MATH 3290 Mathematical Modeling

Chapter 8: Modeling Using Graph Theory

Kuang HUANG

March 1, 2024

Department of Mathematics
The Chinese University of Hong Kong

Course webpage

https://www.math.cuhk.edu.hk/course/2324/math3290

SCAN ME

About midterm

- Date: Mar. 15.
- The exam is a closed-book 90-min exam.
- Laptops, tablets, and smartphones are not permitted; however, calculators are allowed.
- The exam will cover Chap. 1 (Modeling changes), Chap. 3 (Model fitting), Chap. 4 (Experimental modeling), and Chap. 7 (Optimization of discrete models).
- You are expected to perform hand calculations for: least squares fitting, cubic splines, and linear programming, etc.
- There will be a review session for the tutorial and a Q&A session for the lecture on Mar. 13.

Notations

A graph G contains two sets: a vertex set $V(G)$ & an edge set $E(G)$. Every edge is a pair of vertices.

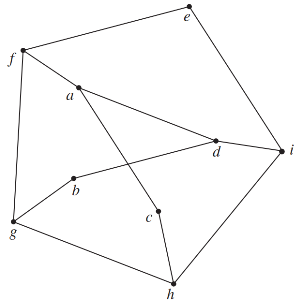
- The vertex set (9 vertices)

$$V(G) = \{a, b, c, d, e, f, g, h, i\}.$$

- The edge set (12 edges)

$$E(G) = \{ac, ad, af, bd, bg, ch, di, ef, ei, fg, gh, hi\}.$$

- Edges bd and ac cross, but the point of intersection is not regarded as a vertex.

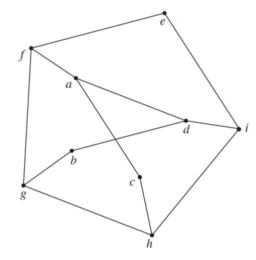


More notations

- An edge ab has two vertices a and b, we say the edge ab is incident with a (also incident with b).
- An edge ab has two vertices a and b, we say a and b are adjacent.
- The degree of a vertex b, $\deg(b)$, is the number of edges having vertex b.

Several facts:

- c and h are adjacent, b and c are not.
- $\deg(d) = 3$, $\deg(e) = 2$.



Example 1: Social network

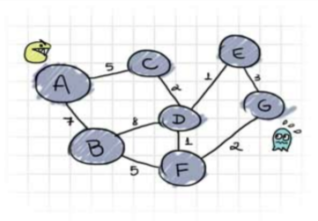
A social network can be modeled by a graph:

- Each user is considered as a vertex.
- Two users can form an edge if they are friends.
- One interesting problem is the degree of separation, it is the shortest distance between any 2 users.
- The average degree of separation of Facebook users is 4.57 in 2016, while 4.74 in 2011.

Example 2: Route planning

Route planning problem can be modeled by a graph:

- Each road intersection is considered as a vertex.
- A road between two adjacent intersections is an edge.
- The problem is to find a path giving the shortest distance between 2 destinations.
- We see that there is a need to give weights to edges.



Finding shortest paths

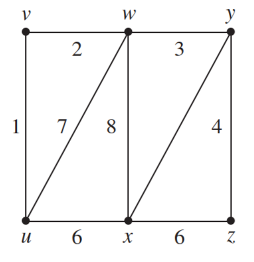
Let G be a graph, with the vertex set $V(G)$ and the edge set $E(G)$.

We assign a length to each edge. For the edge ab, the length is c_{ab}.

Let s and t be given vertices. We find a shortest path from s to t.

- There are 6 vertices.
- Each edge has a length.
- The path $u - v - w - y$ has a length of 6.

Q: can you find the shortest path from u to y?



Dijkstra's algorithm

E. W. Dijkstra (1930-2002)

E. W. Dijkstra, a Dutch computer scientist (Turing Award laureate), programmer, software engineer, systems scientist, and science essayist.

