘어갈까요?

6086_최대유량_자료구조

자.. 여기까지 하면 이제 입력 받을 준비가 되네요.

MF가 다 알아서 해줄거에요!!

```
108
109 int N;
110 char pipe_start, pipe_end;
111 int pipe_capacity;
112 int alpha_to_number[200];
113
114 int main(void)
115 {
116     ios_base::sync_with_stdio(false);
117     cin.tie(NULL);
118
119     // 입력으로 들어오는 알파벳을 숫자로 변환시킨다.
120     for (int i = 0; i < 26; i++)
121     {
122         alpha_to_number['A' + i] = i;
123         alpha_to_number['a' + i] = i + 26;
124     }
125     // A->0 Z->25 a->26
126
127     MaximumFlow MF(100, 0, 25); // 100개의 정점이 있고, 0에서 시작, 25에서 끝
```

6086_최대유량_자료구조

끝!

```
128
129     cin >> N;
130     for (int n = 1; n <= N; n++)
131     {
132         cin >> pipe_start >> pipe_end >> pipe_capacity;
133         MF.add_edge(alpha_to_number[pipe_start], alpha_to_number[pipe_end], pipe_capacity);
134         MF.add_edge(alpha_to_number[pipe_end], alpha_to_number[pipe_start], pipe_capacity);
135     }
136
137     cout << MF.maximum_flow() << "\n";
138
139     return 0;
140 }
```

예제

6086 – 최대 유량

잘 따라 왔다면 다들 “맞았습니다!!”를 받았을 거에요.
이제 스스로의 힘으로 해결해 볼까요?

어떠한 방법도 좋아요.
자신만의 스타일로 “맞았습니다!!”를 받아봅시다!

과제

17412 – 도시 왕복하기 1

최대 유량 찾기 알고리즘들

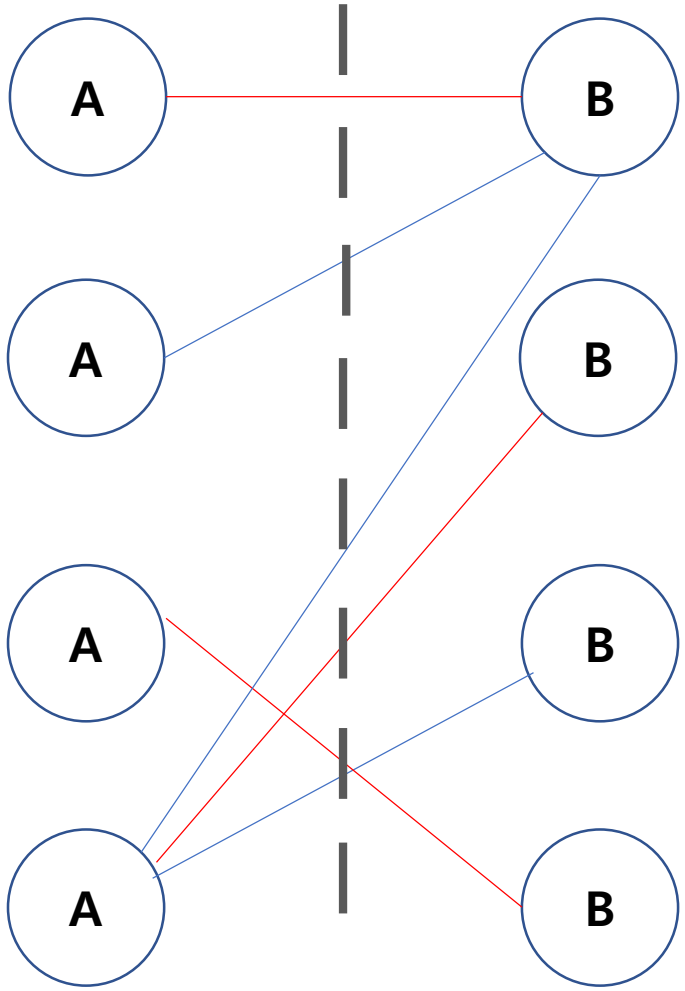
- DFS 이용 (Ford Fulkerson, 지금 배운것)
- BFS 이용 (Edmond Karp, 궁금하신 분들 인터넷으로 공부하세요ㅠ)
- BFS, DFS 이용 (Dinic, 다음주에 배웁니다.)

Bipartite Matching

이분 매칭

갑자기?

