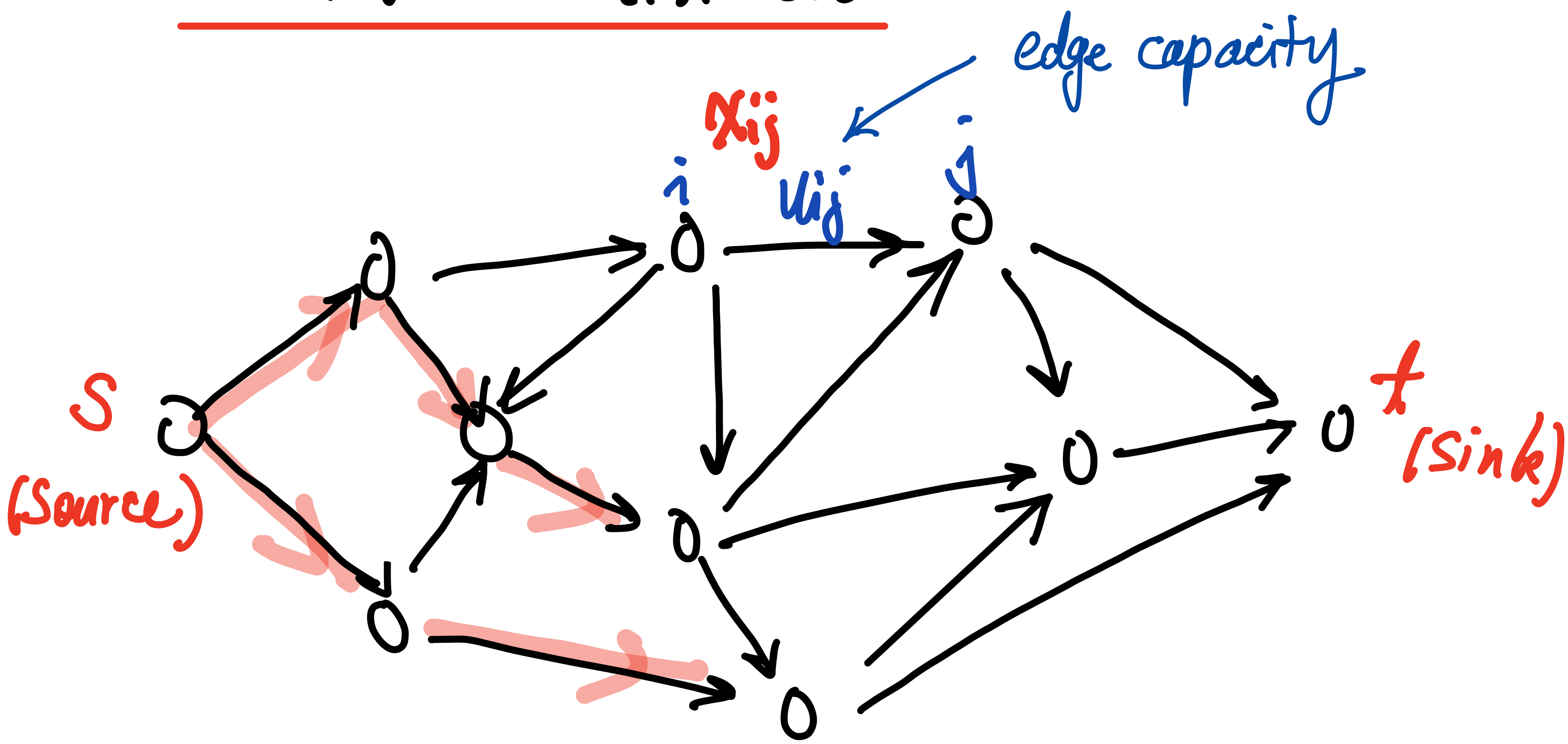
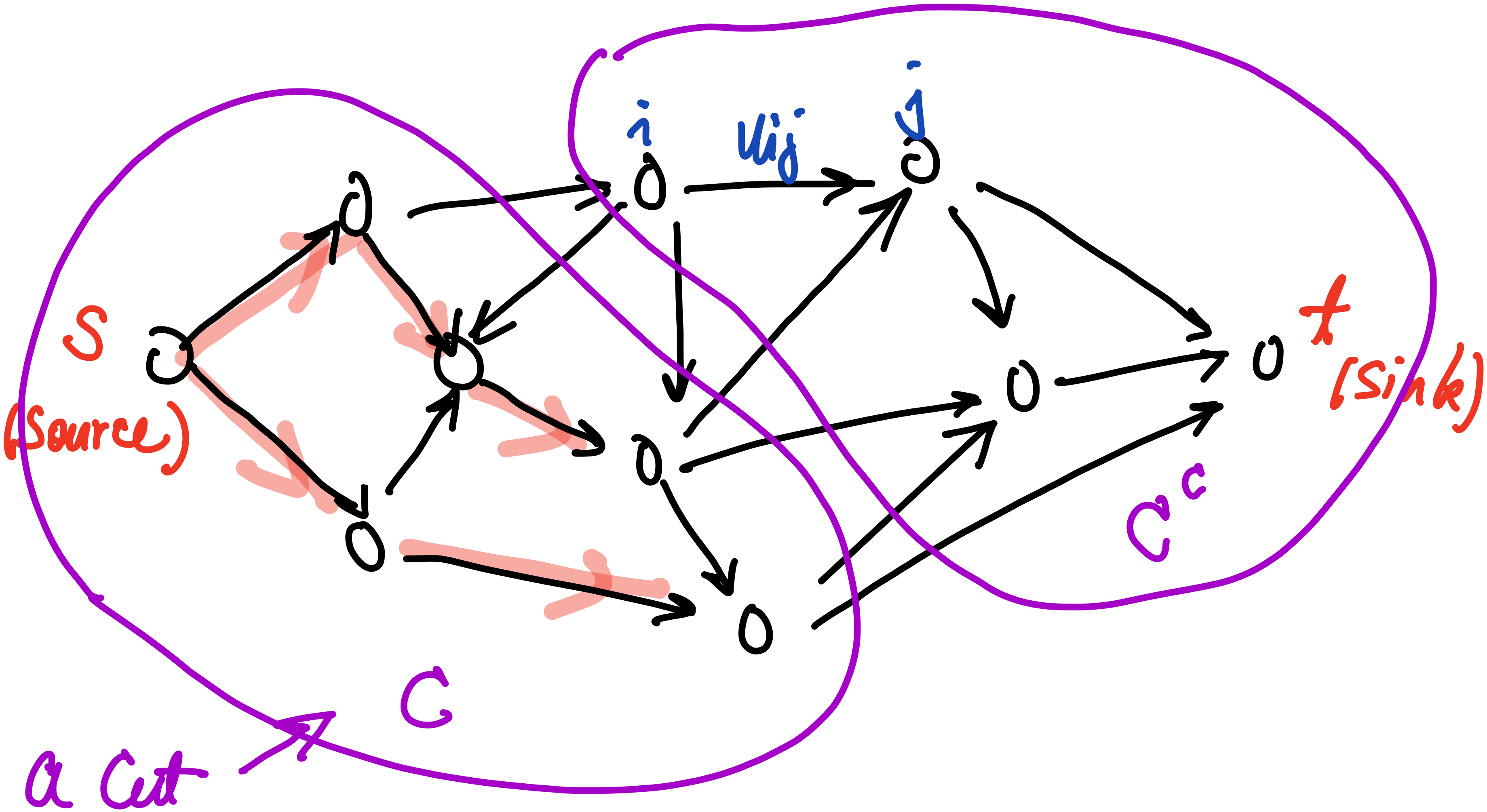
Max Flow Min Cut



What is the max. flow from s to t subject to

$$\boxed{0 \leq x_{ij} \leq u_{ij}}$$?

Max Flow Min Cut

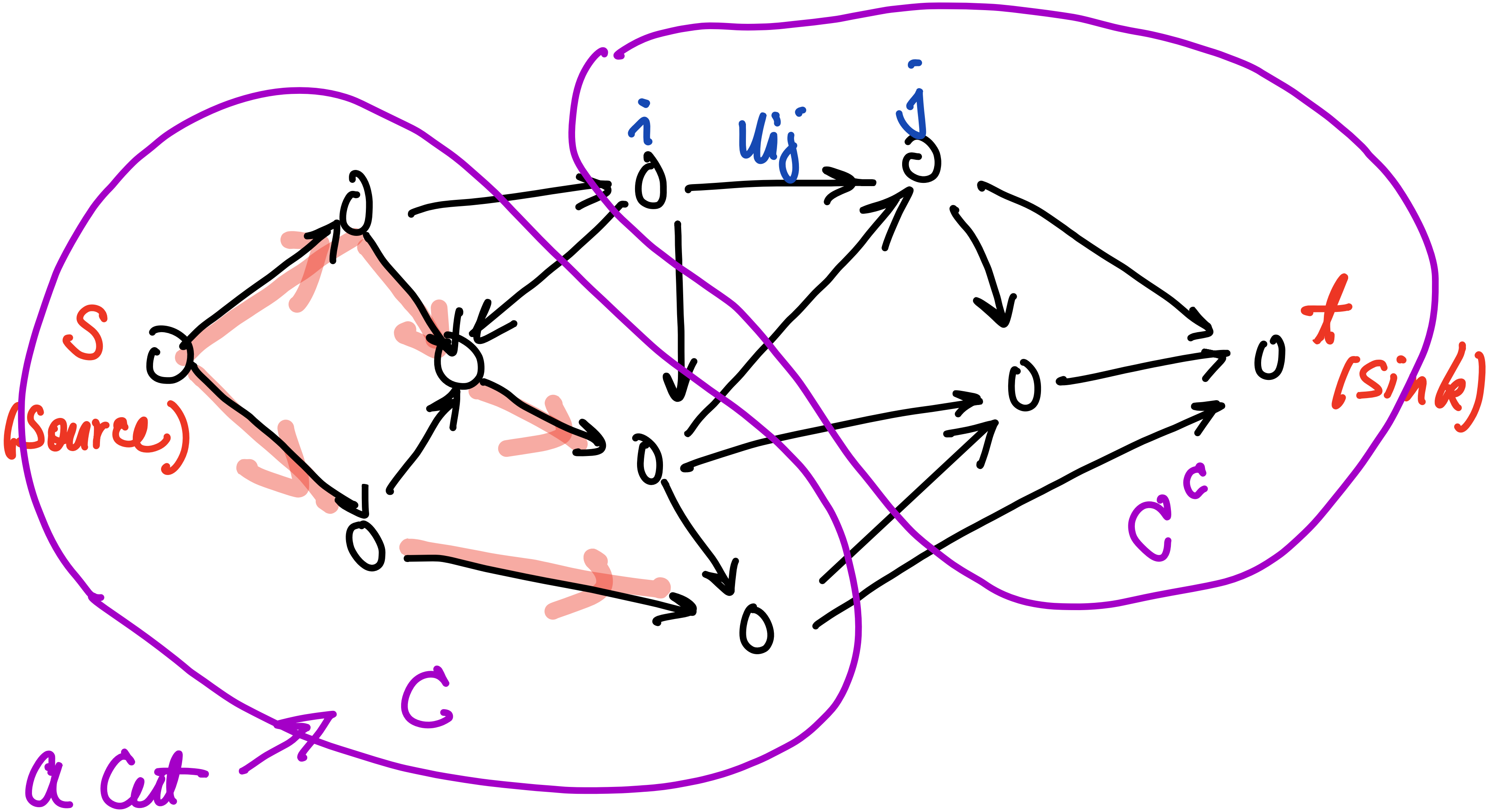


$$N = C \cup C^c, \quad s \in C \quad (t \in C^c)$$

$$\boxed{\text{Flow}(C) = \sum_{i \in C,\, j \notin C} x_{ij}}$$

flow out of the cut C.