Dijkstra's algorithm

Let s and t be given vertices. We find the shortest path from s to t.

Step 1: **(Initialize)** give temporary labels L to vertices, $L(s) = 0$, and $L(i) = \infty$ for other vertices.

Step 2: **(Select one as permanent)** among all vertices with the smallest temporary labels, choose one of them as permanent.

Step 3: **(Update)** for every vertex a without a permanent label, compute a new temporary label $L(a)$ as

$$L(a) = \min\{L(i) + c_{ia}\},$$

where the min is taken over all vertices i with permanent label.

Step 4: go back to **Step 2**.

When the algorithm stops, the number $L(i)$ is the length of the shortest path from s to i (not only for t).

An illustration

We find the shortest paths from the vertex u.

We use a vector with 6 entries, $(L(u), L(v), L(w), L(x), L(y), L(z))$, to represent the labels of the 6 vertices.

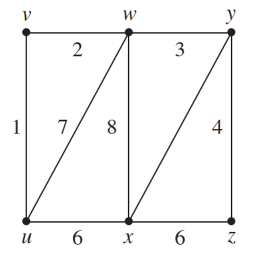
Step 1: (initialize) the source vertex has a label 0 and others have ∞

$$L = (0, \infty, \infty, \infty, \infty, \infty).$$

Step 2: (choose one permanent) the vertex u is chosen as permanent

$$L = (0^*, \infty, \infty, \infty, \infty, \infty),$$

where $*$ represents permanent.



Step $\boxed{3}$: (update),

- for vertex v

$$L(v) = L(u) + c_{uv} = 0 + 1 = 1;$$

- for vertex w

$$L(w) = L(u) + c_{uw} = 0 + 7 = 7;$$

- for vertex x

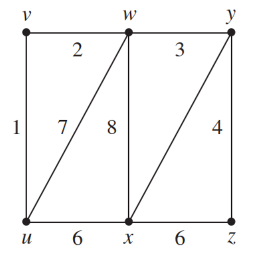
$$L(x) = L(u) + c_{ux} = 0 + 6 = 6;$$

- For vertices y and z, the labels remain the same.

New $L = (0^*, 1, 7, 6, \infty, \infty)$.

Currently,

$$L = (0^*, \infty, \infty, \infty, \infty, \infty).$$



Step **2**: (choose one permanent)

$$L = (0^*, 1^*, 7, 6, \infty, \infty).$$

Step **3**: (update)

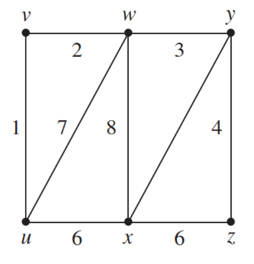
$$\begin{aligned} L(w) &= \min\{L(u) + c_{uw}, L(v) + c_{vw}\} \\ &= \min\{7, 3\} = 3, \end{aligned}$$

$$L(x) = L(u) + c_{ux} = 0 + 6 = 6.$$

No need to update $L(y), L(z)$.

Currently,

$$L = (0^*, 1, 7, 6, \infty, \infty).$$



So, we get

$$L = (0^*, 1^*, 3, 6, \infty, \infty).$$

Step 2: (choose one permanent)

$$L = (0^*, 1^*, 3^*, 6, \infty, \infty).$$

Step 3: (update)

$$\begin{aligned} L(x) &= \min\{L(u) + c_{ux}, L(w) + c_{wx}\} \\ &= \min\{6, 11\} = 6, \end{aligned}$$

$$L(y) = L(w) + c_{wy} = 3 + 3 = 6.$$

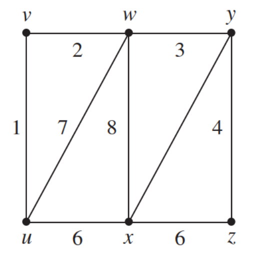
No need to update $L(z)$.

