

Constructions in AlgoLand

Xun Zou

Slides adapted from Hafsteinn Einarsson
December 5, 2018

Exercise 1

Exercise – *Connecting Cities*

As a first step toward building a reliable infrastructure, the people of AlgoLand built a skeleton of the road network. The skeleton of the road network is just a list of possible roads, some of which will be actually built. Moreover, the list of possible roads is such that between every two cities in AlgoLand there exists exactly one path connecting them. You want to start building as many roads as possible at the same time.

There is a constraint, though: to avoid too much disruption you do not want to build two roads ending in the same city.

Goal: Find a largest set of **vertex disjoint edges**
or find a largest matching

Matching

Goal

Find a **largest matching**

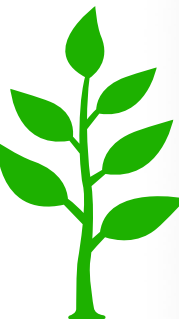
Maximum Cardinality Matching: $O(VE)$

Points There are two test sets:

1. For the first set, worth 30 points, you may assume that the total number of cities is at most 500.
2. For the second set, worth 70 points, the total number of cities is at most 10^6 .

Do we know more about the graph?

Exercise – *Connecting Cities*

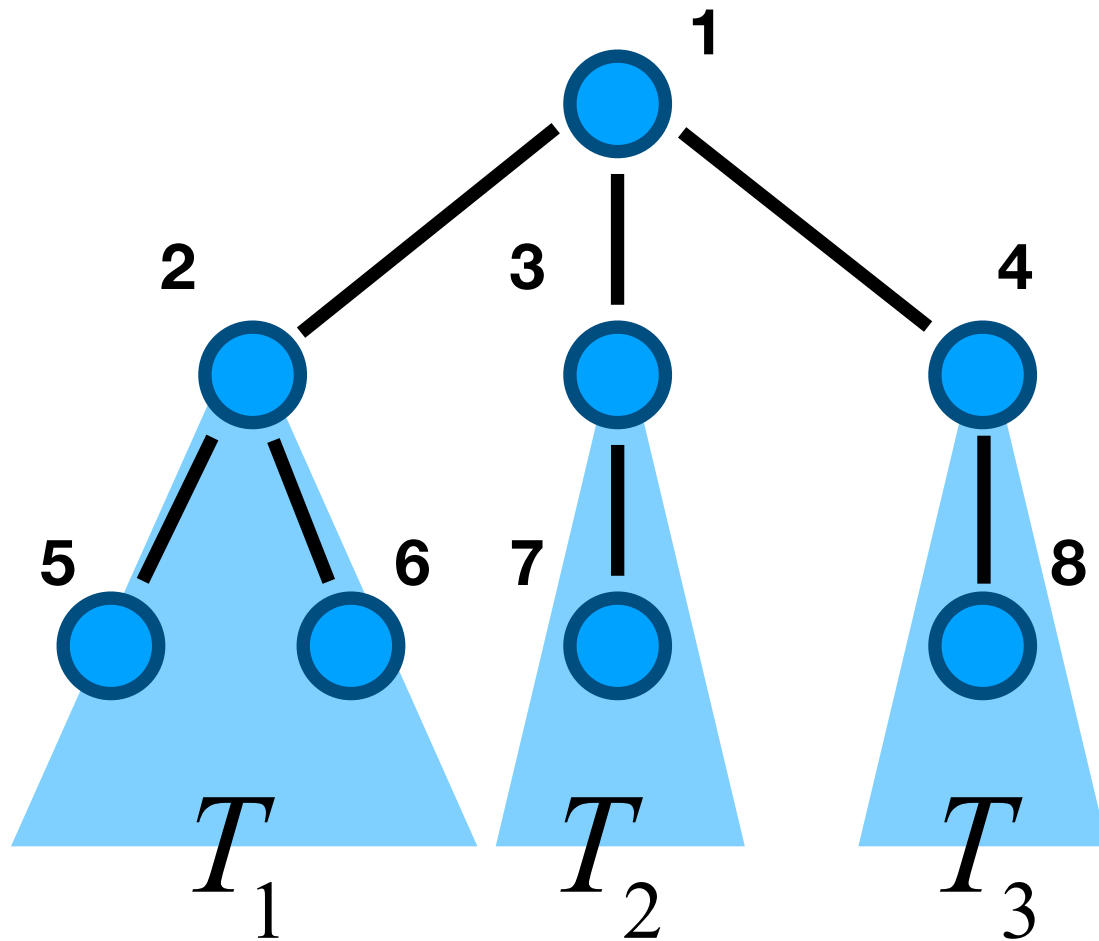


As a first step toward building a reliable infrastructure, the people of AlgoLand built a skeleton of the road network. The skeleton of the road network is just a list of possible roads, some of which will be actually built. Moreover, the list of possible roads is such that between every two cities in AlgoLand there exists exactly one path connecting them. You want to start building as many roads as possible at the same time.

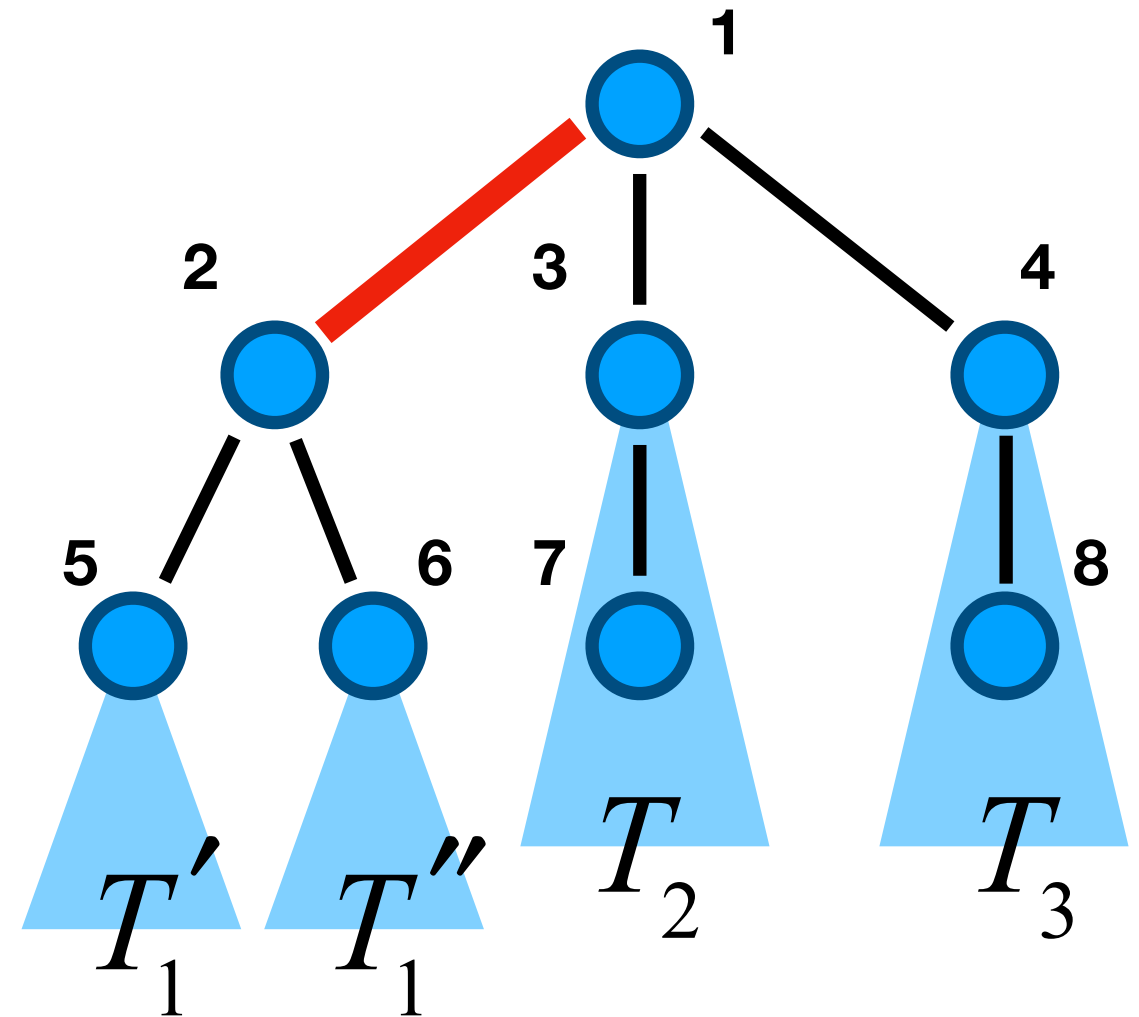
There is a constraint, though: to avoid too much disruption you do not want to build two roads ending in the same city.