Max Flow Min Cut

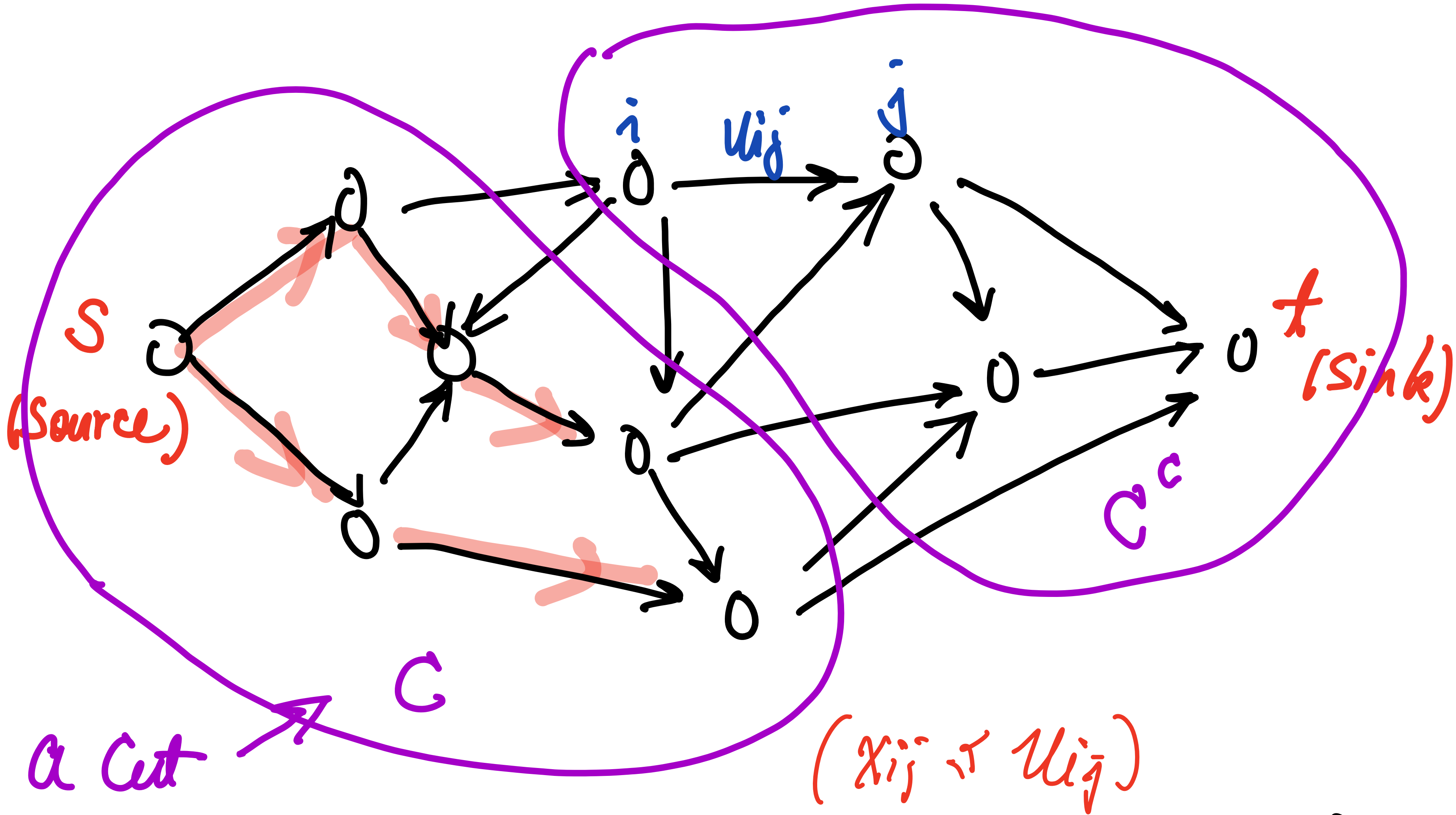


$$N = C \cup C^c, \qquad s \in C \;\; (t \in C^c)$$

$$\boxed{\kappa(C) = \sum_{i \in C,\, j \notin C} u_{ij}}$$

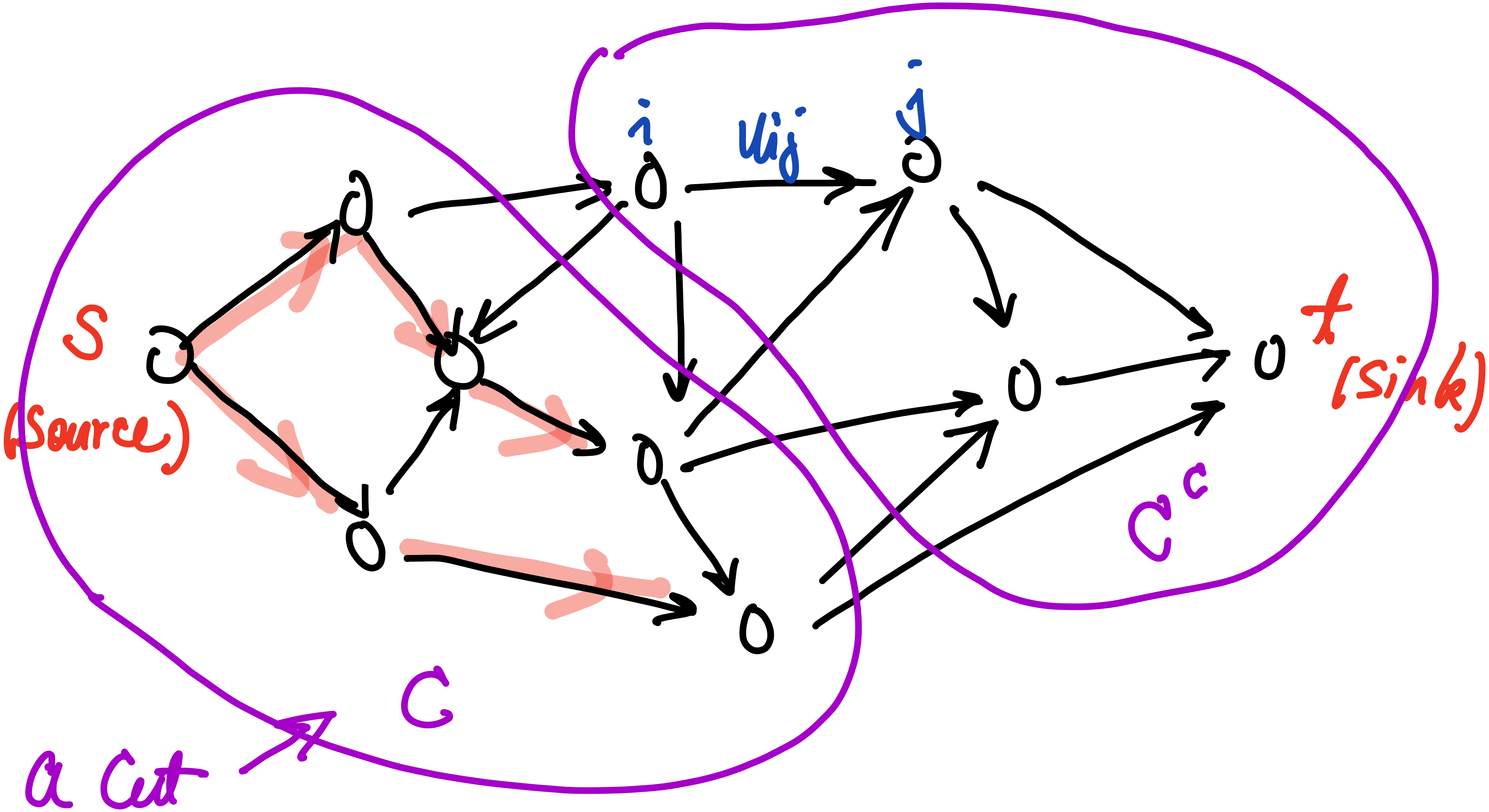
capacity of the cut C.