So, we get

$$L = (0^*, 1^*, 3^*, 6, 6, \infty).$$

Currently,

$$L = (0^*, 1^*, 3, 6, \infty, \infty).$$



Step 2: (choose one permanent)

$$L = (0^*, 1^*, 3^*, 6, 6^*, \infty).$$

Two "6" here, choose one arbitrarily.

Step 3: (update)

$$\begin{aligned} L(x) &= \min\{L(u) + c_{ux}, L(w) + c_{wx}, \\ &\qquad L(y) + c_{yx}\} \\ &= \min\{6, 11, 6+4\} = 6, \end{aligned}$$

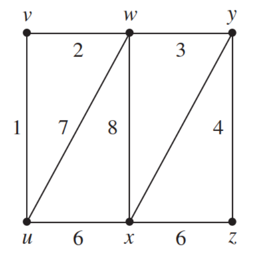
$$L(z) = L(y) + c_{zy} = 6 + 4 = 10.$$

So, we get

$$L = (0^*, 1^*, 3^*, 6, 6^*, 10).$$

Currently,

$$L = (0^*, 1^*, 3^*, 6, 6, \infty).$$



Step **2**: (choose one permanent)

$$L = (0^*, 1^*, 3^*, 6^*, 6^*, 10).$$

Step **3**: (update)

$$\begin{aligned} L(z) &= \min\{L(x) + c_{xz}, L(y) + c_{yz}\} \\ &= \min\{12, 10\} = 10 \end{aligned}$$

So, we get

$$L = (0^*, 1^*, 3^*, 6^*, 6^*, 10).$$

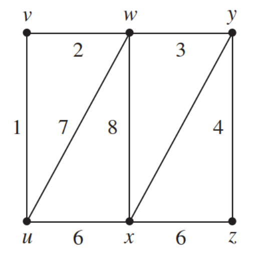
Step **2**: (choose one permanent)

$$L = (0^*, 1^*, 3^*, 6^*, 6^*, 10^*).$$

Done.

Currently,

$$L = (0^*, 1^*, 3^*, 6, 6^*, 10).$$



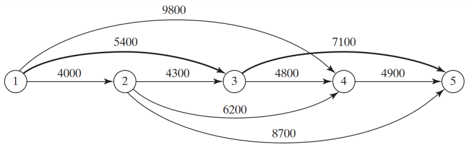
Example: Equipment replacement cost

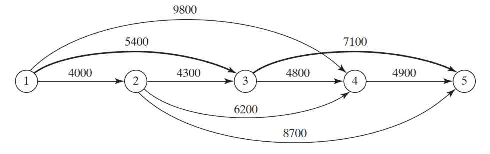
A rental car company is developing a replacement policy for the next 4 years. At the end of each year, a car can be replaced or kept.

A car must be in service for 1 to 3 years. The following is cost:

Equipment acquired at start of year	Replacement cost ($) for given years in operation		
	1	*2*	*3*
1	4000	5400	9800
2	4300	6200	8700
3	4800	7100	—
4	4900	—	—

It can be formulated as graph:





In the above graph, the edges have a direction.

We need to modified **Step** 3 (update) as follows:

For every vertex a without a permanent label, compute a new temporary label $L(a)$ as

$$L(a) = \min\{L(i) + c_{ia}\}$$

where the min is taken over all vertices i with permanent label and vertex a can be reached from vertex i.

For a given vertex i, we take $\widetilde{L}(i) = [L(i), j]$, where j is the vertex before i in the shortest path.

Step 1: **(Initialize)** give temporary labels $\widetilde{L}$ to vertices, $\widetilde{L}(s) = [0, -]$, and $\widetilde{L}(i) = [\infty, -]$ for other vertices.

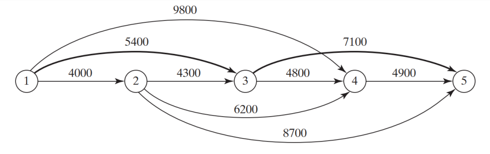
Step 2: **(Select one as permanent)** among all vertices with the smallest temporary labels, choose one of them as permanent.