Goal: Find a largest set of vertex disjoint edges **in a tree**

Observations



1. Don't take any edge incident to vertex 1.
2. Maximum matching in each subtree can be found **independently**.

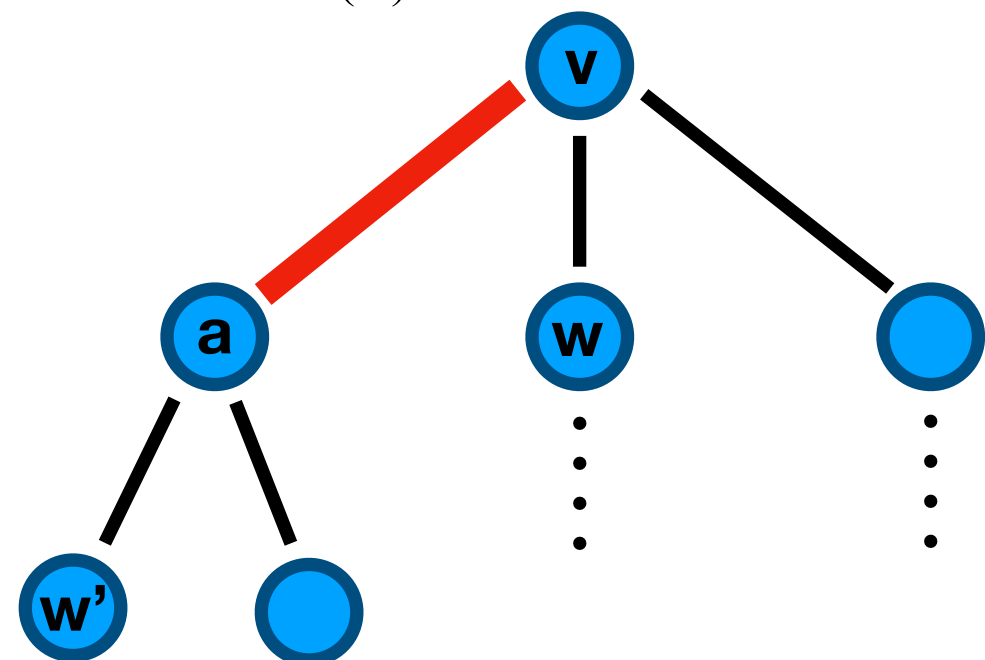
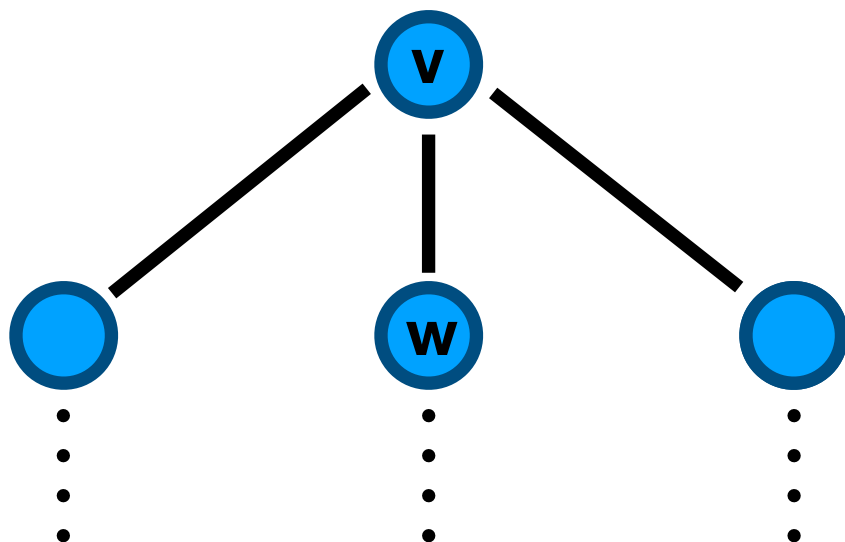


1. Take an edge incident to a child of vertex 1.
2. Maximum matching in each subtree can be found **independently**.

Formulation as a DP

- Let $c(v)$ denote the **set of children** of v in the tree
- Let $M(v)$ denote the **size of the largest matching** in the subtree rooted at v

$$M(v) = \max \left\{ \begin{array}{l} \sum_{w \in c(v)} M(w) \\ \max_{a \in c(v)} 1 + \sum_{w \in c(v) \setminus a} M(w) + \sum_{w' \in c(a)} M(w') \end{array} \right.$$



Implementation

$$M(v) = \max \left\{ \begin{array}{l} \sum_{w \in c(v)} M(w) \\ \max_{a \in c(v)} 1 + \sum_{w \in c(v) \setminus a} M(w) + \sum_{w' \in c(a)} M(w') \end{array} \right.$$

```
matching(v) {  
    S = 0;  
    for all w in c(v) : S = S + M(w);  
    M(v) = S;
```

```
    for all a in c(v):  
        S = 1;  
        for all w in c(v) \ a:  
            S = S + M(w);  
        for all w' in c(a):  
            S = S + M(w');  
    M(v) = max(M(v), S);
```

```
}
```

Can be $\mathcal{O}(n^2)$ if v has n-1 decendants

Implementation

$$M(v) = \max \left\{ \begin{array}{l} \sum_{w \in c(v)} M(w) \\ \max_{a \in c(v)} 1 + \sum_{w \in c(v) \setminus a} M(w) + \sum_{w' \in c(a)} M(w') \end{array} \right.$$

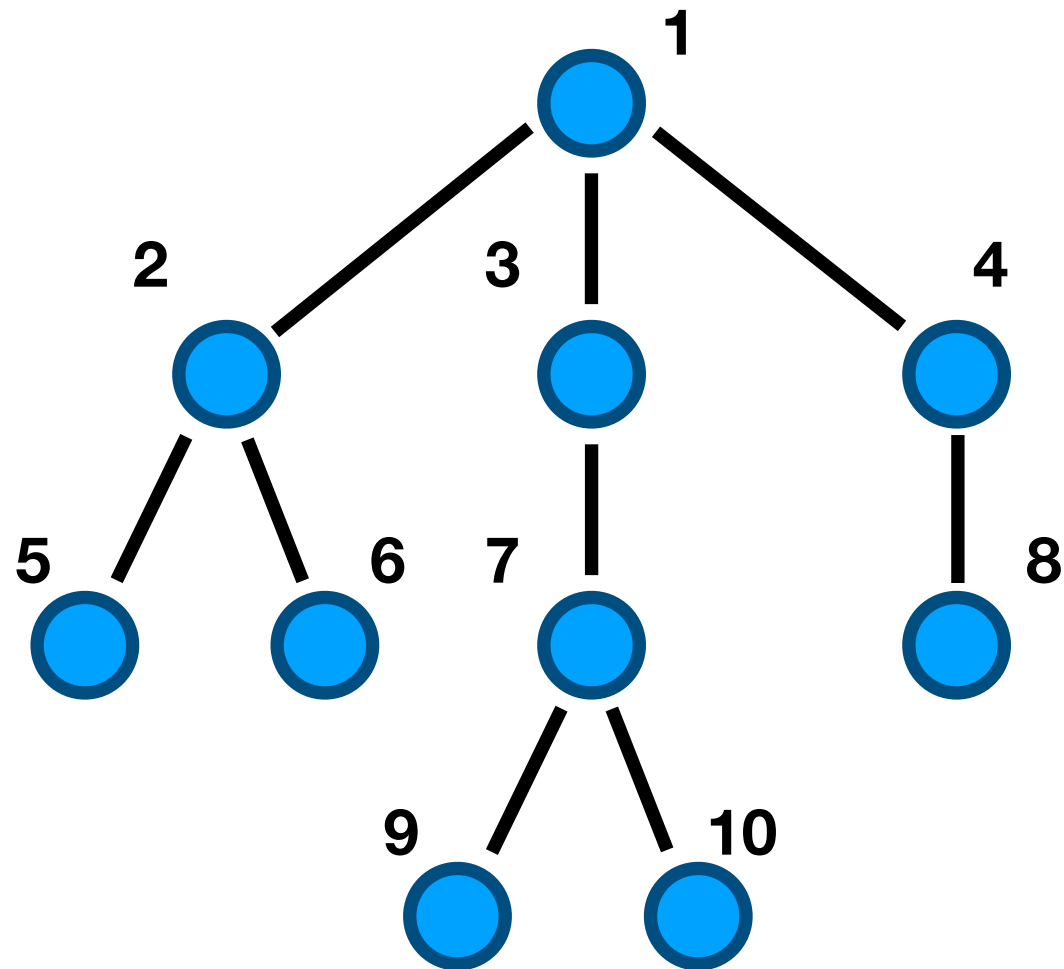
```
matching(v) {  
  S = 0;  
  for all w in c(v) : S = S + M(w);  
  M(v) = S; M'(v) = S;
```

```
  for all a in c(v):  
    S = 1 + M'(v) - M(a);  
    for all w' in c(a):  
      S = S + M(w');  
    M(v) = max(M(v), S);
```

```
}
```

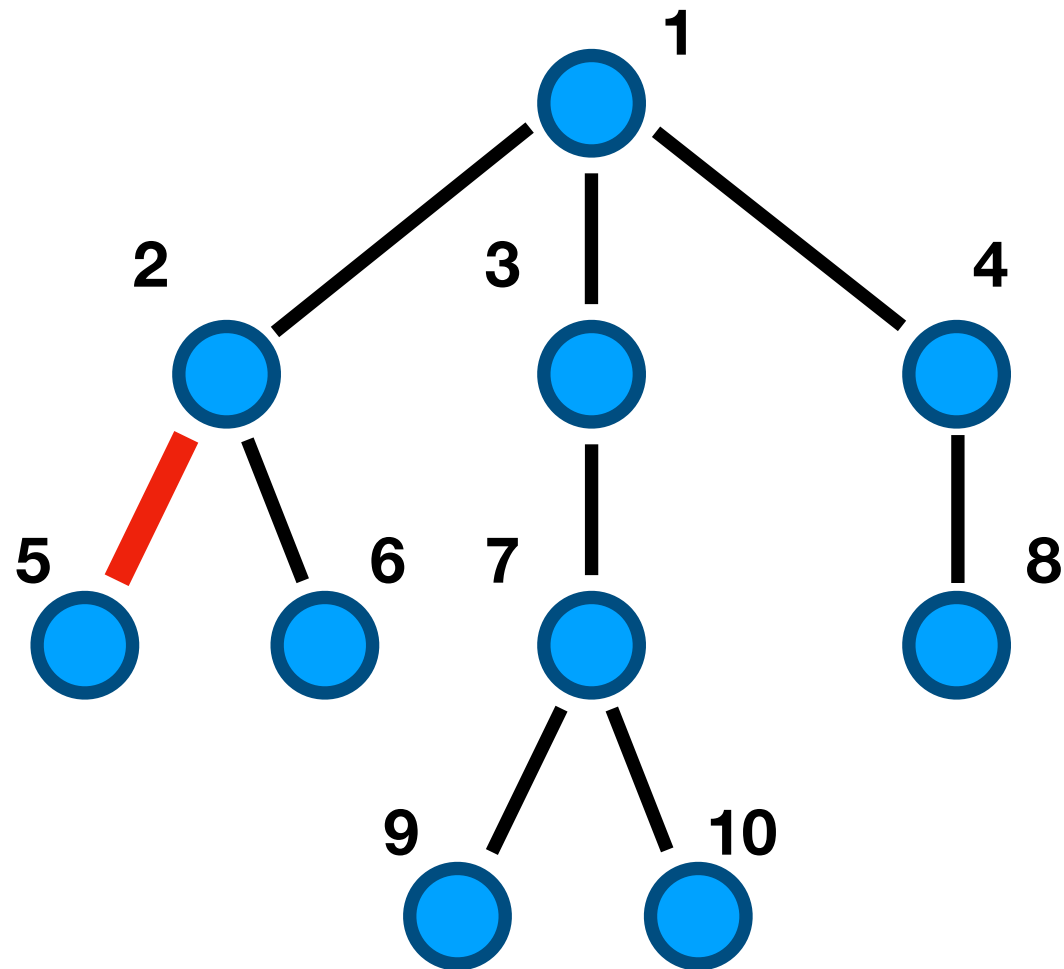
Exercise: Show that the running time is linear

Greedy Solution



Repeat: Take an edge which contains a leaf and remove its endpoints and all edges incident to the parent.