Max Flow Min Cut



(any) Flow $(s \to t)$ $\leq$ (any) $K(C)$

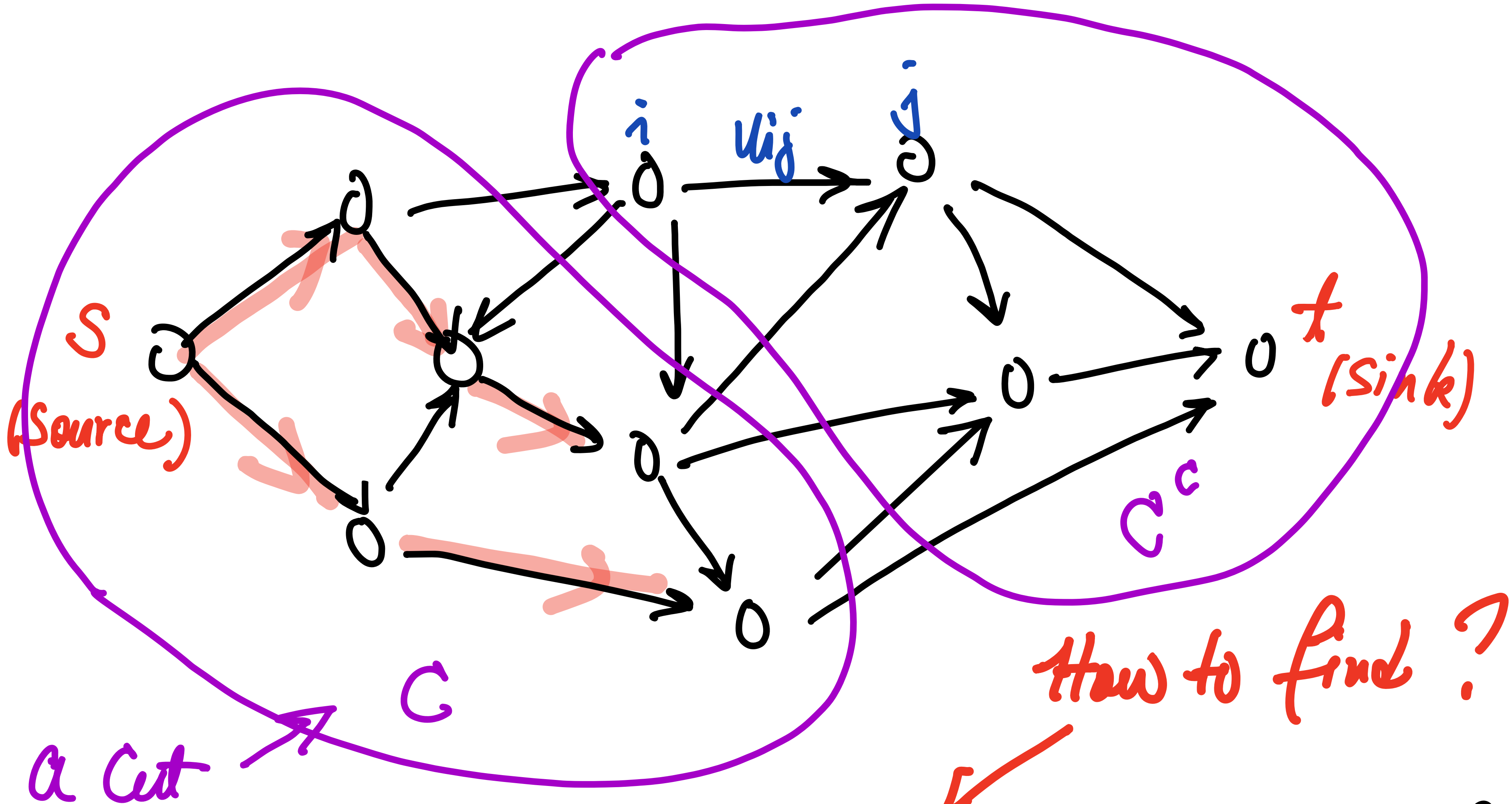
Max Flow Min Cut



$$(\cancel{\text{any}})\ \text{Flow}(s \to t) \le (\cancel{\text{any}})\ K(C)$$

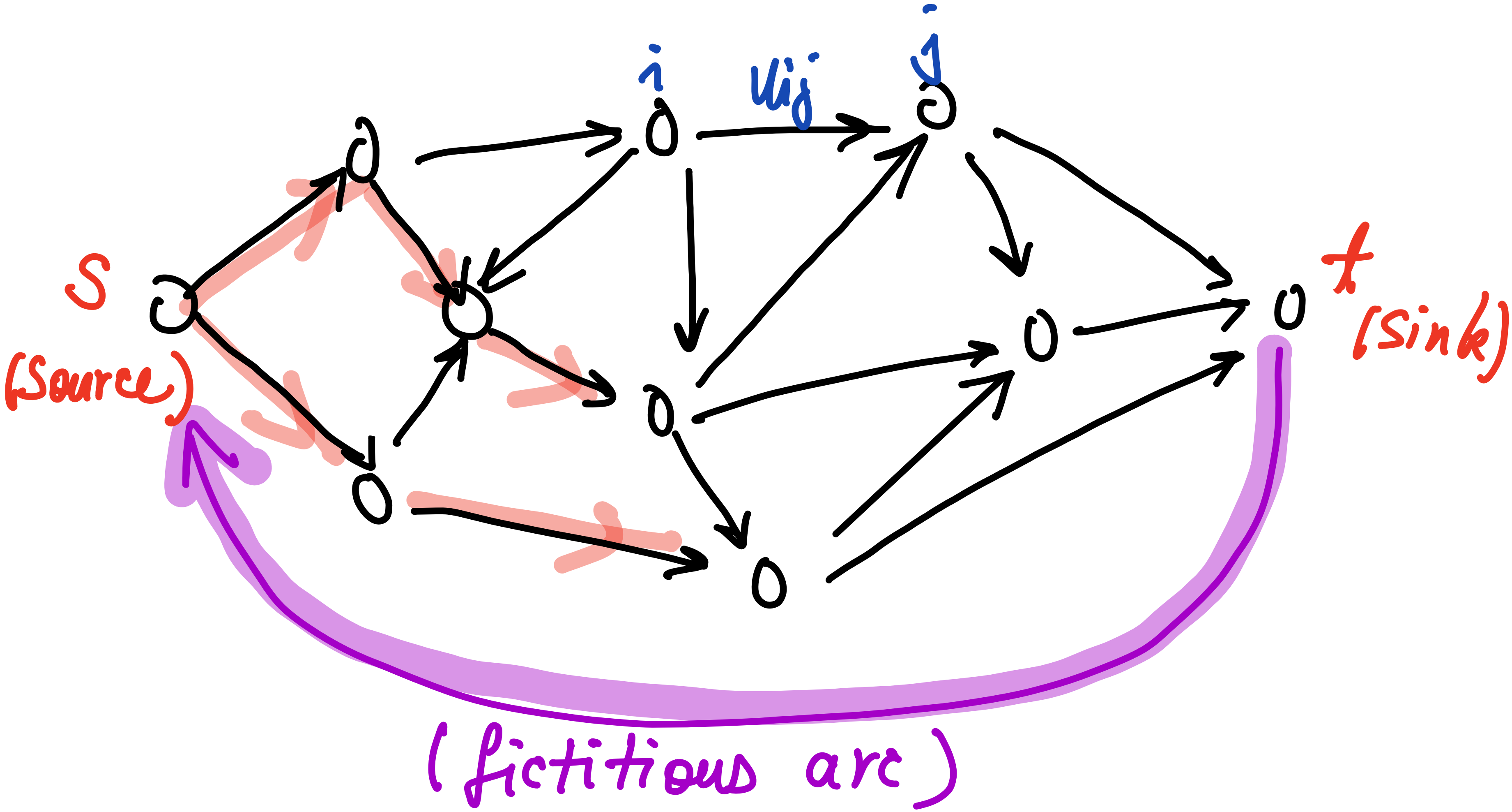
$$\text{max} \qquad\qquad = \qquad \text{min}$$

Max Flow Min Cut



In fact, a flow is max. if and only if it corresponds to a min. cut.

Max Flow Min Cut (LP Formulation)



x_{ts}, $\quad c_{ts} = -1$, $\quad u_{ts} = +\infty$

Max Flow Min Cut (LP Formulation)

$$\min \quad -x_{ts}$$ max flow

s.t.

$$x_{ts} = \sum_j x_{sj}$$ out of source

$$\sum_i x_{it} = x_{ts}$$ into sink

$$\sum_i x_{ik} = \sum_j x_{kj} \quad k \neq s,t$$ in flow = out flow at k

$$0 \leq x_{ij} \leq u_{ij}$$ capacity

$$(\;u_{ts} = +\infty\;)$$

[C] p.368

Table 22.1 The Travelers' Example: Seat Availability

From	To	Number of seats
San Francisco	Denver	5
San Francisco	Houston	6
Denver	Atlanta	4
Denver	Chicago	2
Houston	Atlanta	5
Atlanta	New York	7
Chicago	New York	4