Step 3: **(Update)** for every vertex a without a permanent label, compute a new temporary label $\widetilde{L}(a)$ as

$$\widetilde{L}(a) = [\min\{L(i) + c_{ia}\}, j]$$

where the min is taken over all vertices i with permanent label, and vertex j attains the above min.

Step 4: go back to **Step 2**.



Step **1**: we have $\widetilde{L} = ([0, -], [\infty, -], [\infty, -], [\infty, -], [\infty, -])$.

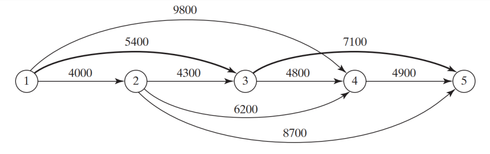
Step **2**: we have $\widetilde{L} = ([0^*, -], [\infty, -], [\infty, -], [\infty, -], [\infty, -])$.

Step **3**: vertex 1 can reach vertices 2, 3, and 4,

$$L(2) = L(1) + c_{12} = 4000, L(3) = L(1) + c_{13} = 5400,$$

$$L(4) = L(1) + c_{14} = 9800.$$

Hence, we have $\widetilde{L} = ([0^*, -], [4000, 1], [5400, 1], [9800, 1], [\infty, -])$.



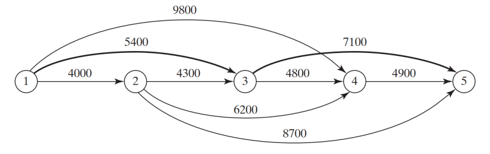
Step 2: we have $\widetilde{L} = ([0^*, -], [4000^*, 1], [5400, 1], [9800, 1], [\infty, -])$.

Step 3:

$$L(3) = \min\{L(1) + c_{13}, L(2) + c_{23}\} = \min\{5400, 8300\} = 5400.$$

Similarly, vertex 4 can only be reached from vertex 1, 2 and vertex 5 can be reached from 2.

Hence, we have $\widetilde{L} = ([0^*, -], [4000^*, 1], [5400, 1], [9800, 1], [12700, 2])$



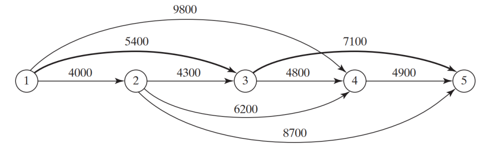
Step 2: we have
$\widetilde{L} = ([0^*, -], [4000^*, 1], [5400^*, 1], [9800, 1], [12700, 2])$.

Step 3:

$$L(4) = \min\{L(1) + c_{14}, L(2) + c_{24}, L(3) + c_{34}\} = 9800.$$

$$L(5) = \min\{L(2) + c_{25}, L(3) + c_{35}\} = 12500.$$

So, we have $\widetilde{L} = ([0^*, -], [4000^*, 1], [5400^*, 1], [9800, 1], [12500, 3])$.



Step 2: we have
$\widetilde{L} = ([0^*, -], [4000^*, 1], [5400^*, 1], [9800^*, 1], [12500, 3])$

Step 3:

$$L(5) = \min\{L(2) + c_{25}, L(3) + c_{35}, L(4) + c_{45}\} = 12500.$$

So, we have $\widetilde{L} = ([0^*, -], [4000^*, 1], [5400^*, 1], [9800^*, 1], [12500, 3])$.

Step 2: we have
$\widetilde{L} = ([0^*, -], [4000^*, 1], [5400^*, 1], [9800^*, 1], [12500^*, 3])$.

Hence, the minimum cost is 12,500. The path is $1 \to 3 \to 5$.

Hence, the minimum cost is 12,500. The path is $1 \to 3 \to 5$.

Remark: Scipy provides a module called csgraph that incorporates useful graph algorithms.