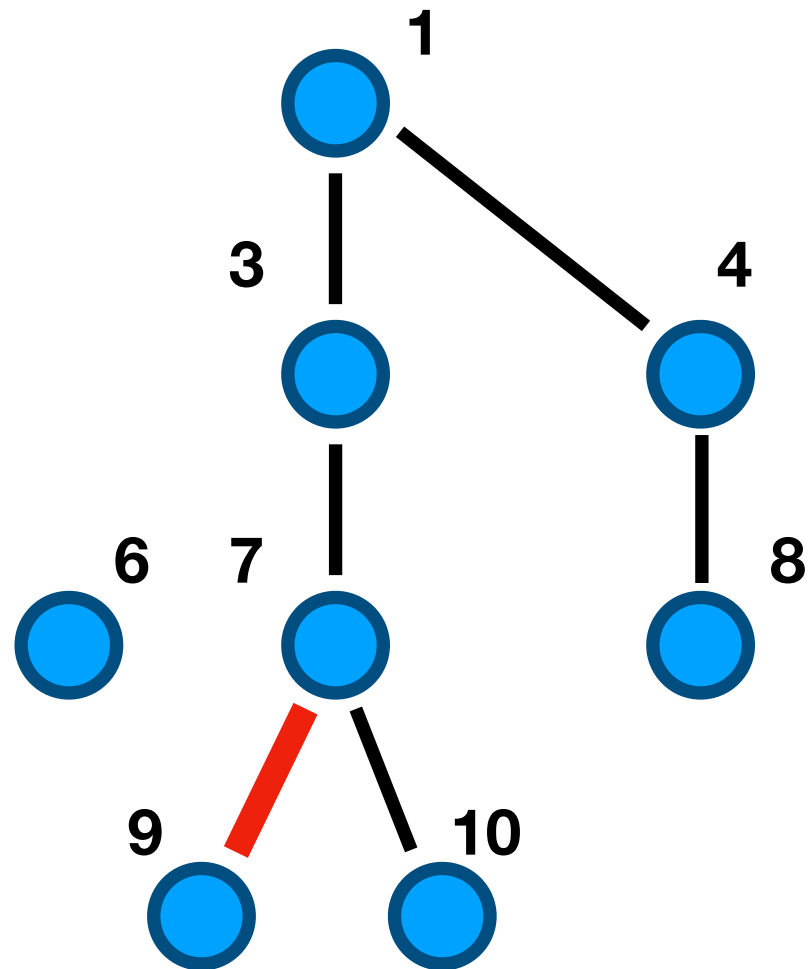
Greedy Solution



Repeat: Take an edge which contains a leaf and remove its endpoints and all edges incident to the parent.

{2,5}

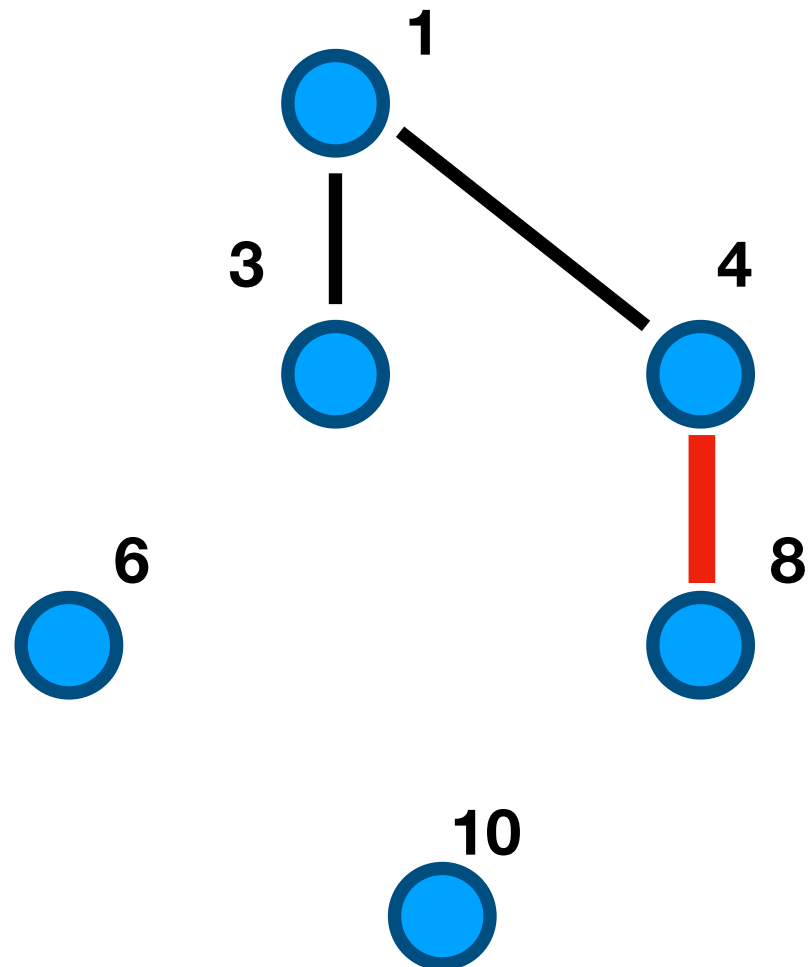
Greedy Solution



Repeat: Take an edge which contains a leaf and remove its endpoints and all edges incident to the parent.

{2,5} {7,9}

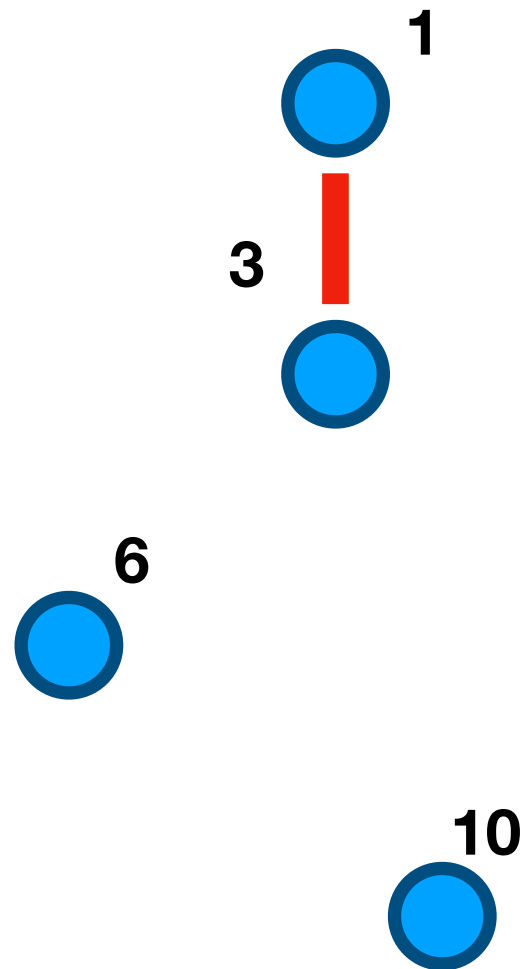
Greedy Solution



Repeat: Take an edge which contains a leaf and remove its endpoints and all edges incident to the parent.