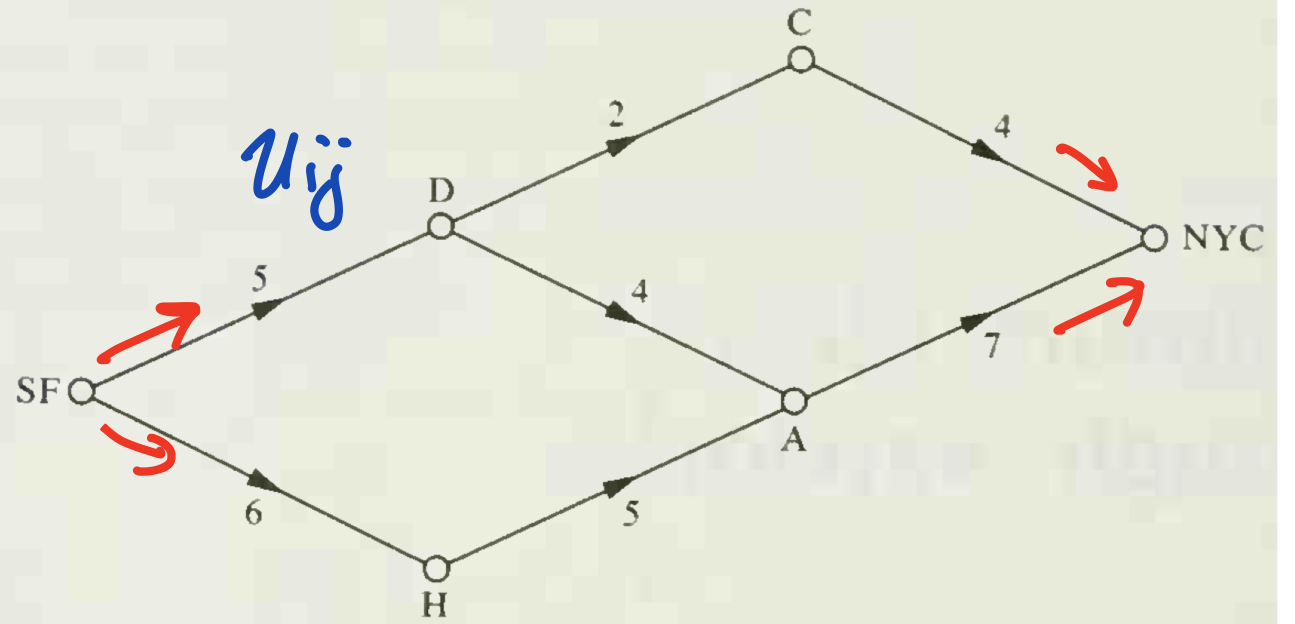


Figure 22.1

[C] p.368

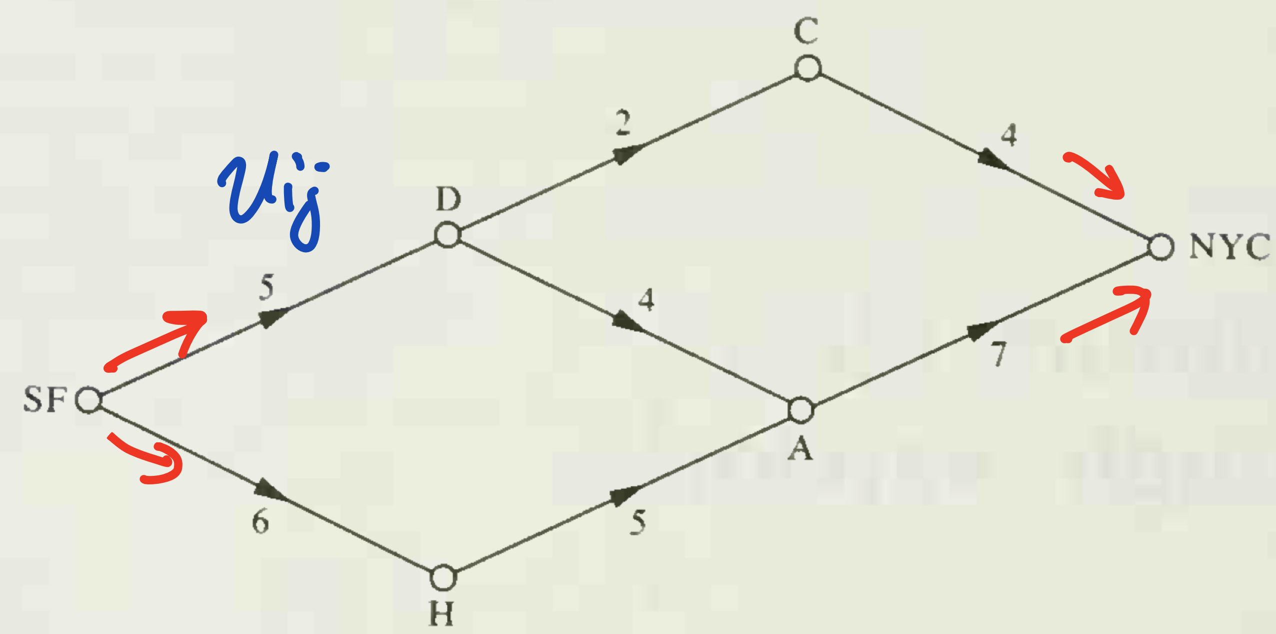


Figure 22.1

Capacities

Flow = 8

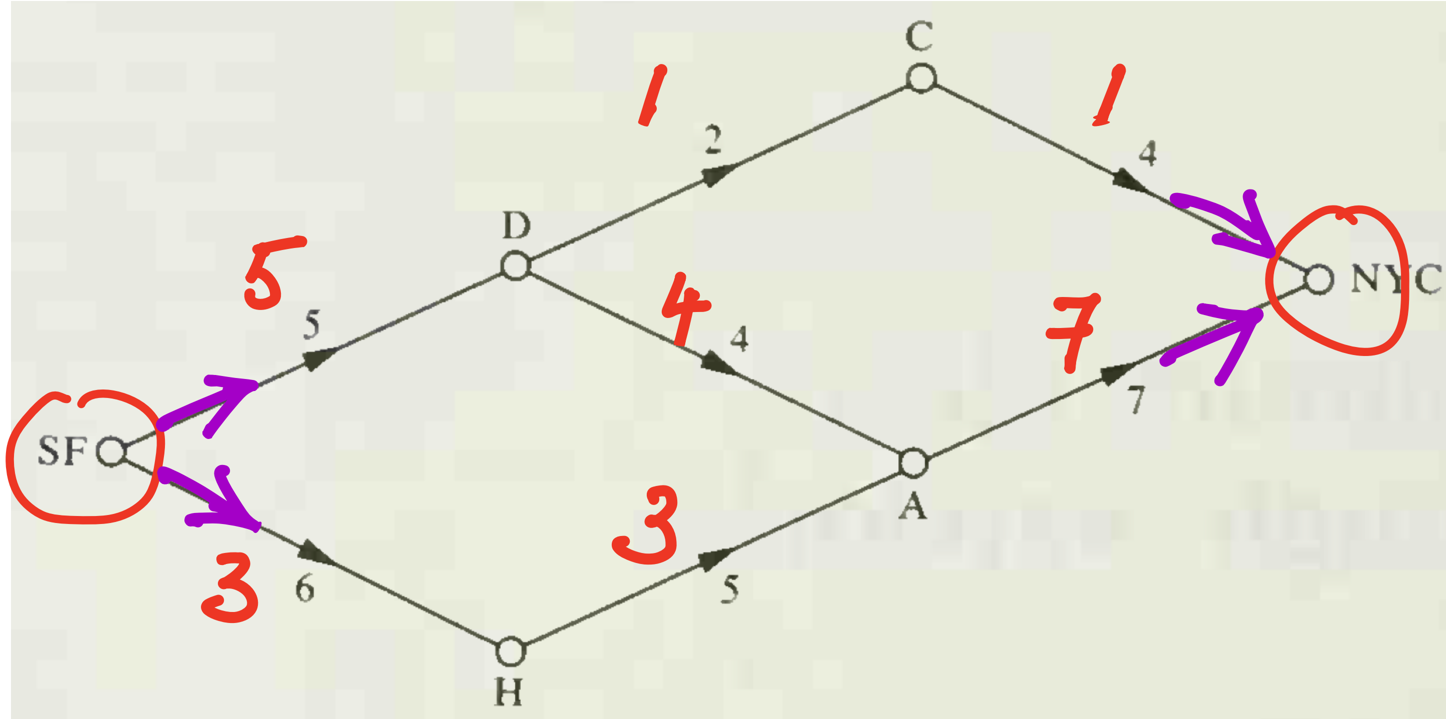


Figure 22.1

[C] p.368

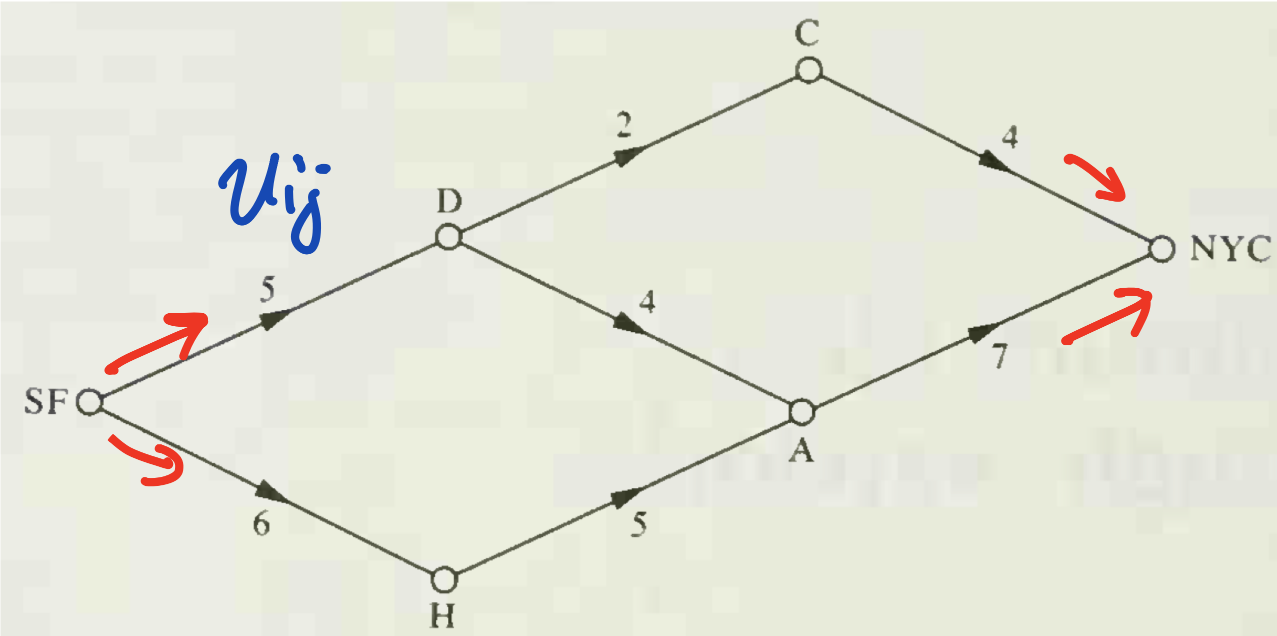


Figure 22.1

u_{ij}

Capacities

Flow = 9

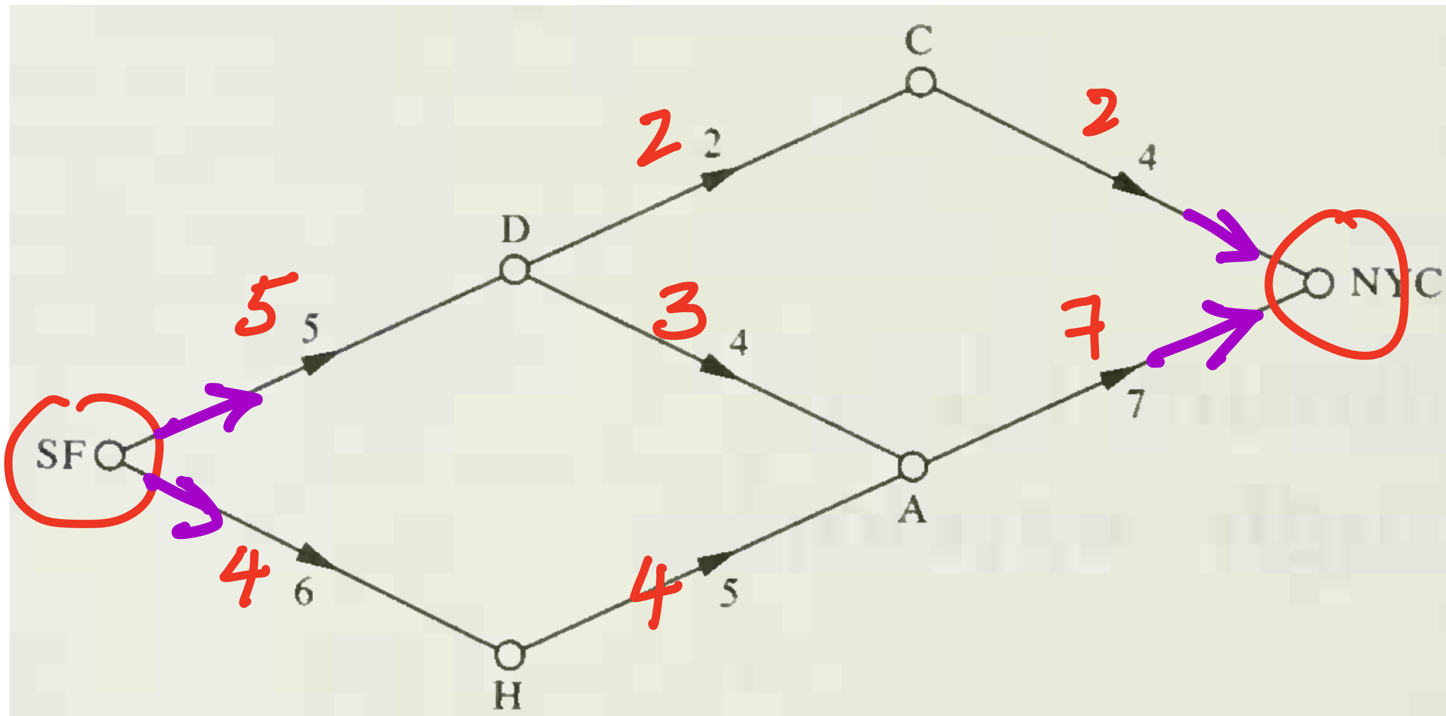


Figure 22.1

[C] p.368

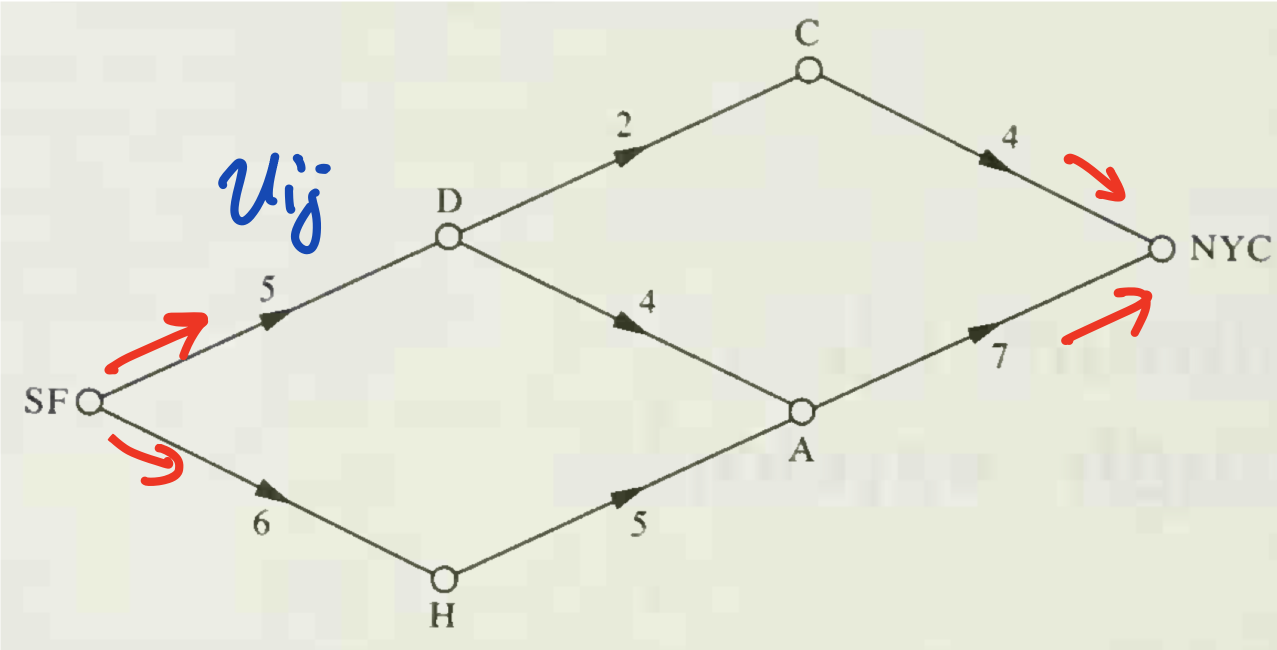


Figure 22.1

Capacities

Flow = 9

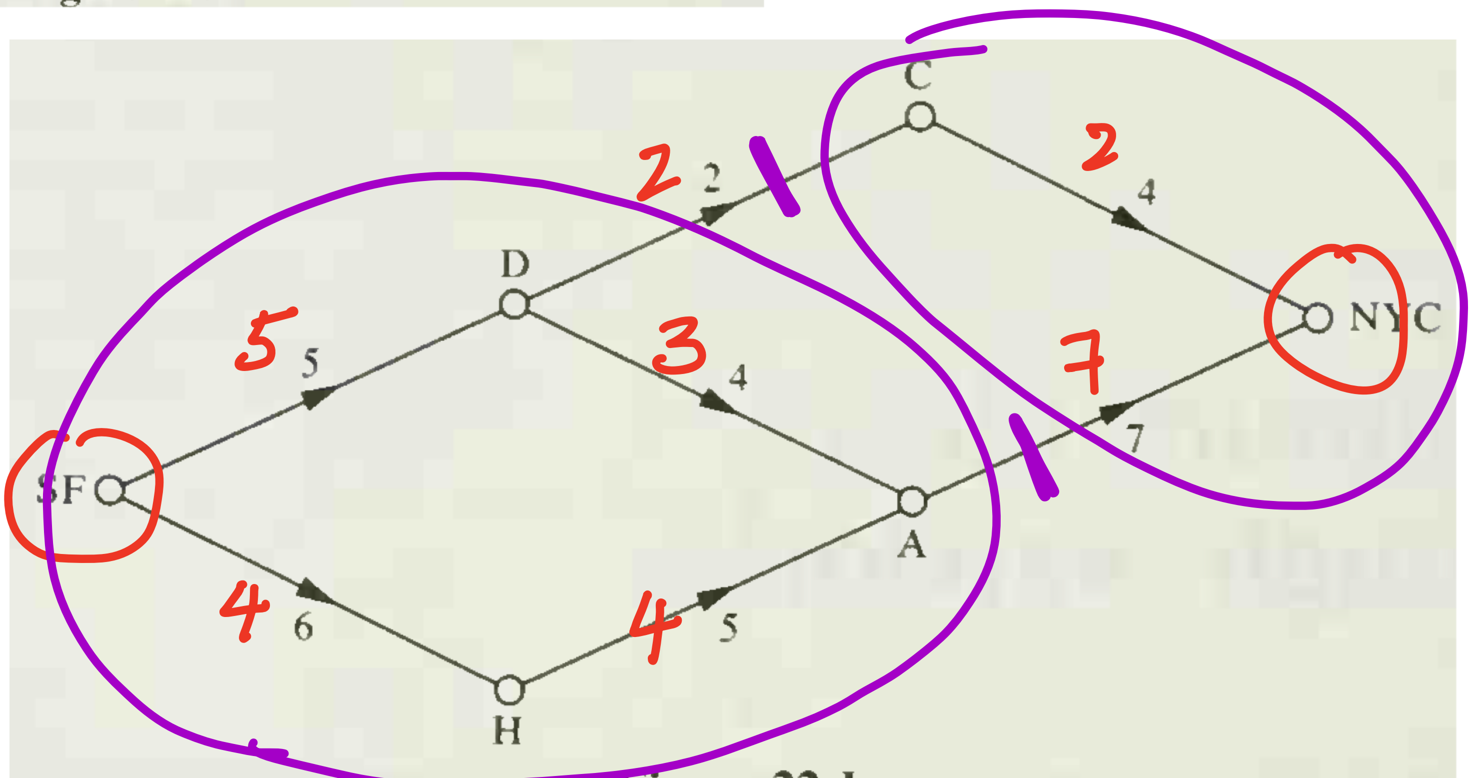


Figure 22.1

Proof of Max Flow = Min Cut

- Let $\{x^*_{ij}\}$ be a (optimal) max flow
- Let C^* be those nodes that some more flow can be pushed from s.
- $t \notin C^*$ (By default, $s \in C^*$)