Dynamic programming

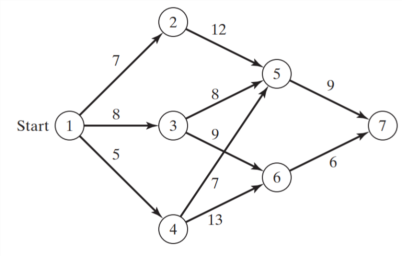
We can solve the above problem using dynamic programming (DP).

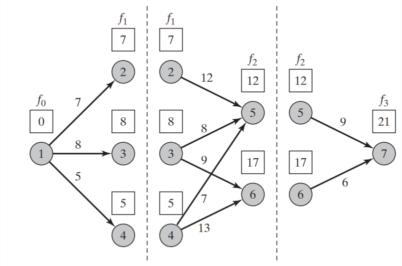
Main idea:

- decompose the problem into smaller ones;
- solve smaller problems easily;
- combine the answers;
- small problems are solved recursively, a problem depends on the solution of another problem.

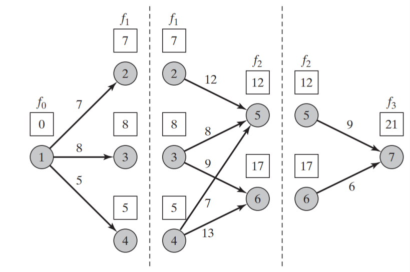
DP for shortest path

Consider the shortest path problem from vertex 1 to vertex 7:





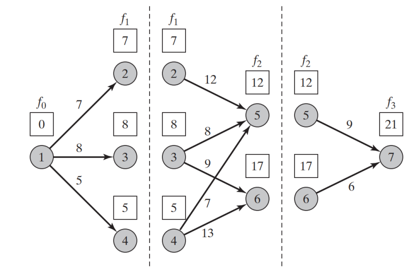
We can solve this problem by dividing it into stages. Each stage has start vertices and end vertices.



Stage 1: compute the shortest distance at end vertices v2, v3 and v4.

The shortest distances are:

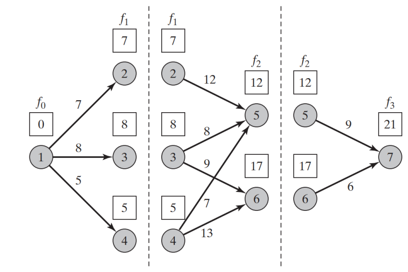
- at v2, 7, from v1;
- at v3, 8, from v1;
- at v4, 5, from v1.



Stage 2: compute the shortest distance at end vertices v5 and v6. We also use the shortest distance at the start vertices v2, v3, v4.

The shortest distances are:

- at v5, 12, from v4;
- at v6, 17, from v3.

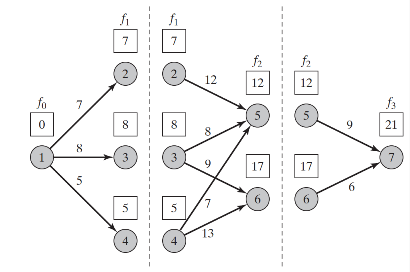


Stage 3: compute the shortest distance at end vertex v7. We also use the shortest distance at the start vertices v5 and v6.

The shortest distances are:

- at v7: 21, from v5.

Summary: the length of the shortest path is 21, and the path is $1 \to 4 \to 5 \to 7$.



To summarize the main ideas, we define

$$f_i(v) = \text{ shortest distance to the vertex } v \text{ at the stage } i$$

E.g., $f_0(1) = 0$, $f_1(2) = 7$, $f_1(3) = 8$ and $f_1(4) = 5$.