{2,5} {7,9} {4,8}

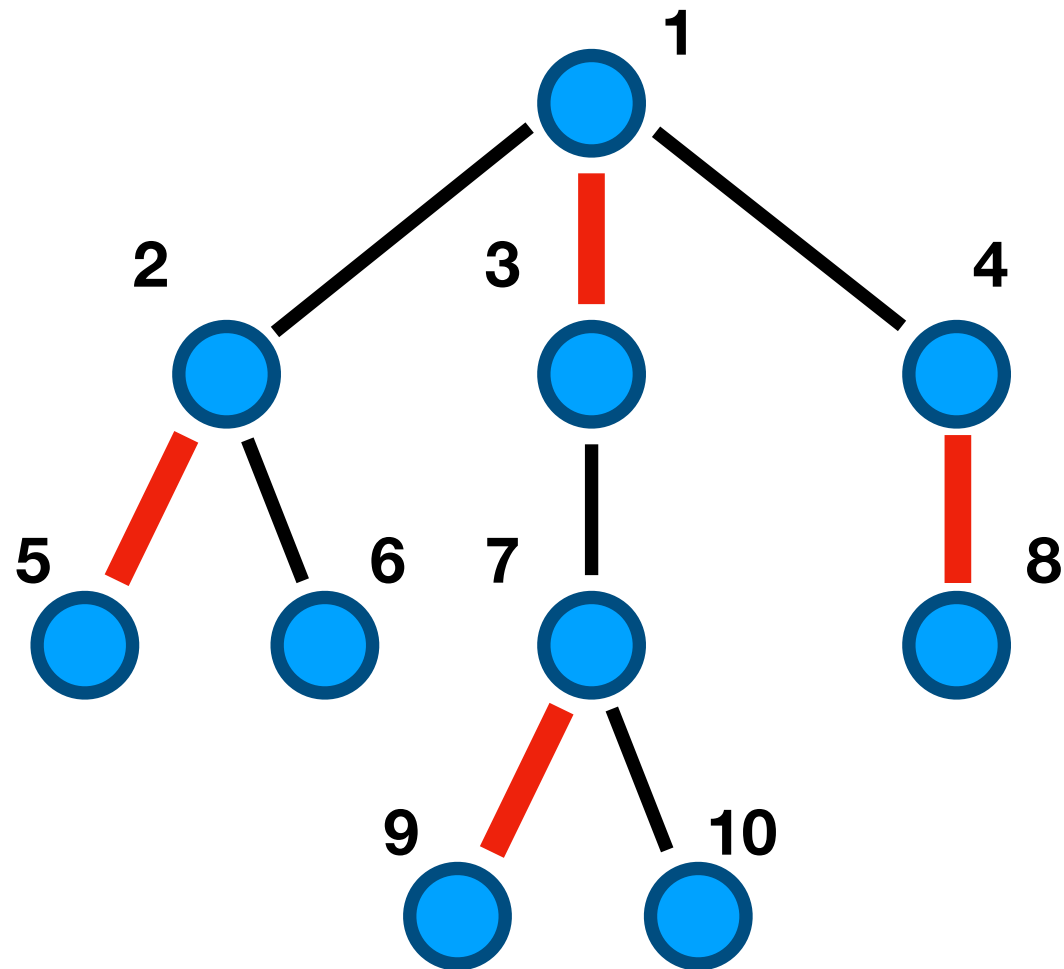
Greedy Solution



Repeat: Take an edge which contains a leaf and remove its endpoints and all edges incident to the parent.

{2,5} {7,9} {4,8} {1,3}

Greedy Solution



Repeat: Take an edge which contains a leaf and remove its endpoints and all edges incident to the parent.

Exercise: prove its correctness by using the exchange argument (see slides from the 5th week)

$\{2,5\}$ $\{7,9\}$ $\{4,8\}$ $\{1,3\}$

Exercise 2

Exercise – *Consecutive Constructions*

The basic roads in Algoland are already built, but the road network still needs to be extended.

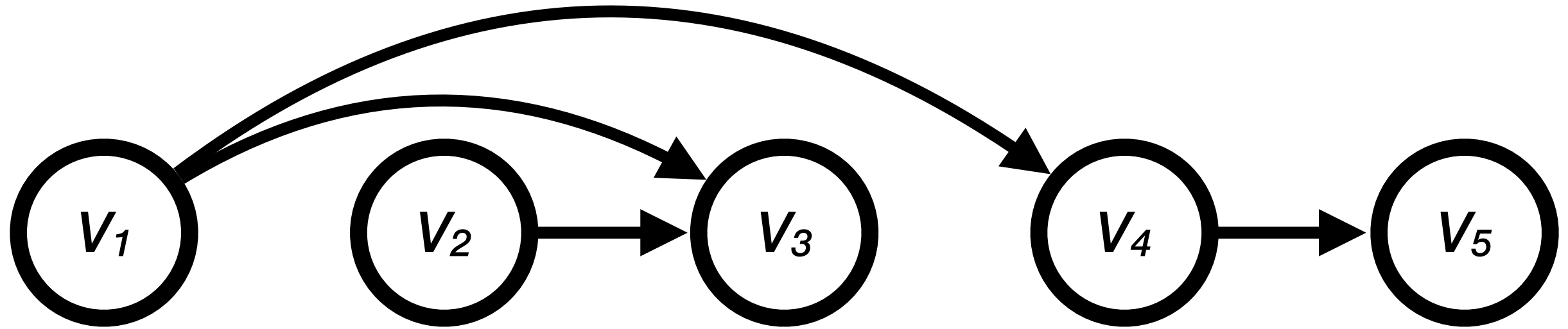
The roads to be constructed now are in a more difficult terrain, so, if possible, you want to build them in consecutive intervals. A single construction team with all the heavy equipment can start building the next road in the same city just after finishing the previous one.

The rules are as follows:

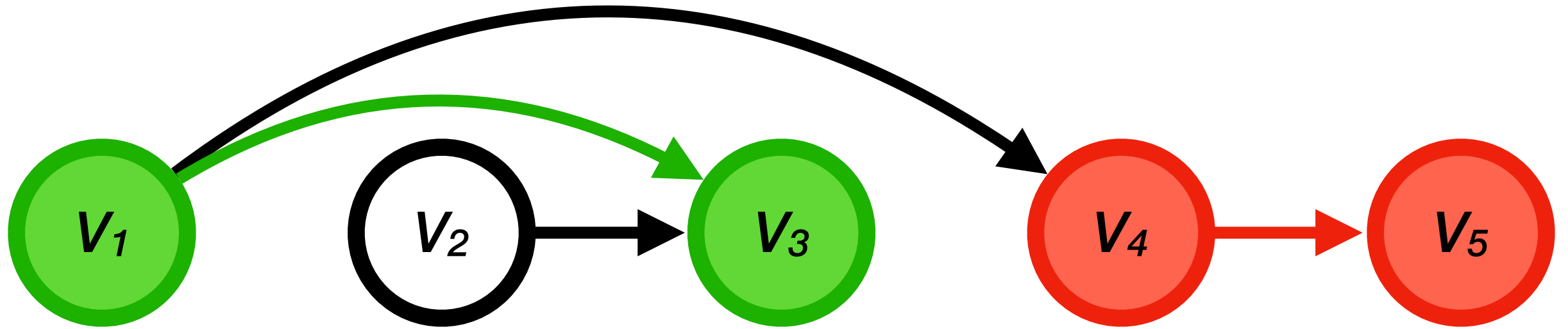
- You can use as many teams as you need for operating the construction.
- Each team will start in some city u , and then build a road to a city with larger number v . After finishing the construction the team will be in v . Then it might build another road to a city with even larger number, and so on.
- To avoid disruptions each city can be visited at most once by at most one team. Starting or finishing in a city counts as a visit.
- You want to build as many roads as possible.

Goal: Given a **directed acyclic graph G** , find the maximum number of edges that can be packed in **vertex-disjoint paths** in G .

Example

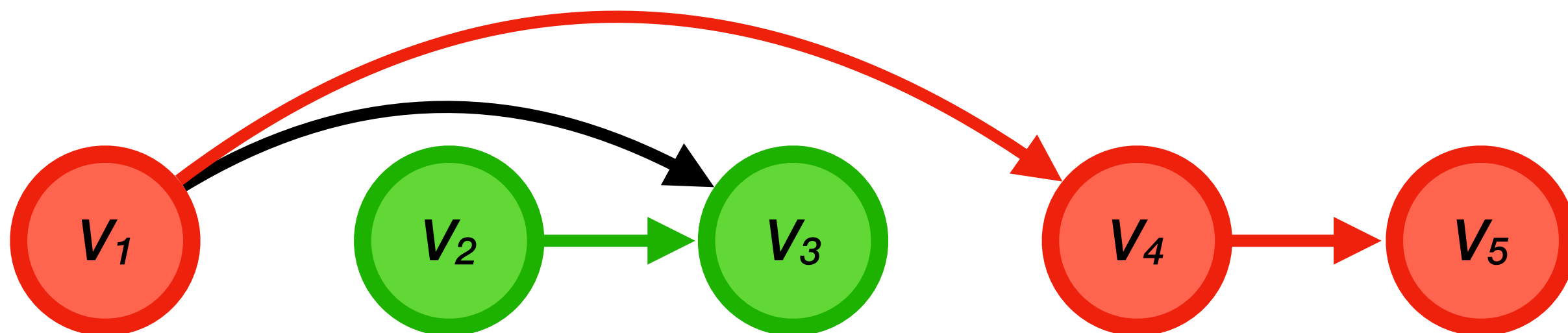


Greedy Strategy



Extend as much as possible

Optimal Solution



Three edges

What about a DP?

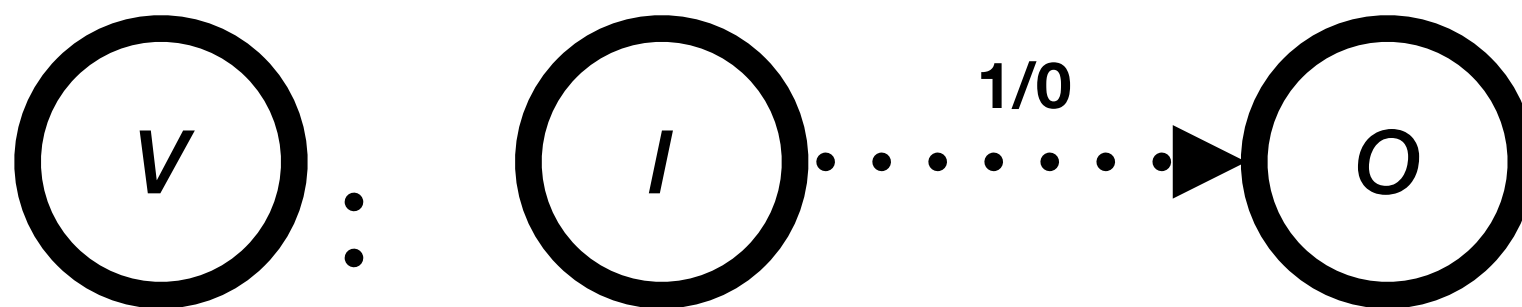
If we choose an edge $\{v_1, v_k\}$ we still need to find a solution on the vertices $v_2, \dots, v_{k-1}, v_{k+1}, \dots, v_n$ where parts before and after v_k are not independent.