Claim

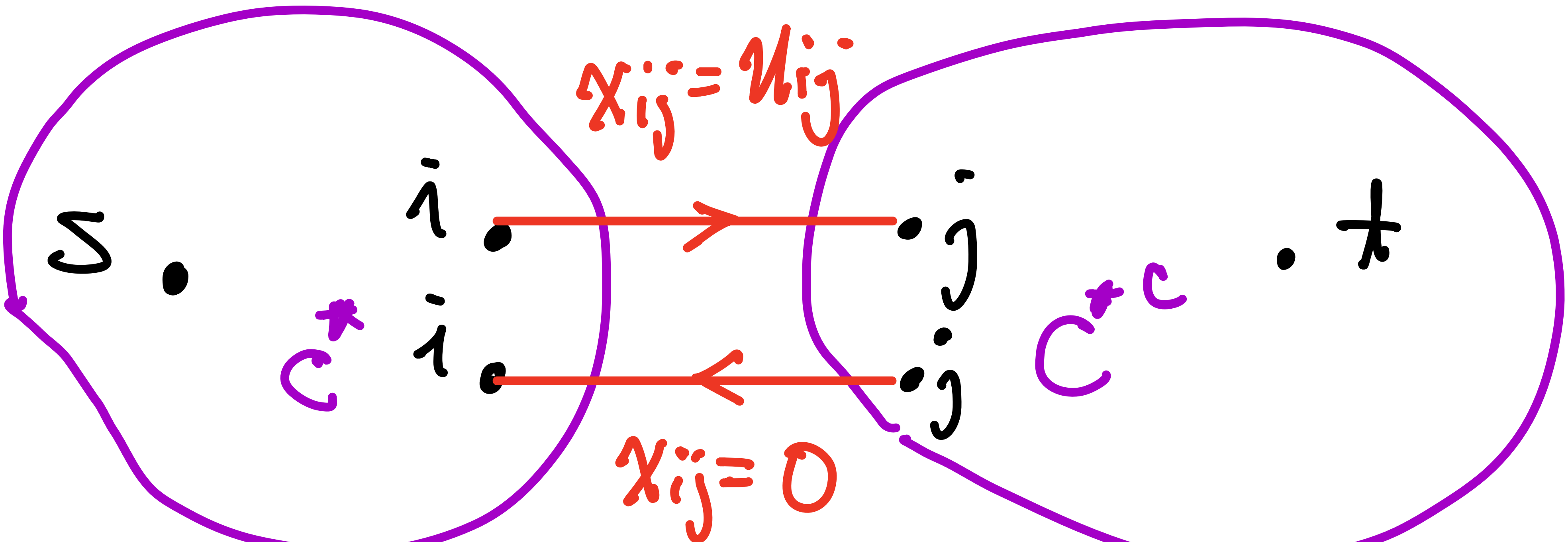
$$\boxed{\begin{gathered} \text{Flow}(C^* \text{ to } C^{*c}) = K(C^*) \\ \sum_{i \in C^*, j \notin C^*} x^*_{ij} = \sum_{i \in C^*, j \notin C^*} u_{ij} \end{gathered}}$$

Proof of Max Flow = Min Cut

Claim

$$\text{Flow}(C^* \text{ to } C^{*c}) = K(C^*)$$

$$\sum_{i \in C^*,\, j \notin C^*} x_{ij}^* = \sum_{i \in C^*,\, j \notin C^*} u_{ij}$$



Ford-Fulkerson Algorithm (Augmented Path)

Flow and Capacities

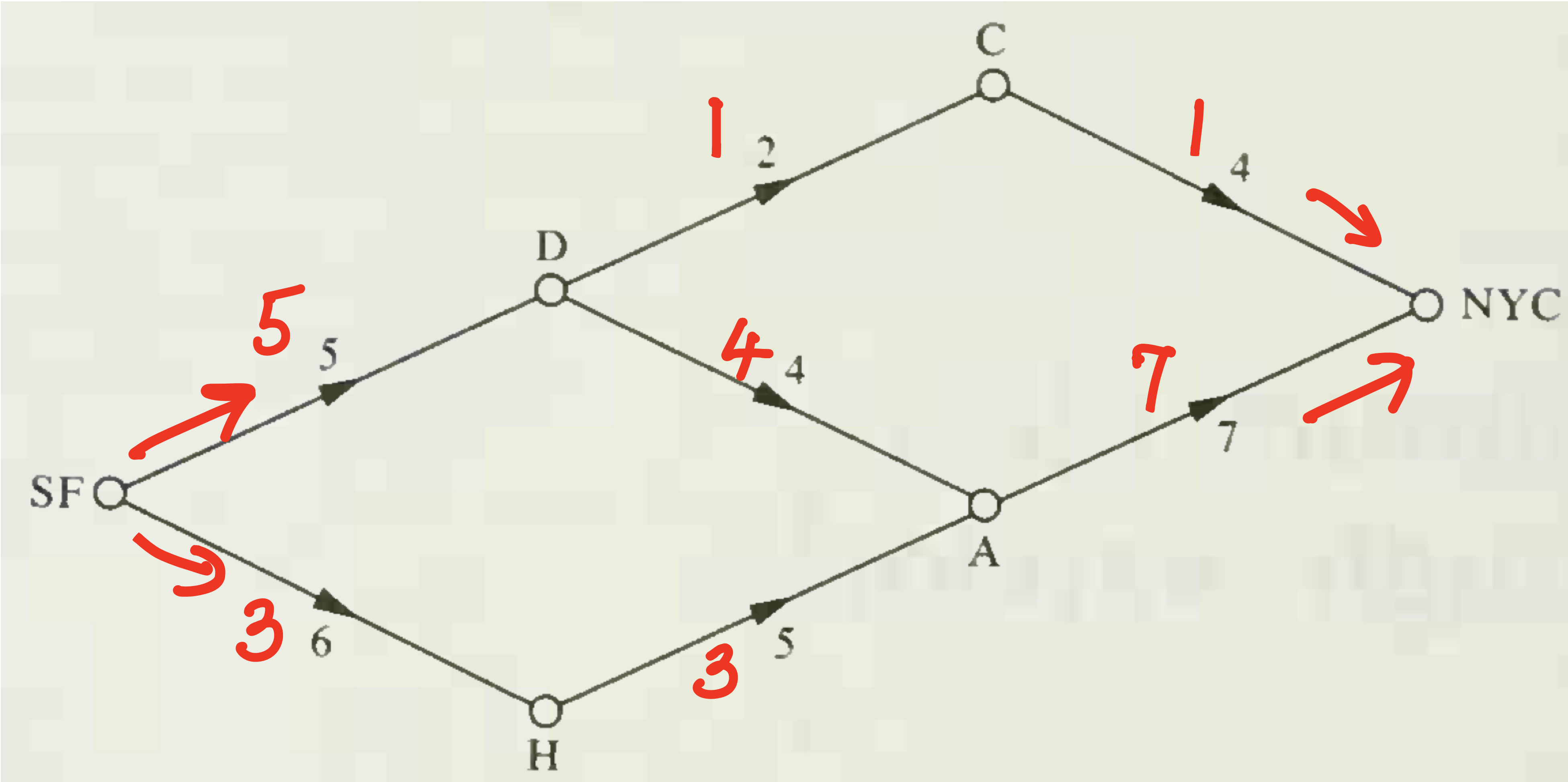


Figure 22.1

Ford- Fulkerson Algorithm (Augmented Path)

Flow and Capacities

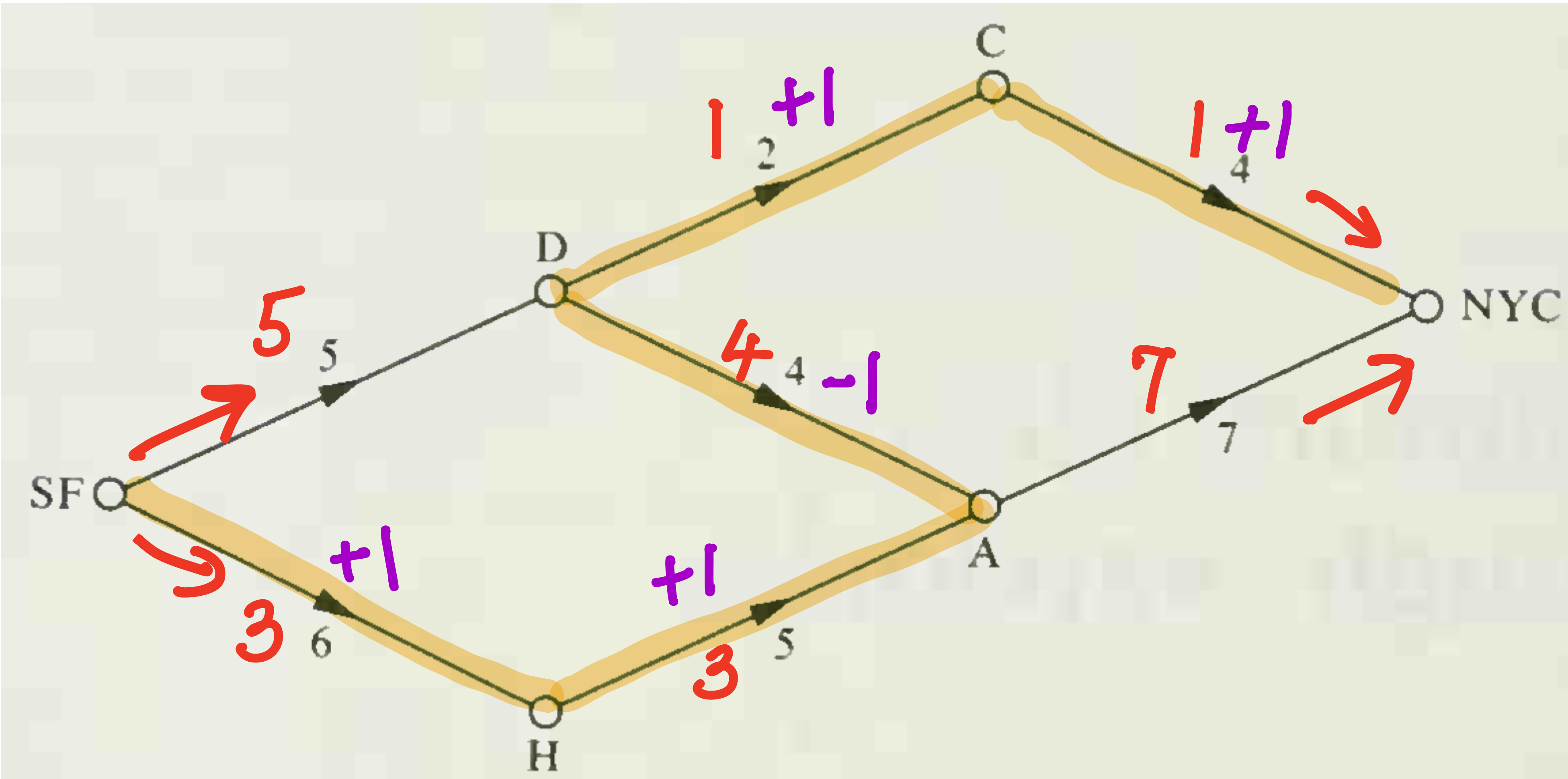


Figure 22.1

Ford-Fulkerson Algorithm (Augmented Path)