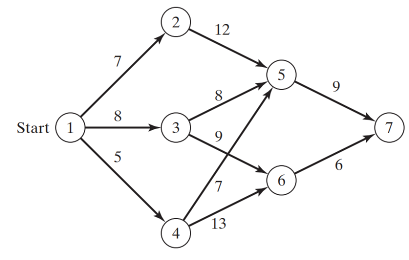
Each f_i depends only on f_{i-1}. That is

$$f_i(v) = \min\left\{d(\tilde{v}, v) + f_{i-1}(\tilde{v}) \mid \forall\, \tilde{v} \to v\right\}.$$

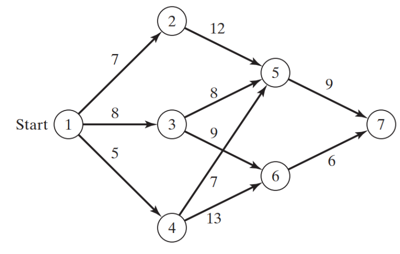
Computations are performed recursively.

Backward recursion

A better way to solve the problem: backward recursion.



We solve the problem from Stage 3, then Stage 2 and Stage 1.

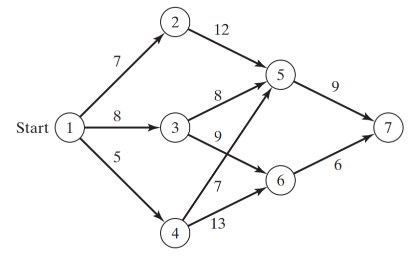


Stage	Vertices
4	7
3	5, 6
2	2, 3, 4
1	1

Main idea of backward recursion: define

$$f_i(v) = \text{shortest distance from the vertex } v \text{ to the target vertex at the stage } i.$$

The target vertex is v7.

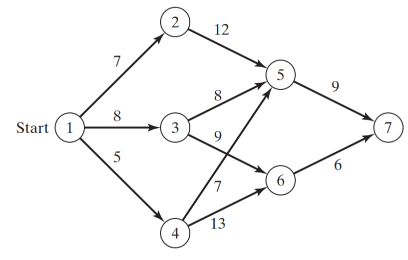


$$f_i(v) = \text{shortest distance from the vertex } v$$
$$\text{to the target vertex at the stage } i.$$

We have

$$f_i(v) = \min\left\{d(v,\tilde{v}) + f_{i+1}(\tilde{v}) \mid \forall\, v \to \tilde{v}\right\}.$$

Note that $f_4(7) = 0$. This is the terminal condition.

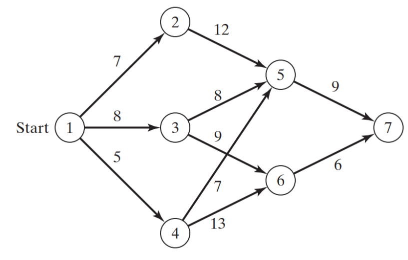


Stage 3: we have

$$f_3(5) = d(5,7) + f_4(7) = 9,$$
$$f_3(6) = d(6,7) + f_7(7) = 6.$$

Stage 2: we have

$$f_2(2) = d(2,5) + f_3(5) = 21(\ 2 \to 5),$$
$$f_2(3) = \min\{d(3,5) + f_3(5),\ d(3,6) + f_3(6)\} = 15(\ 3 \to 6),$$
$$f_2(4) = \min\{d(4,5) + f_3(5),\ d(4,6) + f_3(6)\} = 16(\ 4 \to 5).$$



Stage 2: we have

$$f_2(2) = d(2,5) + f_3(5) = 21(\ 2 \to 5),$$
$$f_2(3) = \min\{d(3,5) + f_3(5),\ d(3,6) + f_3(6)\} = 15(\ 3 \to 6),$$
$$f_2(4) = \min\{d(4,5) + f_3(5),\ d(4,6) + f_3(6)\} = 16(\ 4 \to 5).$$

Stage 1: we have

$$\begin{aligned} f_1(1) &= \min\{d(1,2) + f_2(2),\ d(1,3) + f_2(3),\ d(1,4) + f_2(4)\} \\ &= 21(\ 1 \to 4). \end{aligned}$$

Finally, the path is $1 \to 4 \to 5 \to 7$, and the shortest distance is 21.