Min-Cost Flow

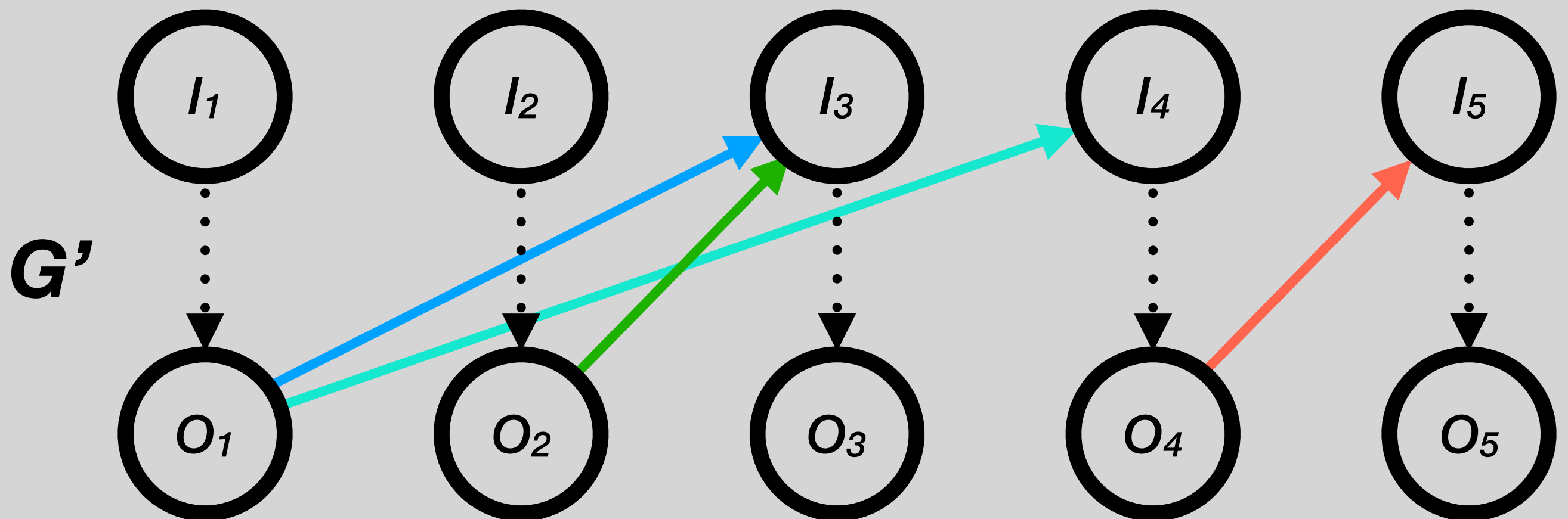
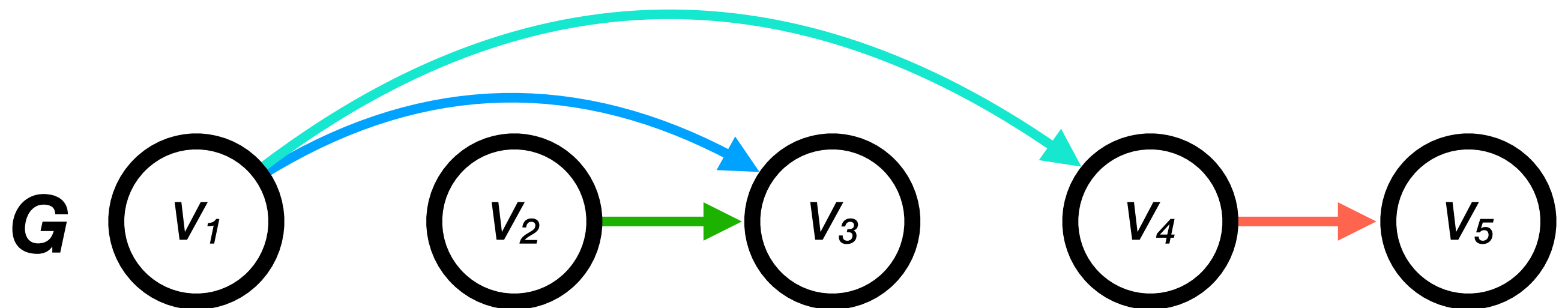
Model each team as a unit flow

Each edge is of capacity 1 and cost -1

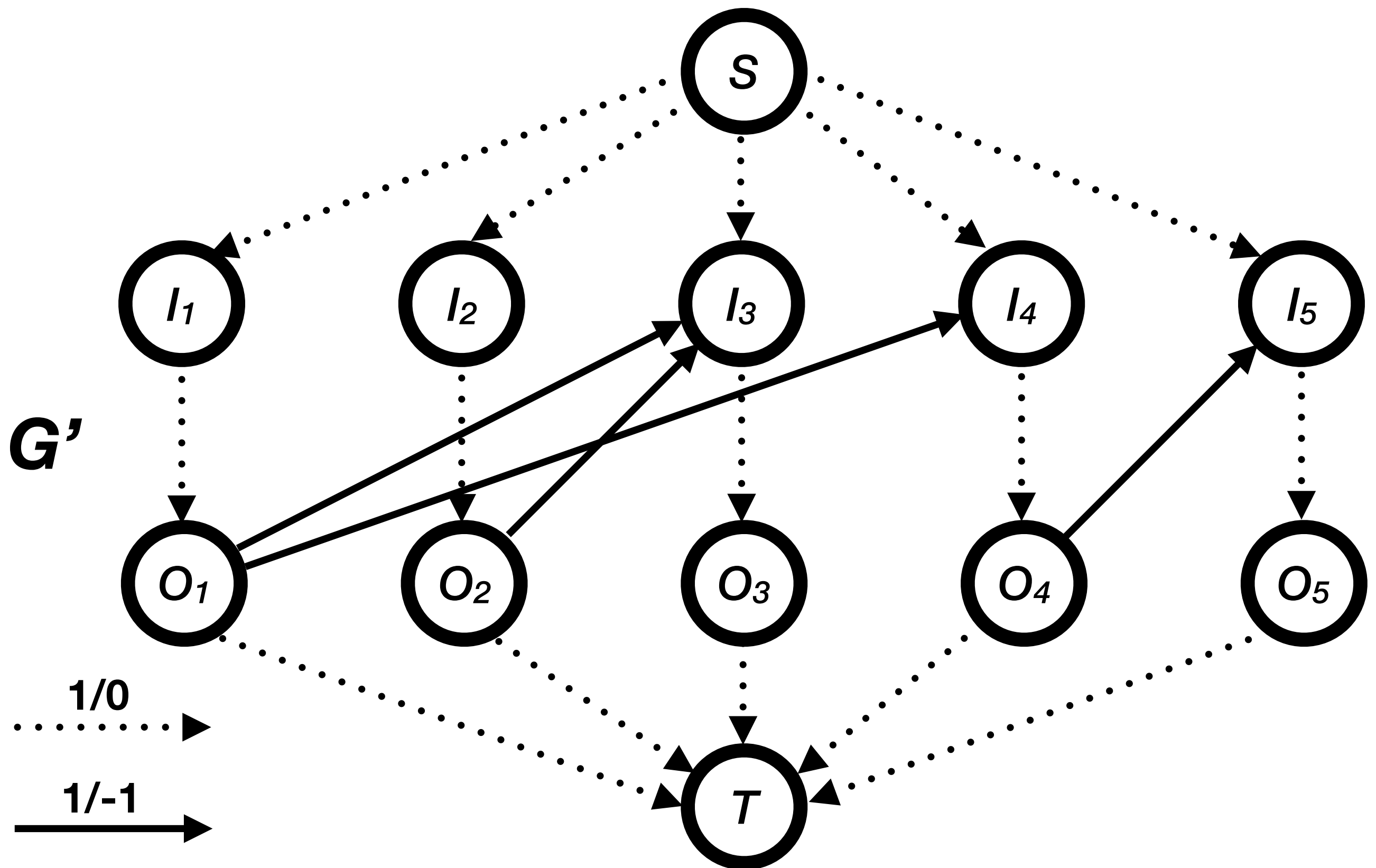
Vertex-disjoint paths: vertices are of capacity 1