Flow and Capacities

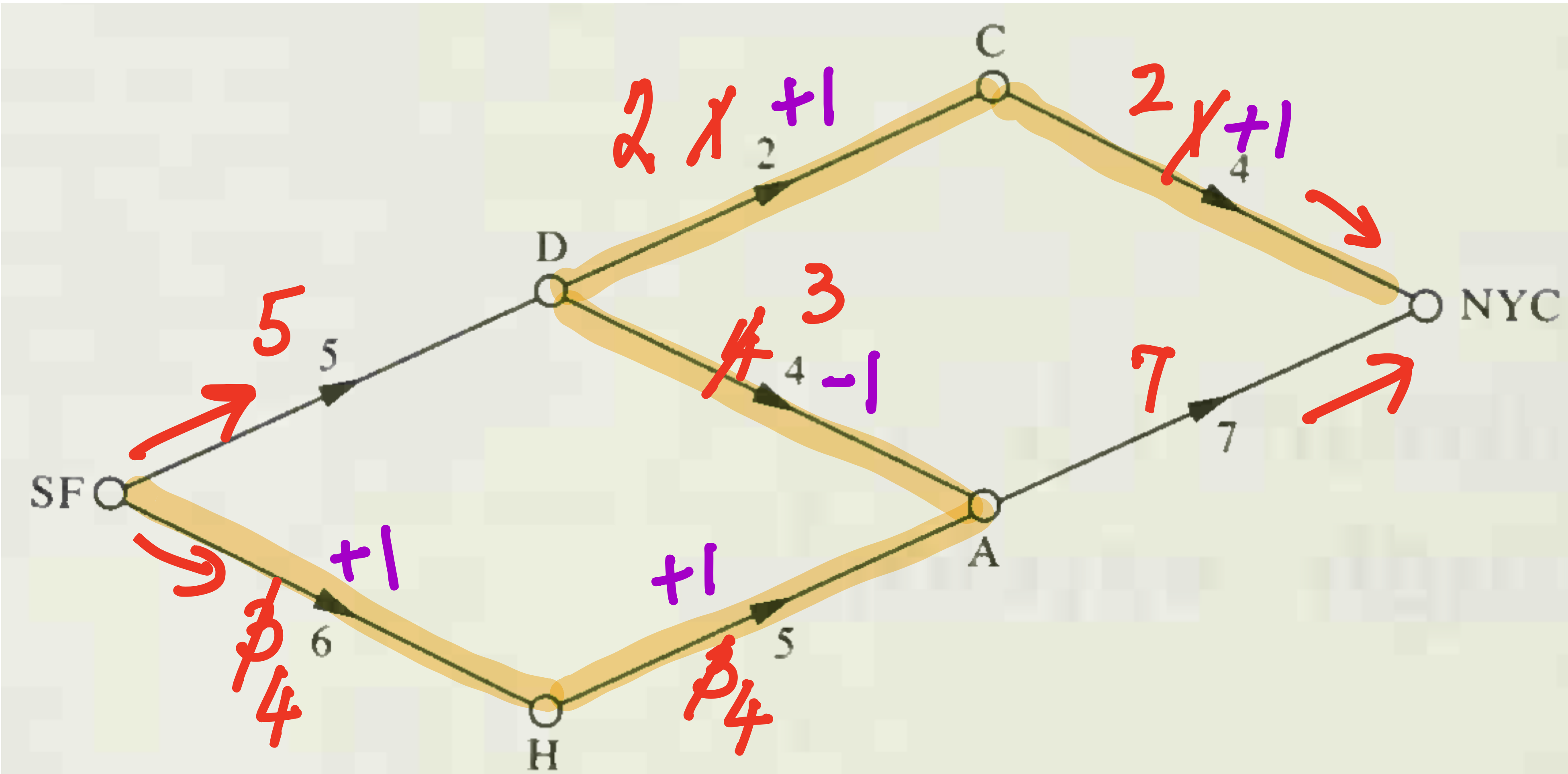


Figure 22.1

Ford-Fulkerson Algorithm (Augmented Path)

Flow and Capacities

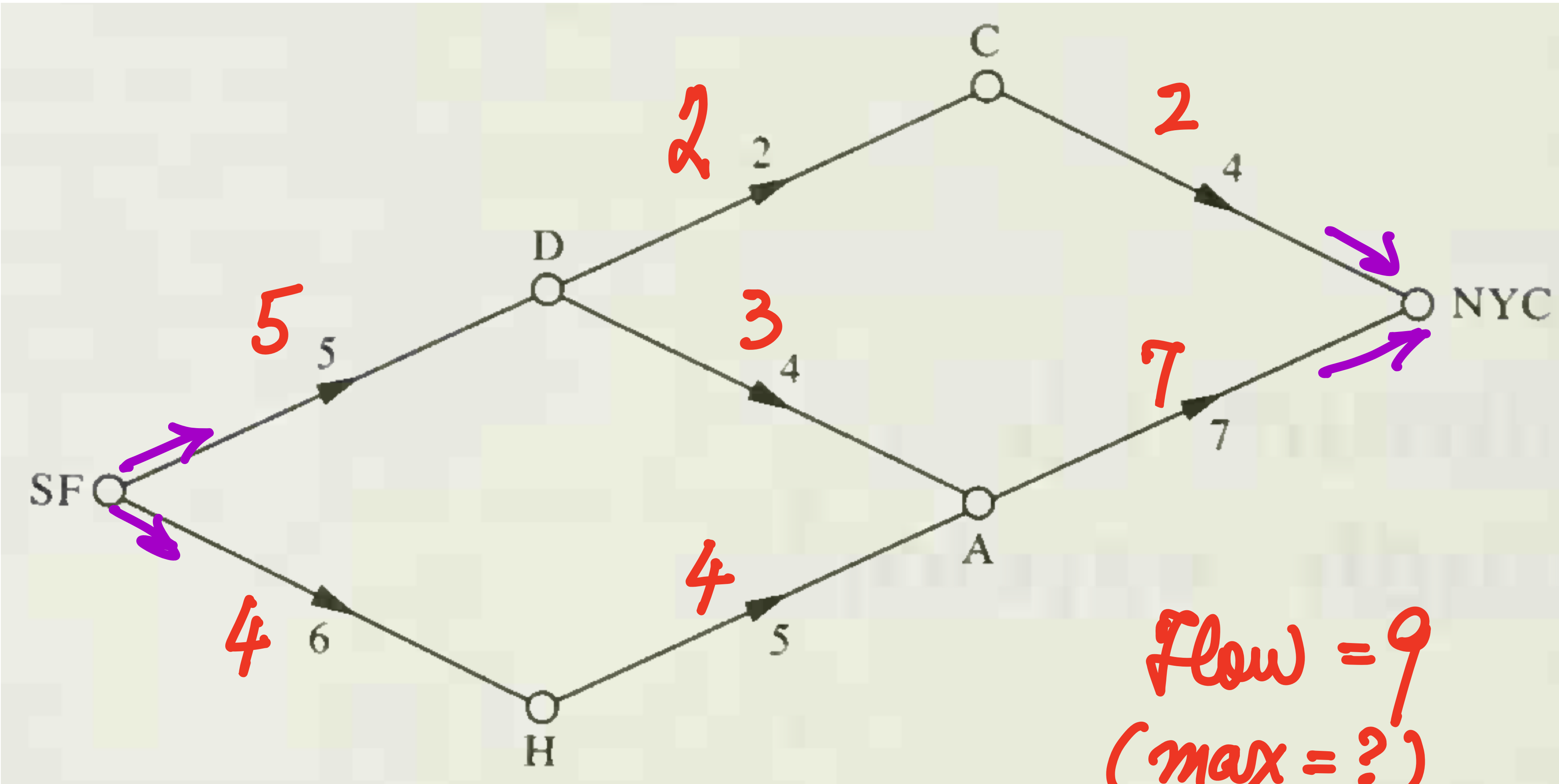
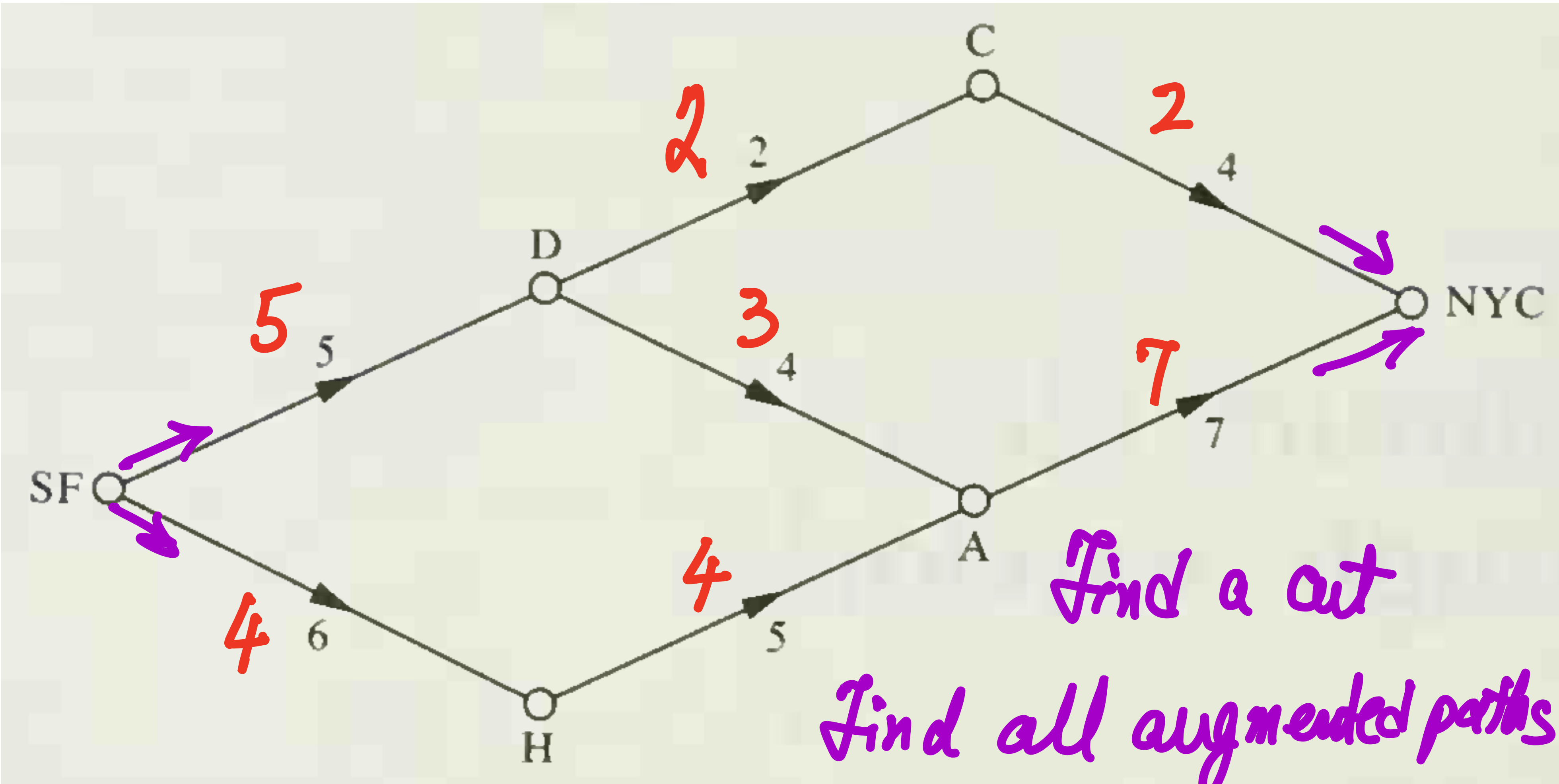