Bipartite Matching_개념



이분 그래프

정점을 두 집합(A 집합과 B 집합)으로 나누었을 때
모든 간선의 양쪽 정점이 A 집합과 B 집합에 각각 속하는 그래프

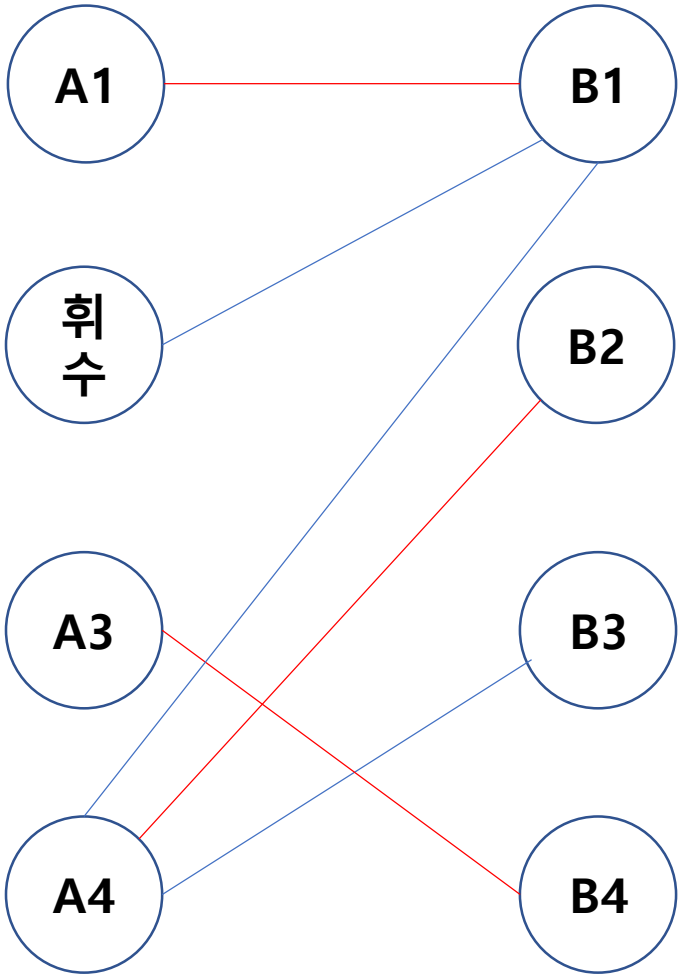
$$E_{uv}(u \in A, v \in B)$$

최대 이분 매칭

이분 그래프에서 간선을 선택하는데
각각의 간선들끼리 같은 정점을 공유하지 않게 고를 때
최대로 고를 수 있는 간선 수

- 왼쪽 그래프에서의 빨간 간선

Bipartite Matching_사랑의 작대기



A 그룹을 남자, B그룹을 여자라고 생각해보자.

간선을 서로 간의 호감의 표시라고 가정해보자.

동성교제는 없다고 가정한다.(이분 그래프의 정의)

여러명을 동시에 만날 수 없다.(이분 매칭의 정의)
여러명에게 대쉬할수는 있다!~~포기하지말고 들이대~~

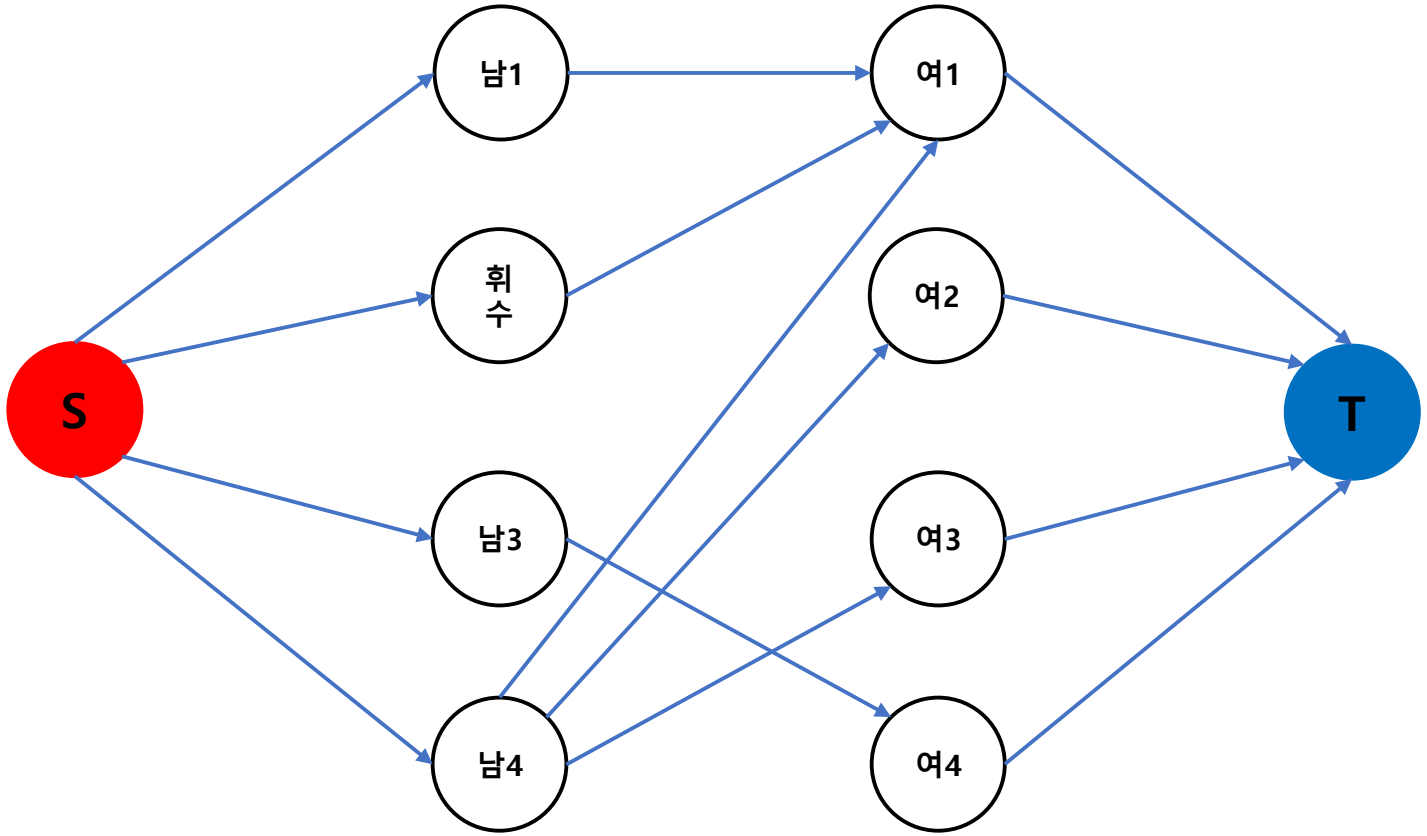
우리의 목표는 여기서 가장 많은 커플을 만드는 것이다.~~여욱사 컴공~~