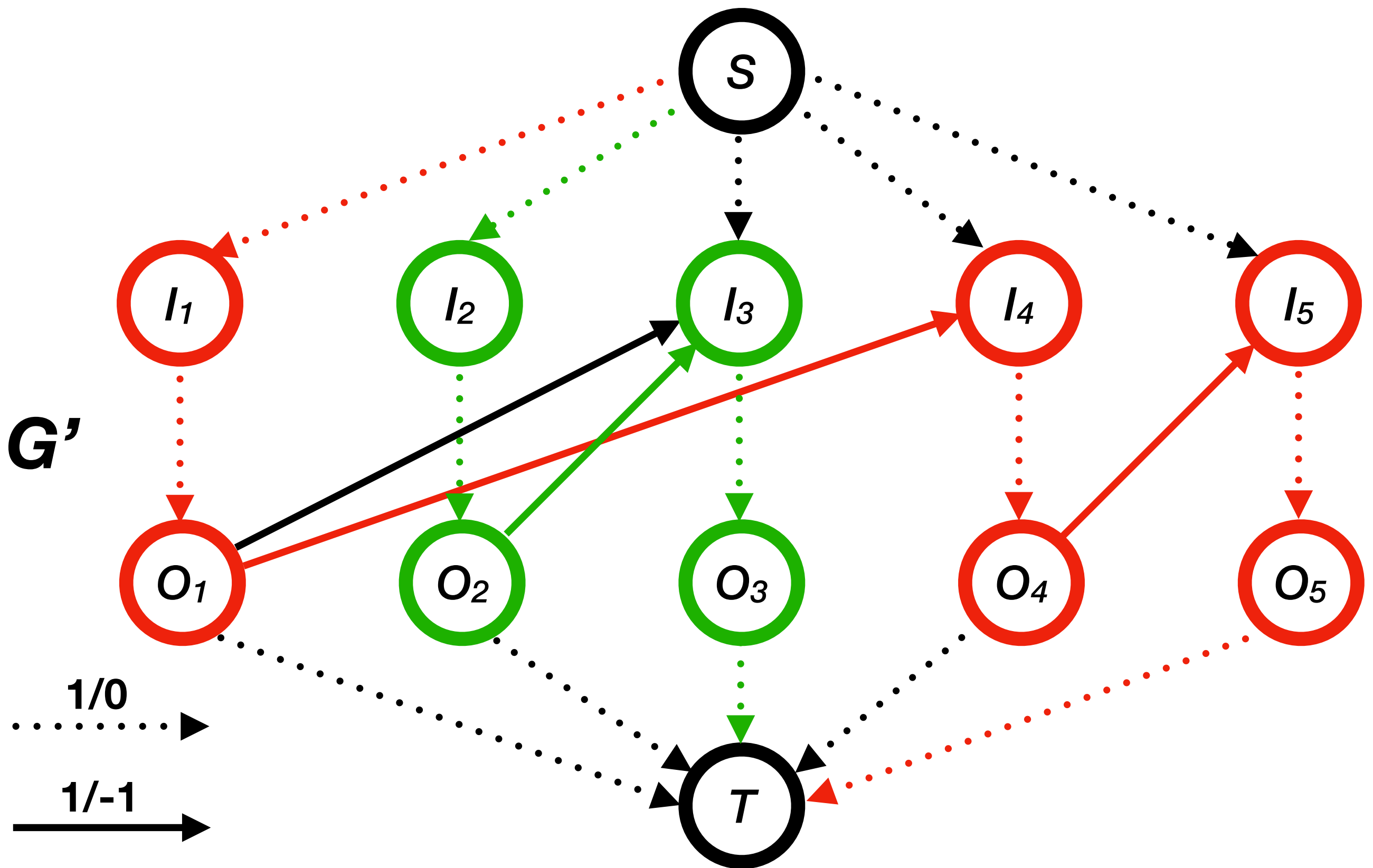
Doubling Vertices



Adding S , T , Capacity and Cost



Min-Cost Flow



Matching

Do we really need a min-cost flow?

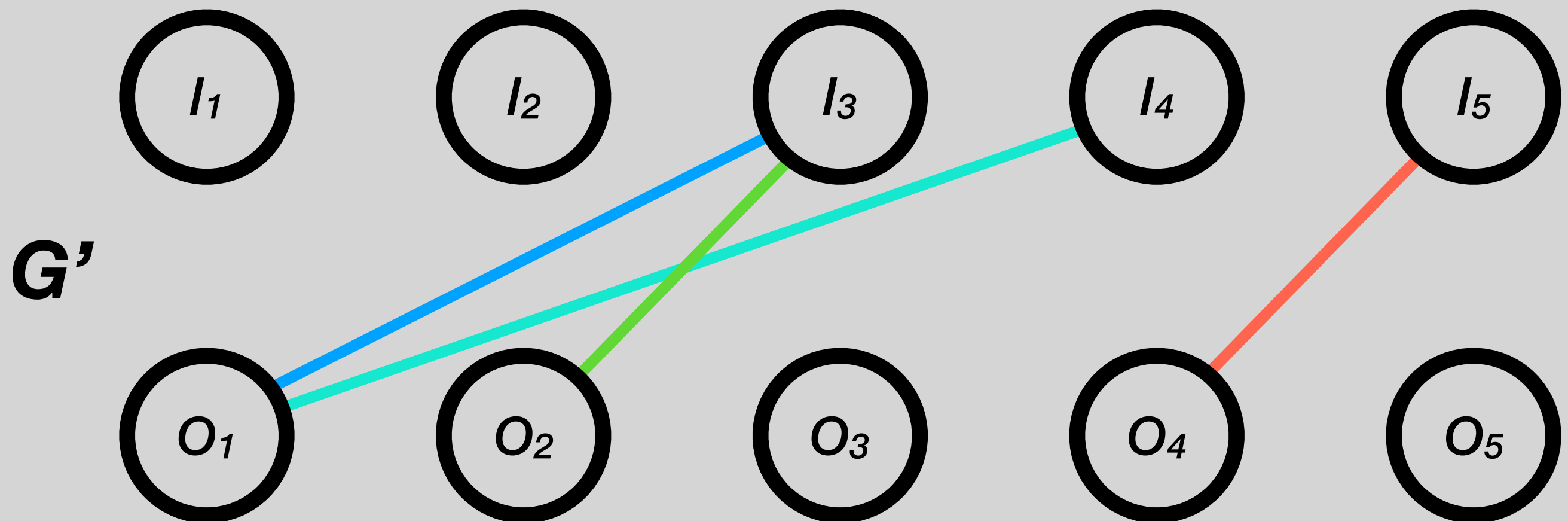
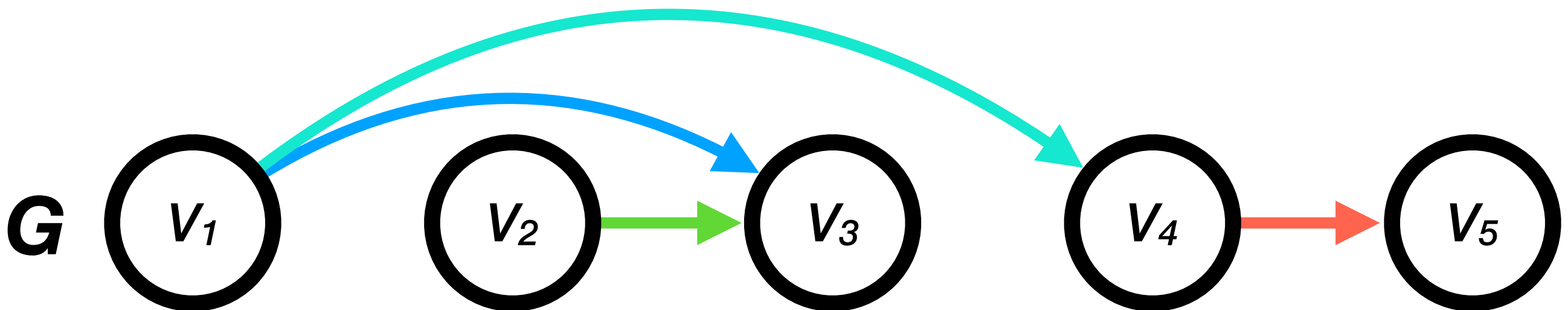
Only solid edges contribute to the cost.

Each vertex is incident to at most one solid edge.

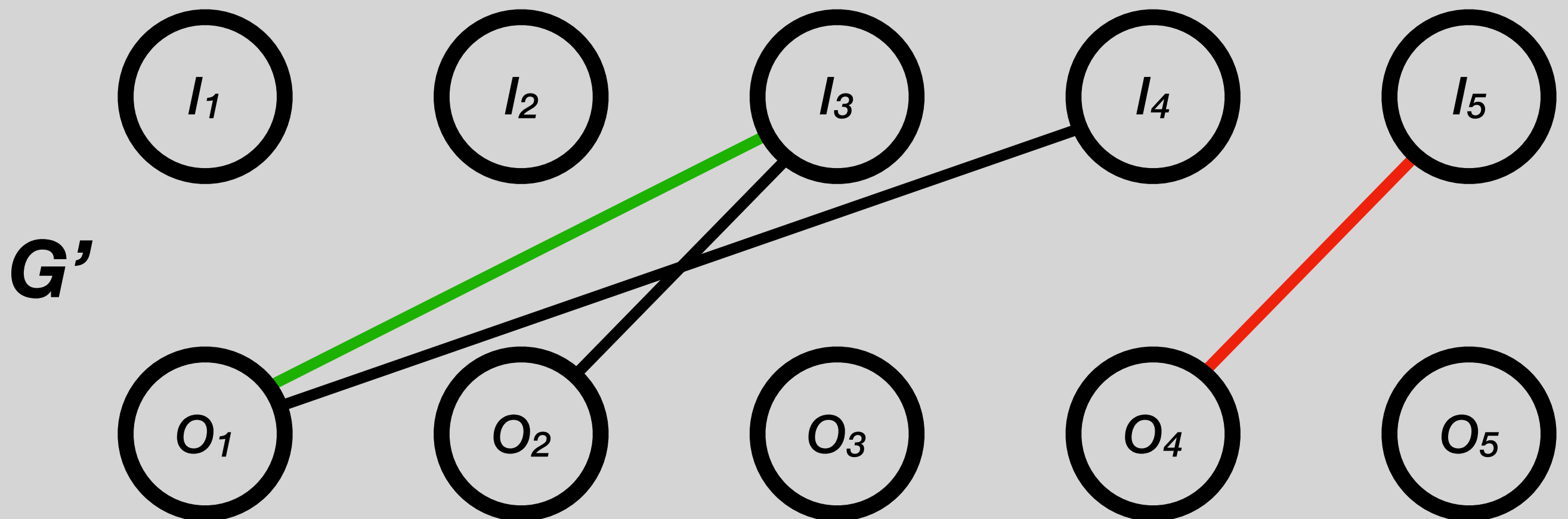
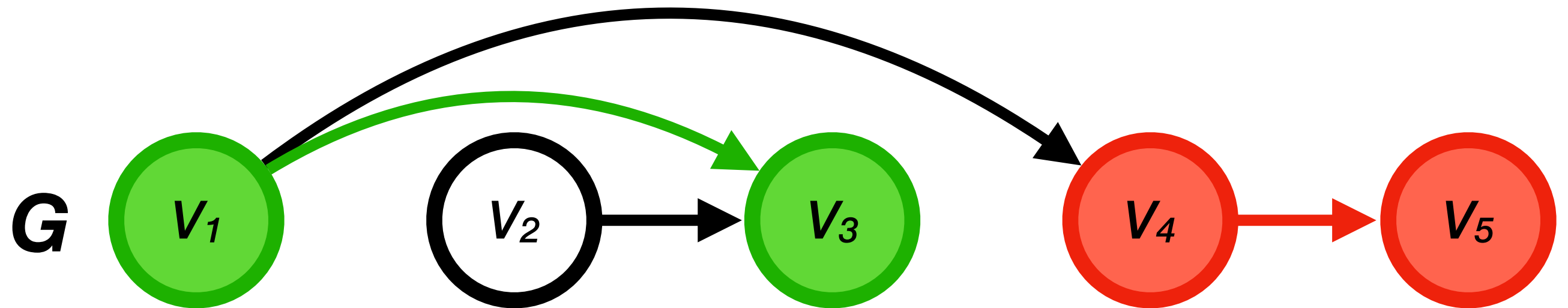
Lemma

There is a 1-1 correspondence between solutions in G with k edges and matchings in G' of size k .