Figure 22.1

Ford-Fulkerson Algorithm (Augmented Path)

Flow and Capacities



Find a cut

Find all augmented paths

Figure 22.1

Ford-Fulkerson Algorithm (Augmented Path)

Flow and Capacities

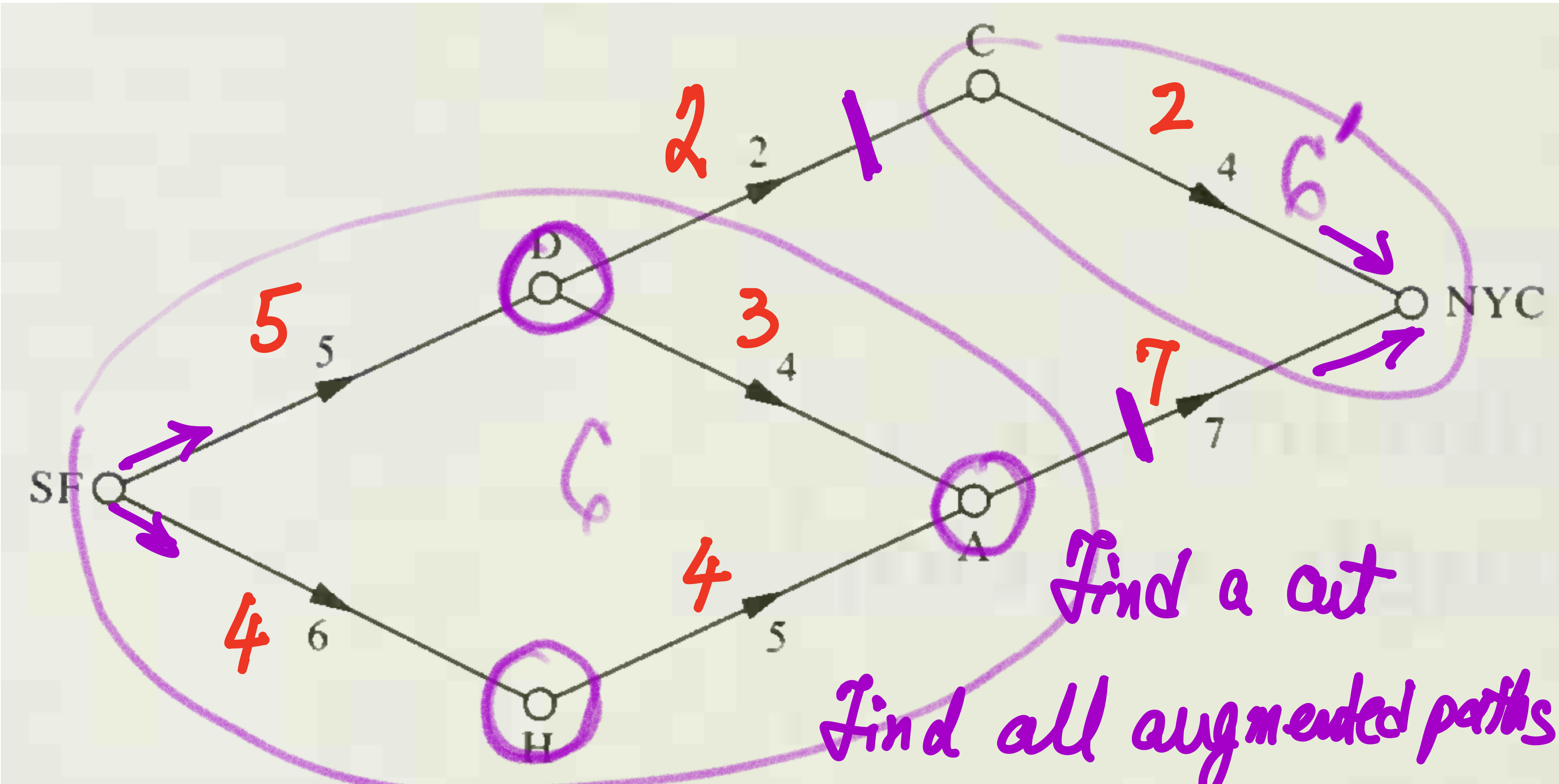


Figure 22.1

Ford- Fulkerson Algorithm (Augmented Path)

Implementations

Our description of the augmenting path method does not specify a way of searching for the arcs ij such that $i \in C$, $j \notin C$, $x_{ij} < u_{ij}$, and the arcs ji such that $j \notin C$, $i \in C$, $x_{ji} > 0$. Ford and Fulkerson did specify a way of doing that. In their terminology, nodes in C are called *labeled* and nodes outside C are called *unlabeled*; the labeled nodes are divided further into *scanned* and *unscanned*. Initially, the source s is labeled but unscanned and all the remaining nodes are unlabeled. Scanning a labeled node i means *examining all the arcs* ij *and, whenever such an arc satisfies* $x_{ij} < u_{ij}$, $j \notin C$, *setting* $j \in C$, $p(j) = ij$ and *examining all the arcs* ji *and, whenever such an arc satisfies* $x_{ji} > 0$, $j \notin C$, *setting* $j \in C$, $p(j) = ji$. The search may be described as in Box 22.1.

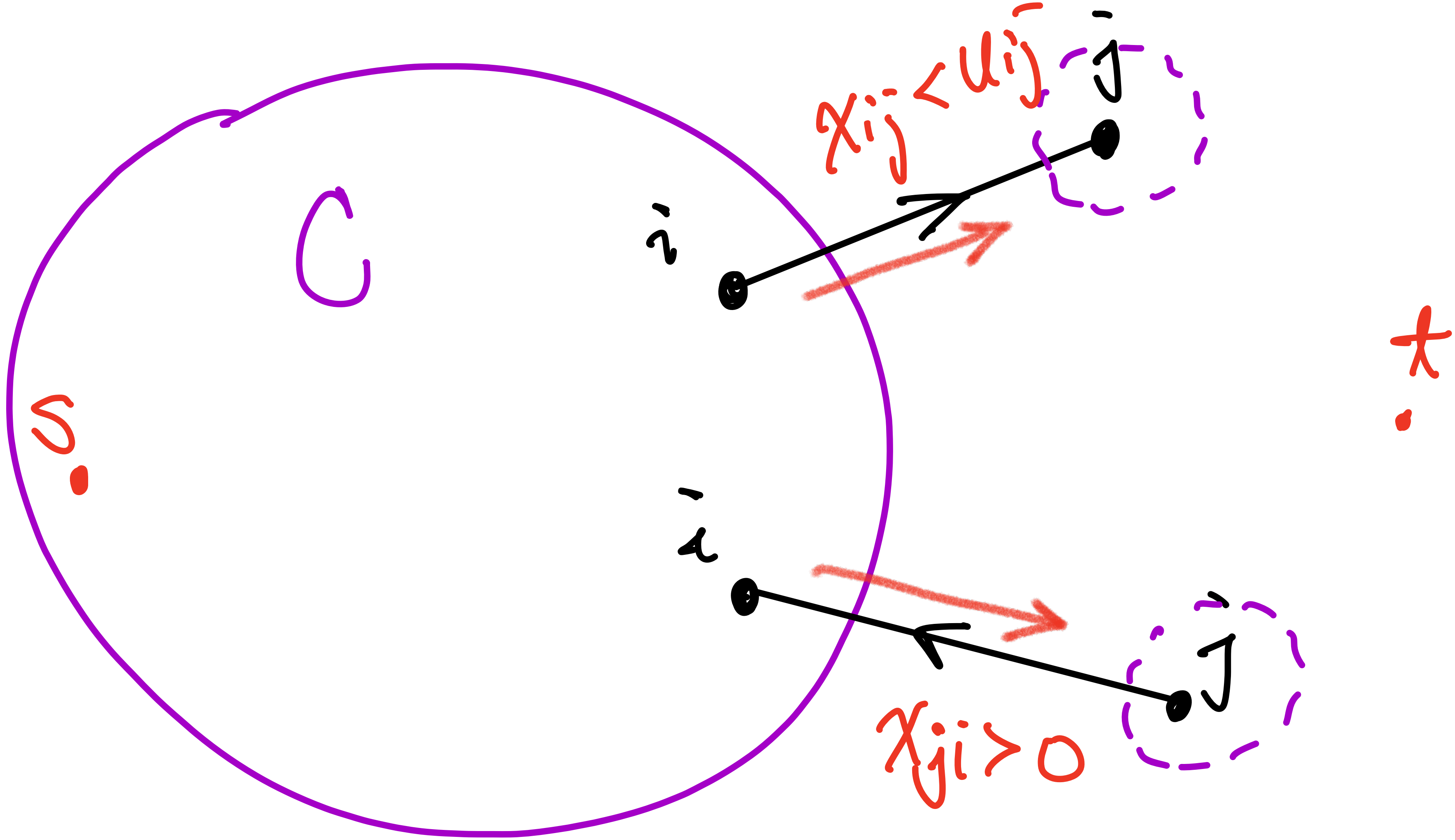
BOX 22.1 Search for an Augmenting Path

Step 0. Mark s as labeled unscanned; mark the remaining nodes as unlabeled.