여기서 최대한 많은 커플을 만들어야 한다.(최대 이분 매칭)

그래도 안될놈은 안돼~~휘수~~

Bipartite Matching_사랑의 작대기

이렇게 모델링 해보자

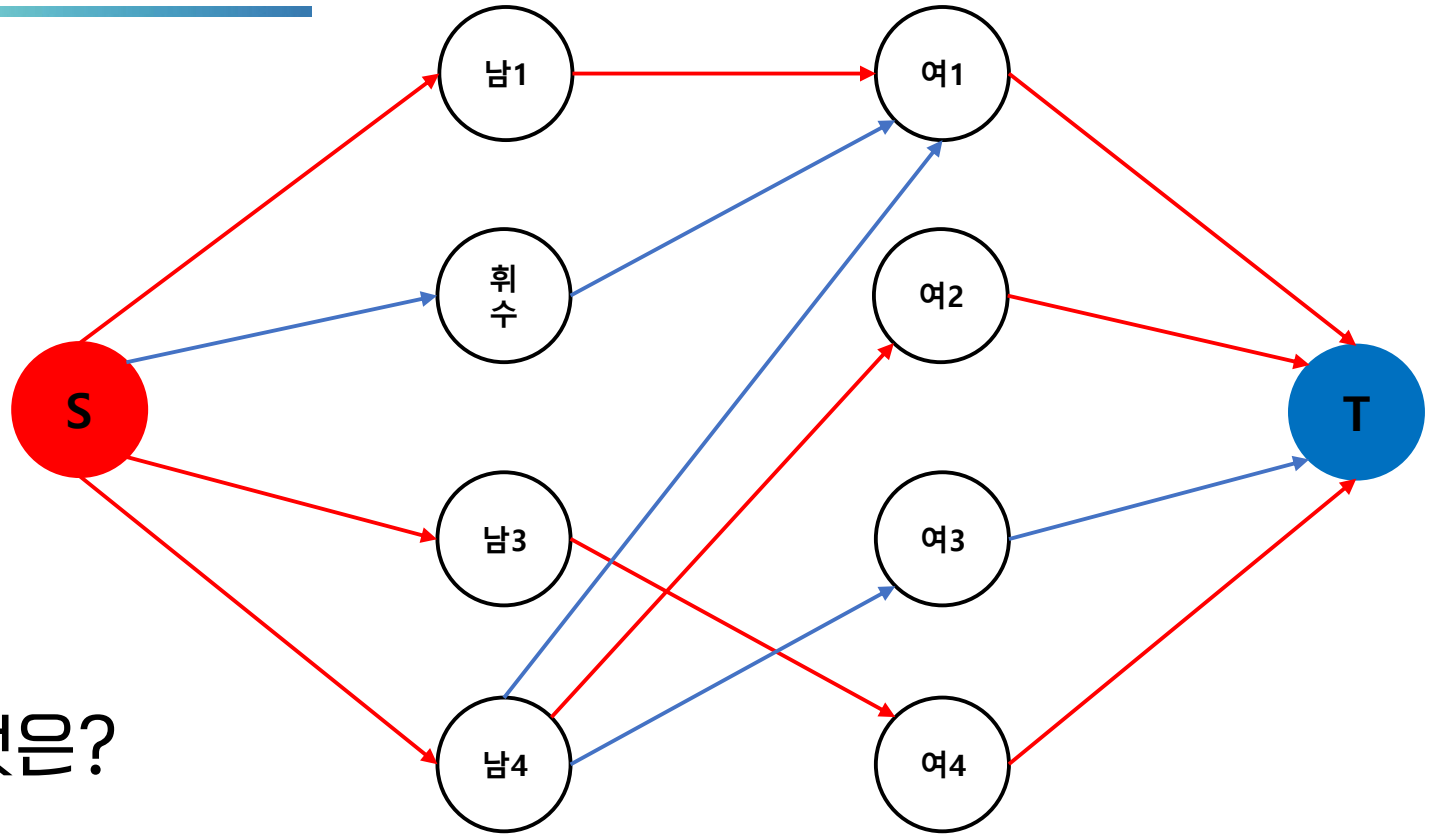


남자들이 여자들에게 대쉬 한다고 가정해 보자
화살표는 대쉬의 방향이다.

여자가 남자의 고백을 승낙한다
-> 간선이 선택되었다.
-> 간선을 통해 사랑이 흐른다.
-> 흐른다?

그래프를 유량 네트워크라 생각하고
각 간선의 용량을 1이라 하면?
유량이 흐른다는 것은?

Bipartite Matching_사랑의 작대기



유량이 흐른다는 것은?

커플의 탄생을 알리는 것이다.

Bipartite Matching & Maximum Flow

이분 그래프에서 최대 이분 매칭과 최대유량은 같다!

이분 매칭과 관련된 알고리즘들
Minimum Vertex Cover
Maximum Independent Set
Minimum Path Cover
Maximum Anti-Chain

여기에 담기엔 너무 많아서ㅜ
궁금하시면 공부해보세요.

예제

2188 - 축사배정

2188_축사배정

- 이게 바로 사랑의 작대기 아닙니까?

- source 에서 소들까지 연결해주고, 축사에서 sink까지 연결해서(capacity 0) 최대 유량을 구하면?

2188_축사배정

```
// 1<= N,M <=200 이므로
 // source:0, sink:1000
 // 소번호는 그대로, 축사번호는 번호+1000으로 설정
 // source:0, 1 ~ N, N+1 ~ N+M, sink:N+M+2 등 설정 자유롭게

 MaximumFlow MF(2000, 0, 1000);

 cin >> N >> M;
 for ( int n = 1; n <= N; n++ )
 {   // 소스에서 소들까지 연결