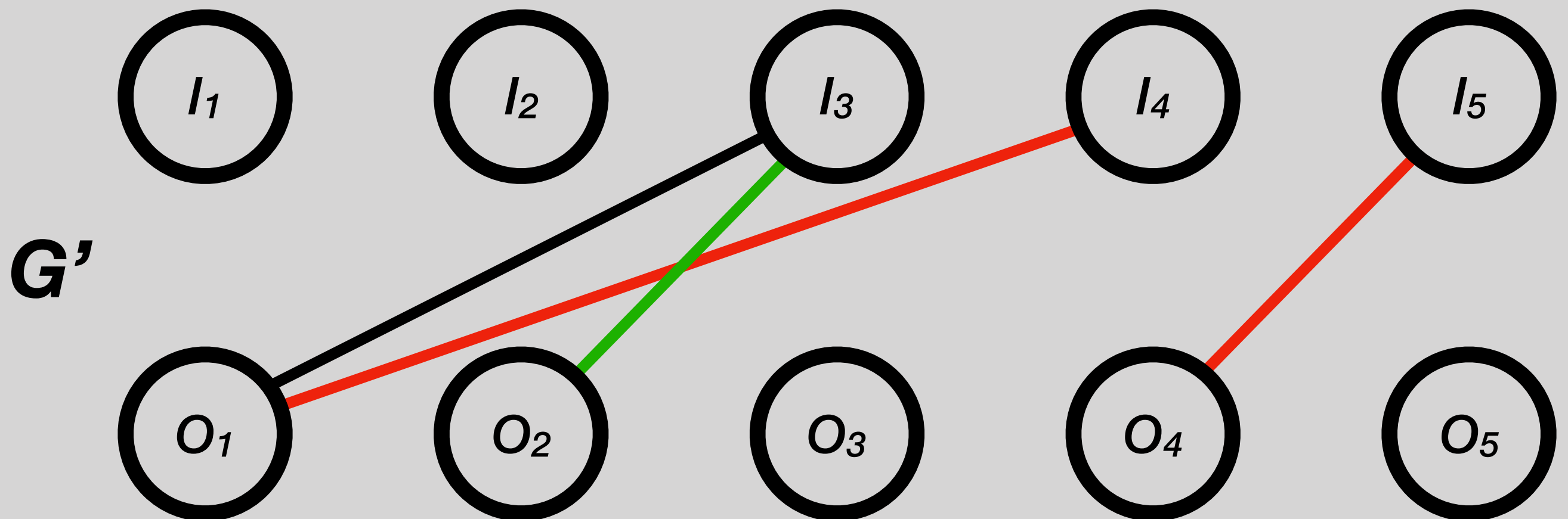
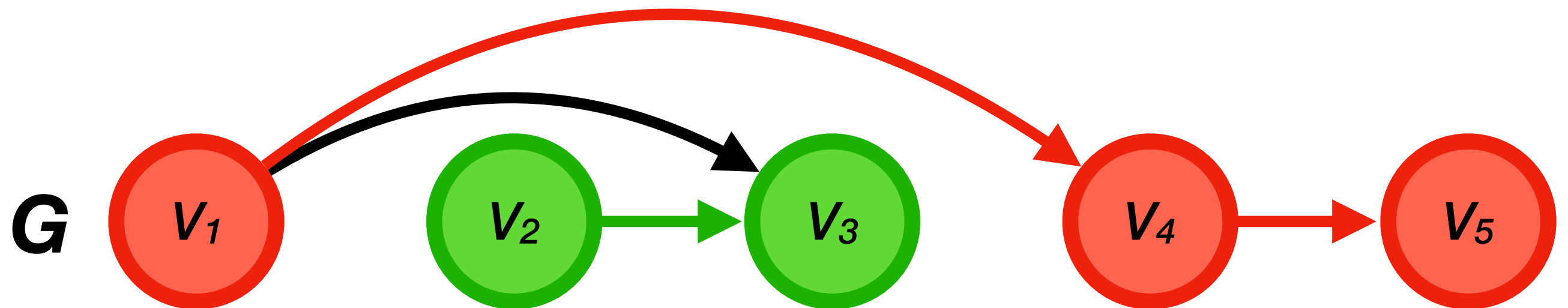
Modeling



Sub-optimal Solution



Optimal Solution



Exercise 3

Exercise – Missing Roads

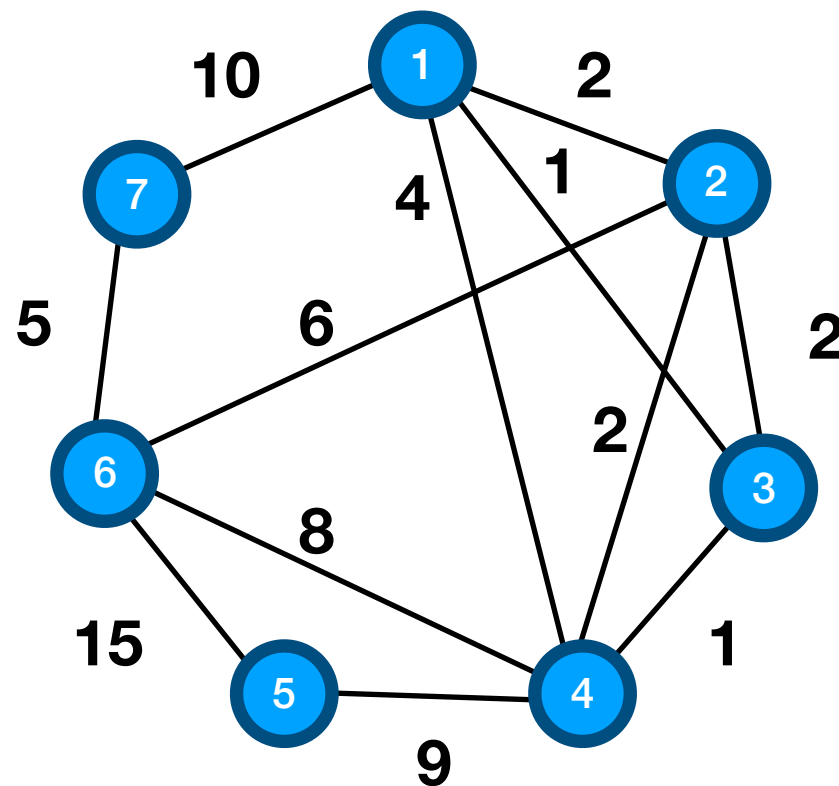
A lot of roads were built in Algoland, but some of them are still missing. Since many accidents happened last year at construction sites, the government decided to decrease the risk of injury by using a separate construction team for each road. However, as the road-building equipment makes a lot of noise when it is started, no two construction teams are allowed to start building from the same city. There is no problem though in two construction teams ending the road in the same city. A team assigned for one road can *choose* from which city will it start the construction.

There is not a lot of time and all roads must be built simultaneously, but you need to pay attention to the budget. Thus, given a construction cost for each potential road, estimate the minimum possible cost of building at least k of them, without starting a construction of more than one road at each city. Each given possible road has a cost between 1 and 100.

Goal: Given an undirected graph G and an integer k find **a set of vertices $\{v_1, \dots, v_k\}$** and edges $\{e_1, \dots, e_k\}$ such that

- vertex v_i is incident to e_i , and
- the sum of edge costs is the smallest possible.

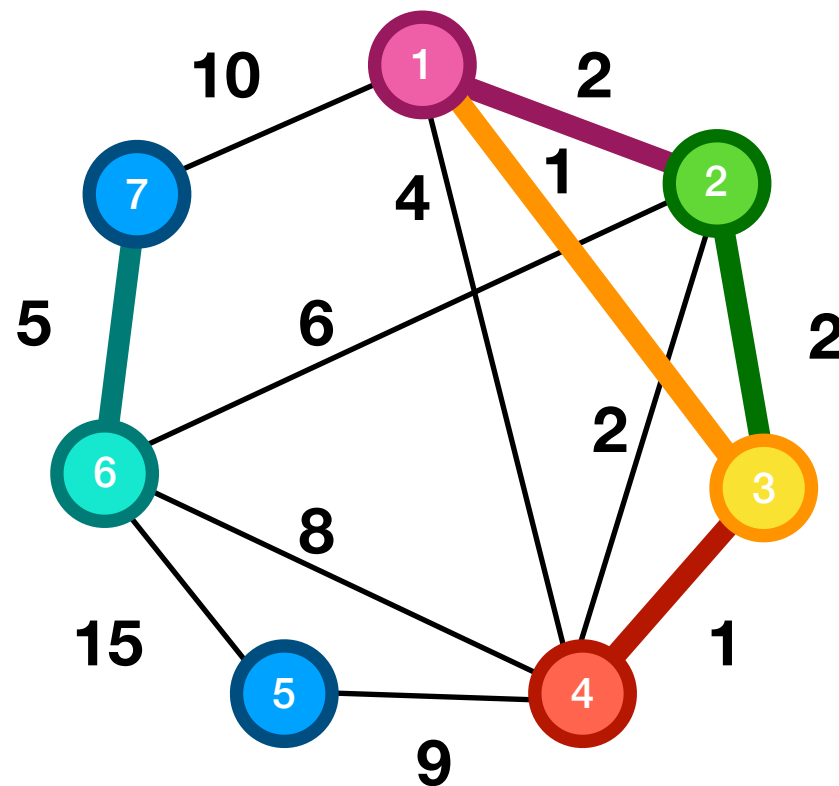
Example



Goal: Given an undirected graph G and an integer k find a set of vertices $\{v_1, \dots, v_k\}$ and edges $\{e_1, \dots, e_k\}$ such that

- vertex v_i is incident to e_i , and
- the sum of edge costs is the smallest possible.