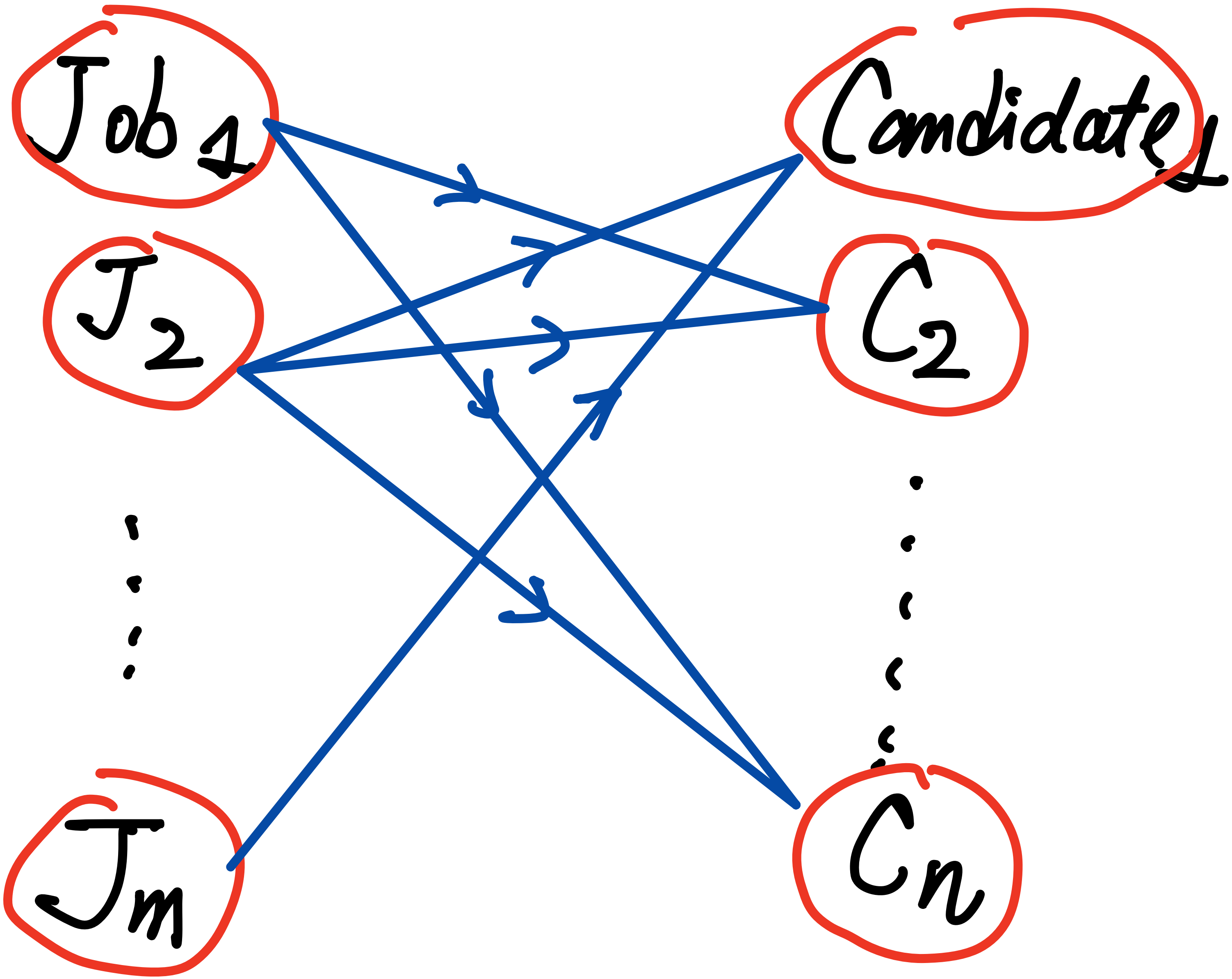
Step 1. If all the labeled nodes are scanned then stop [the set C of labeled nodes satisfies (22.7) and (22.8)]; otherwise, choose a labeled unscanned node i.

Step 2 Scan i. If t has become labeled then stop (an x-augmenting path has been found); otherwise return to step 1.

Ford-Fulkerson Algorithm (Augmented Path)



Max. Matching as a Max. Flow

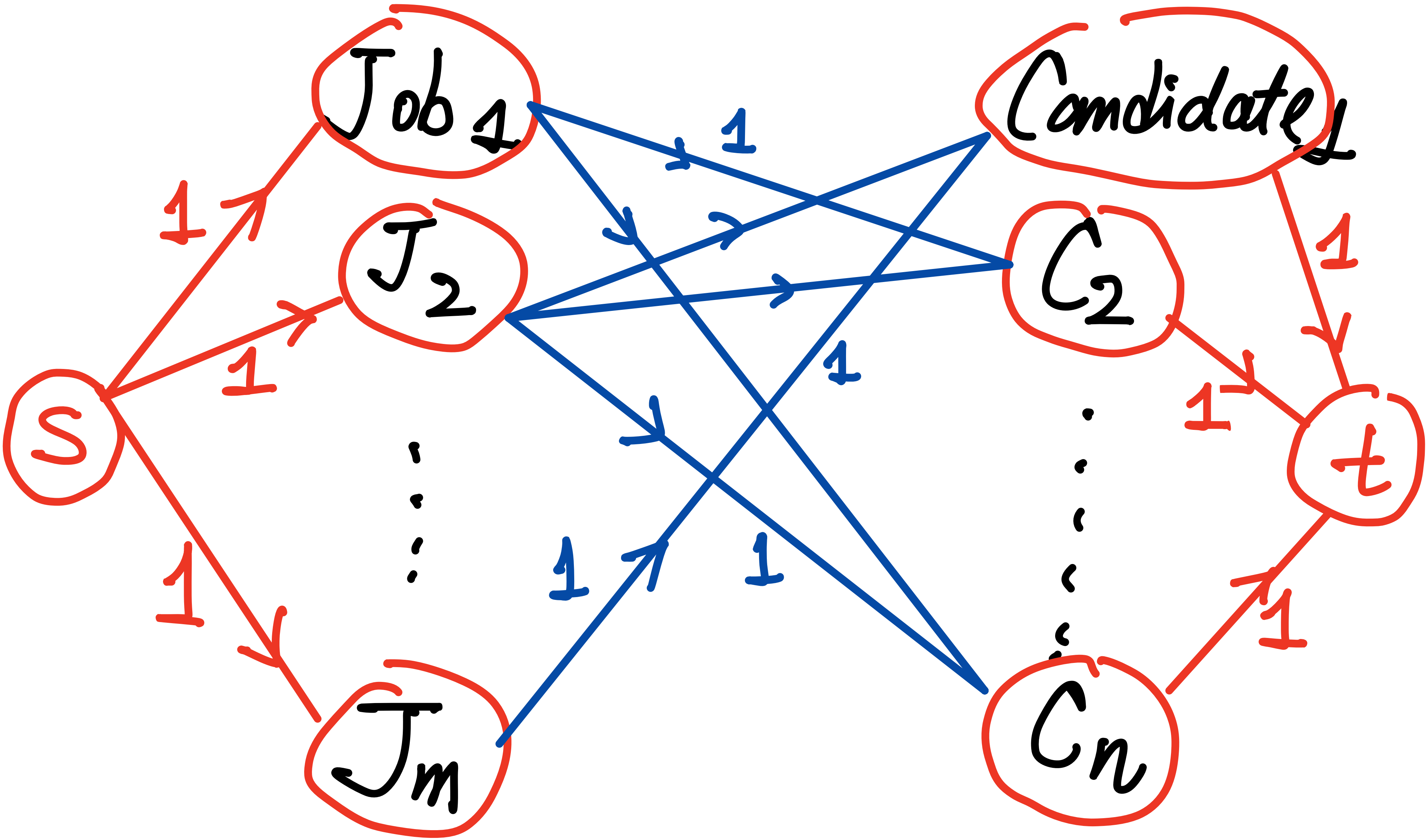


Max. Matching as a Max. Flow

- Assign J_i to C_j only if there is an edge connecting them.
- Each job is assigned to one candidate and vice versa.

Q: Find the max. number of assignment.

Max. Matching as a Max. Flow



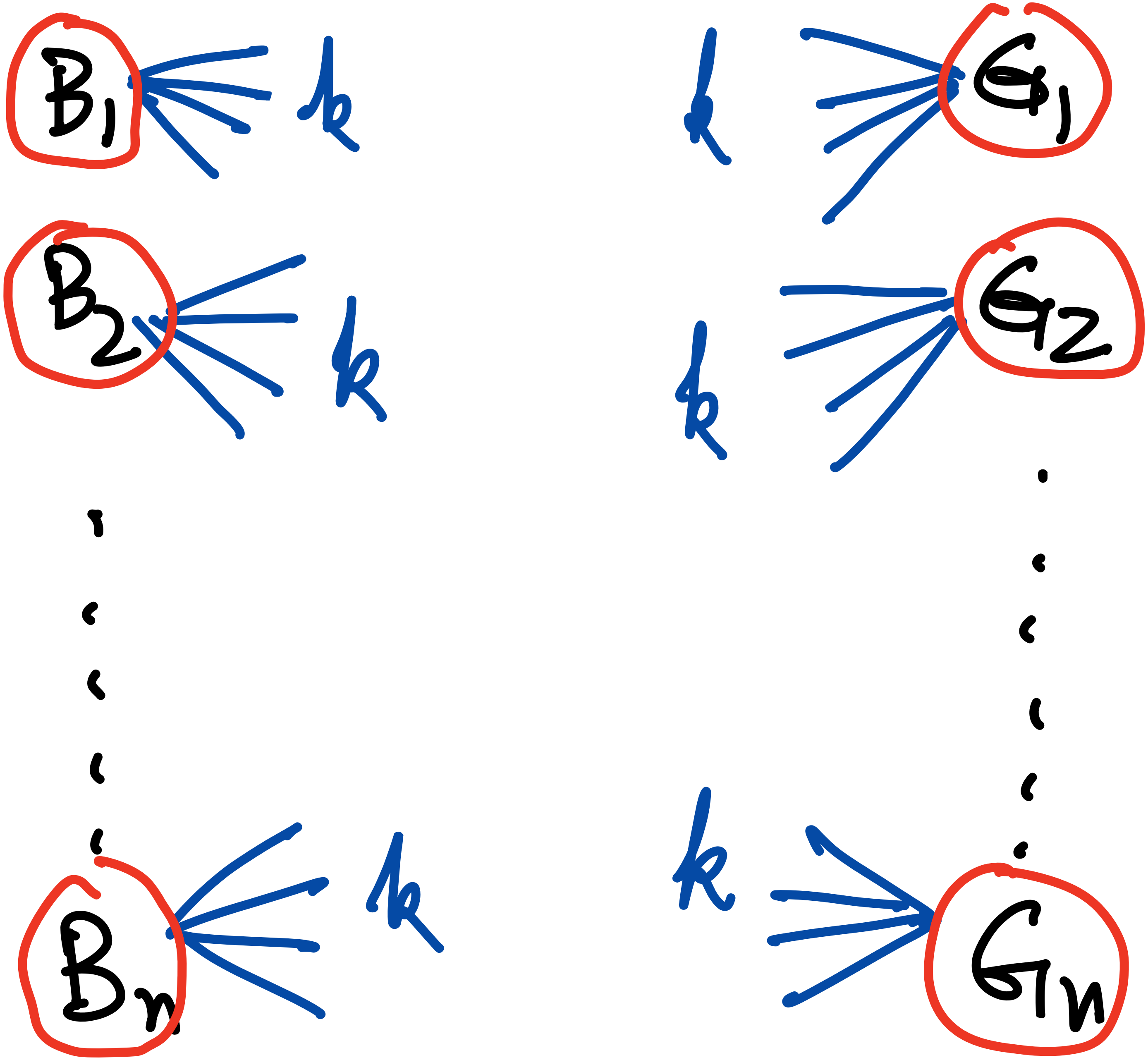
[V, p.248]

6.1. König's Theorem. In addition to its importance in real-world optimization problems, the integrality theorem also has many applications to the branch of mathematics called combinatorics. We illustrate with just one example.

THEOREM 14.3. *König's Theorem. Suppose that there are n girls and n boys, that every girl knows exactly k boys, and that every boy knows exactly k girls. Then n marriages can be arranged with everybody knowing his or her spouse.*

Before proving this theorem it is important to clarify its statement by saying that the property of "knowing" is symmetric (for example, knowing in the biblical sense). That is, if a certain girl knows a certain boy, then this boy also knows this girl.

PROOF. Consider a network with nodes $g_1, g_2, \ldots, g_n, b_1, b_2, \ldots, b_n$ and an arc from g_i to b_j if girl i and boy j know each other. Assign one unit of supply to each girl node and a unit of demand to each boy node. Assign arbitrary objective coefficients to create a well-defined network flow problem. The problem is guaranteed to be feasible: just put a flow of $1/k$ on each arc (the polygamists in the group might prefer this nonintegral solution). By the integrality theorem, the problem has an integer-valued solution. Clearly, the flow on each arc must be either zero or one. Also, each girl node is the tail of exactly one arc having a flow of one. This arc points to her intended mate. $\square$



B1
k
B2
k
Bn
k
k
G1
k
G2
k
Gn