     MF.add_edge(MF.source, n, 1);
 }
 for ( int m = 1; m <= M; m++ )
 {   // 축사에서 싱크까지 연결
     MF.add_edge(1000 + m, MF.sink, 1);
 }
```

```
for ( int n = 1; n <= N; n++ )
{   // 소와 축사 연결해주고
    cin >> S;
    for ( int s = 1; s <= S; s++ )
    {
        cin >> Si;
        MF.add_edge(n, 1000 + Si, 1);
    }
}

cout << MF.maximum_flow() << "\n";
```

main함수 부분입니다.

나머지는 최대유량과 동일합니다.

MCMF

최대유량을 배웠습니다.

그런데 우리가 궁극적으로 하고자 하는 것은
최대 유량을 흘리는데 최소 비용으로 흘리고자 하는 것입니다.

사실 생각을 조금만 해보면 어렵지 않습니다.
~~(이거 몰라도 과제 다 풀 수 있음. 풀고 나서 아 이게 MCMF였구나 할 거임. 내가 그랬음.)~~

여기에 담기엔 너무 많아서ㅜ 여기를 클릭하세요

스스로 공부해보고 궁금한점 있으면 언제든 질문해주세요!

과제

- 고급반 답게 과제에 대한 해설은 하지 않겠습니다.
- 알고리즘을 완벽하게 이해하는 것이 물론 중요하지만
- 문제를 해결하는 데에는 '모델링'이 더욱 중요합니다.
- 궁금한 점이 있으면 언제든 질문해주세요~

과제

11375 - 열혈강호

과제

11376 – 열혈강호2

과제

1671 – 상어의 저녁식사

과제

1017 – 소수 쌍

과제

2316 – 도시 왕복하기 2

과제

2316 – 도시 왕복하기 2

첫주는 가볍게 끝낼게요~

"가볍게??????????"

고생하셨습니다!!

"나도 고생했어……."