$k=5$



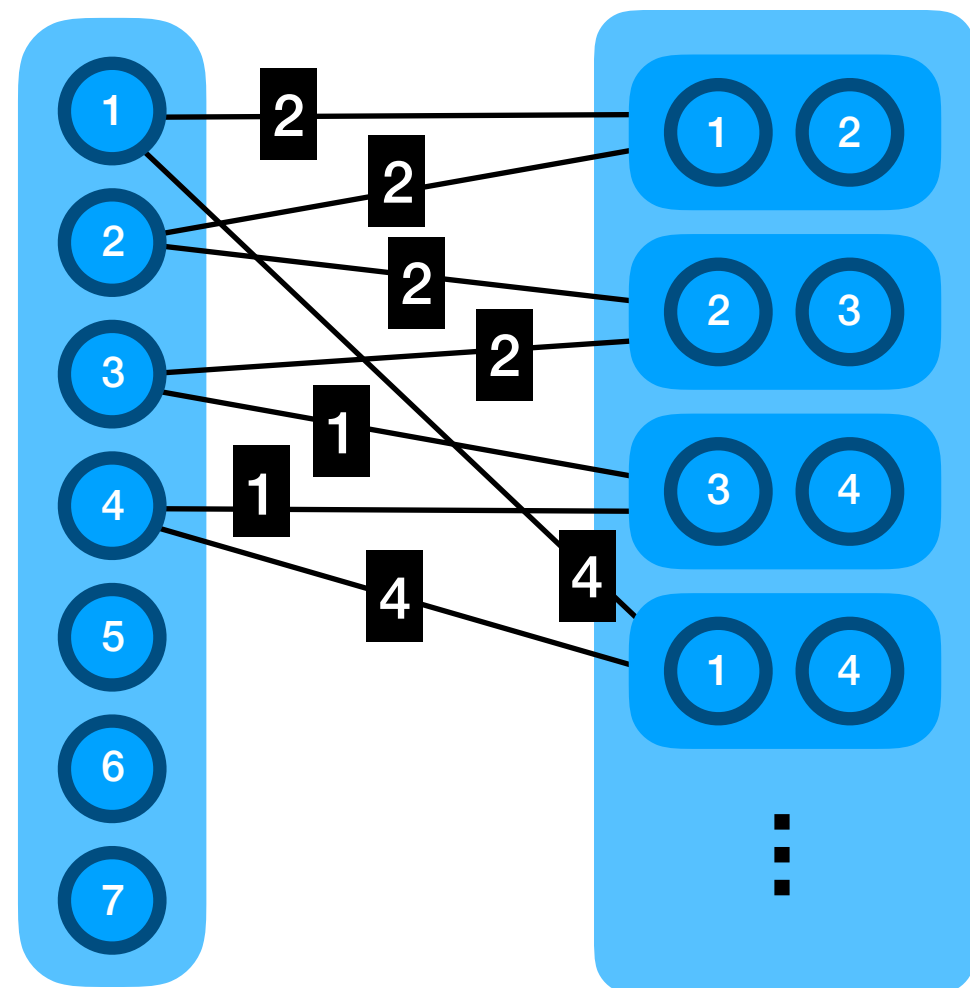
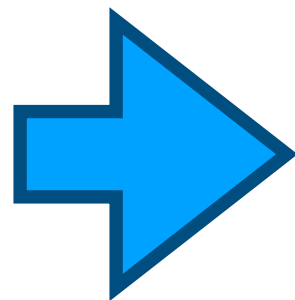
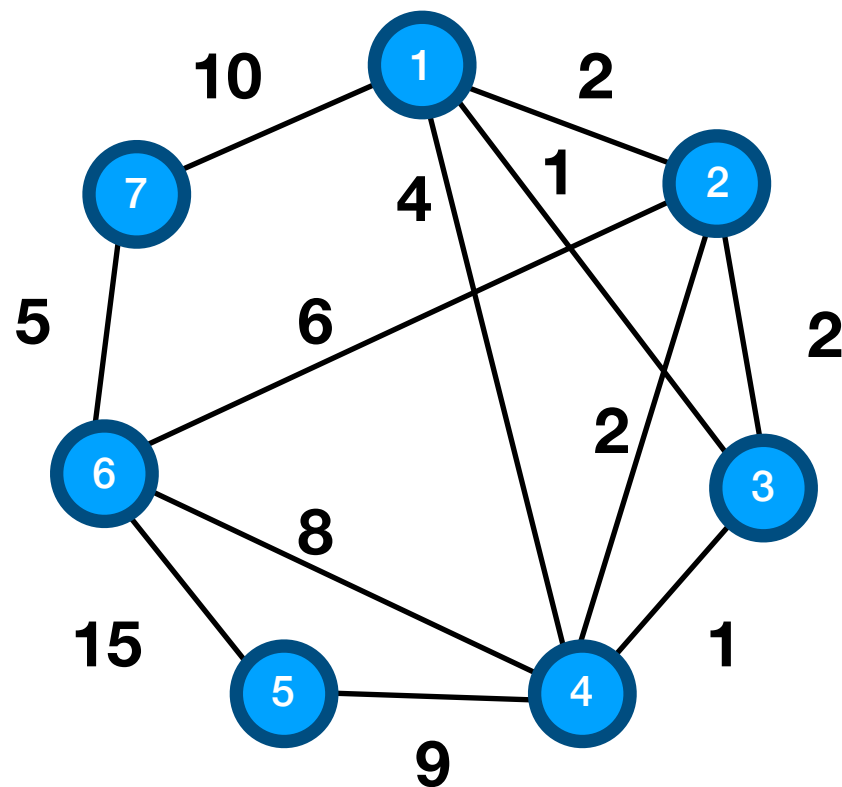
Goal: Given an undirected graph G and an integer k find a set of vertices $\{v_1, \dots, v_k\}$ and edges $\{e_1, \dots, e_k\}$ such that

- vertex v_i is incident to e_i , and
- the sum of edge costs is the smallest possible.

Solution

Create an **auxiliary bipartite** graph B :

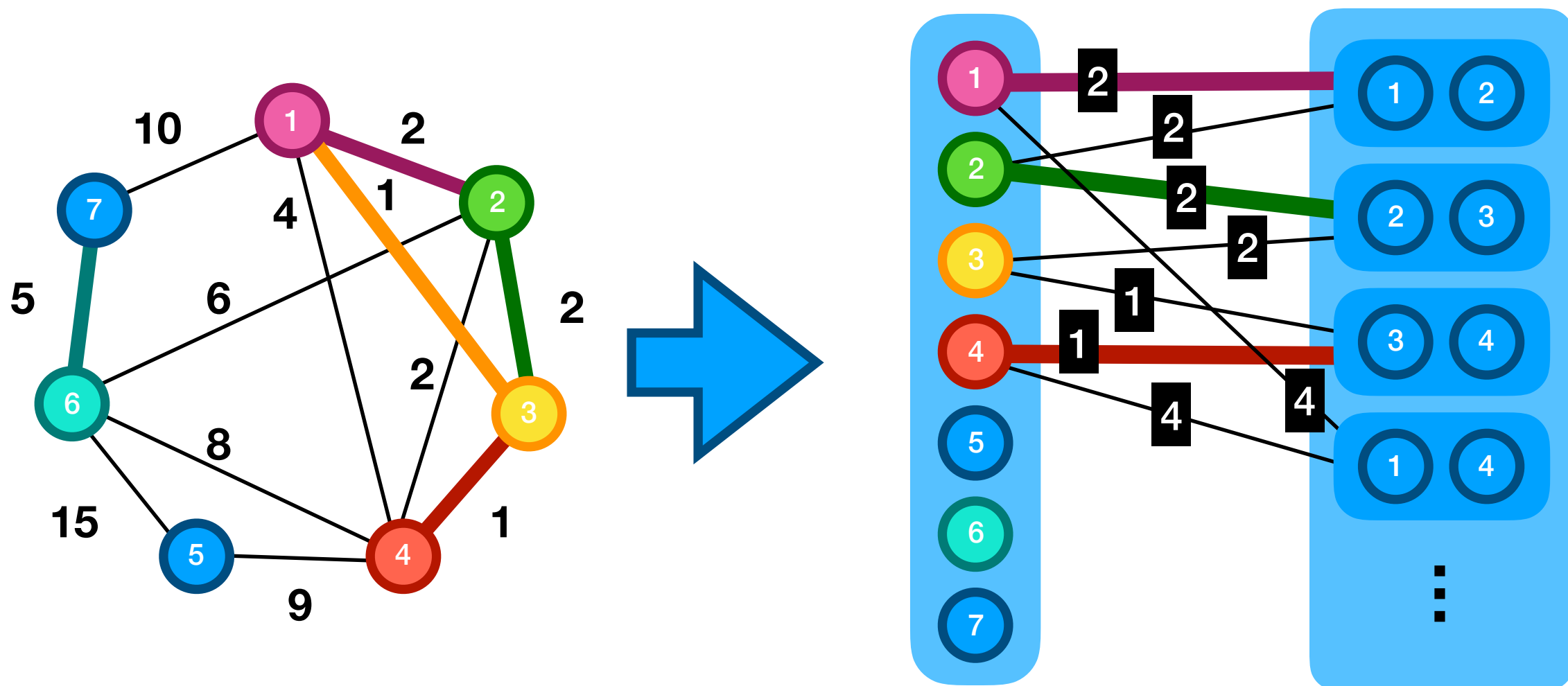
- One side contains vertices of G .
- The other side contains edges (as vertices)
- Connect a vertex v to an edge e if they are connected in G .



Solution

Create an **auxiliary bipartite** graph B :

- One side contains vertices of G .
- The other side contains edges (as vertices)
- Connect a vertex v to an edge e if they are connected in G .



Solution

Goal: Find cheapest matching of size k in B

- If all costs are the same check whether the maximum matching is of size at least k . (50 points)
- Otherwise reduce the problem to a min-cost max-flow problem.