Example: Equipment replacement cost

Equipment acquired at start of year	Replacement cost ($) for given years in operation		
	1	*2*	*3*
1	4000	5400	9800
2	4300	6200	8700
3	4800	7100	—
4	4900	—	—

Stage i is the beginning of the i-th year, where $i = 1, 2, 3, 4$, and 5.

Define f_i as the min cost paid at the beginning of year i. So, $f_5 = 0$.

$$f_i = \min \Big\{ (\text{One replacement cost at year } j) \ + f_j \Big\},$$

where the min is taken over all possible replacement strategies available at year j. Our goal is the value of f_1.

Equipment acquired at start of year	Replacement cost ($) for given years in operation		
	1	*2*	*3*
1	4000	5400	9800
2	4300	6200	8700
3	4800	7100	—
4	4900	—	—

Year 4: replace the car 1 year later,

$$f_4 = 4900 + f_5 = 4900.$$

Year 3: replace the car 1 or 2 years later,

$$f_3 = \min\{4800 + f_4, 7100 + f_5\} = 7100 (\text{replace 2 years later}).$$

Year 2: replace the car 1 or 2 or 3 years later,

$$f_2 = \min\{4300 + f_3, 6200 + f_4, 8700 + f_5\} = 8700 (\text{replace 3 years later}).$$

Equipment acquired at start of year	Replacement cost ($) for given years in operation		
	1	*2*	*3*
1	4000	5400	9800
2	4300	6200	8700
3	4800	7100	—
4	4900	—	—

Year 1: replace the car 1 or 2 or 3 years later,

$$f_1 = \min\{4000+f_2, 5400+f_3, 9800+f_4\} = 12500 (\text{replace 2 years later}).$$

Hence, the min cost is 12,500. The strategy is:

- replace the car in Year 3,
- then replace the car in Year 5.

Lesson from the Dijkstra's and DP algorithms: turn a single problem into a set of problems!

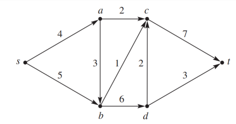
Maximum flow problem

A directed graph G contains a vertex set $V(G)$ and an arc set $A(G)$.

An arc ab is an arrow pointing from a to b.

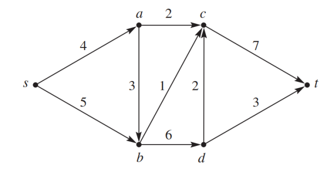
Each arc ab also has a flow capacity u_{ab} ($u_{ab} \geq 0$).

- $V(G) = \{s, a, b, c, d, t\}$;
- $A(G) = \{sa, sb, ab, ac, bc, bd, ct, dc, dt\}$;
- each arc has a positive flow capacity.



Maximum flow problem

Consider the directed graph



Assume that water is injected at vertex s. What is the maximum amount of water received at vertex t?

Initially 9 units of water is injected.

Only 2 units of water can go through the path $s - a - c - t$. Similarly, only 1 unit of water can go through $s - b - c - t$.

Maximum flow problem: mathematical formulation

Let $G(V, E)$ be a graph, and for each edge from u to v, let $c(u, v)$ be the capacity and $f(u, v)$ be the flow. We want to find the maximum flow from the source s to the sink t. We have the following constraints:

- **Capacity constraints**: $\forall (u, v) \in E, f(u, v) \leq c(u, v)$. The flow along an edge cannot exceed its capacity.
- **Skew symmetry**: $\forall (u, v) \in E, f(u, v) = -f(v, u)$. The net flow from u to v must be the opposite of the net flow from v to u.
- **Flow conservation**: $\forall u \in V$ with $u \neq s$ and $u \neq t$, $\sum_{(u,w)\in E} f(u, w) = 0$. The net flow to a node is zero, except for the source, which “produces” flow, and the sink, which “consumes” flow.

We want to maximize

$$V(f) := \sum_{(s,u)\in E} f(s, u) = \sum_{(v,t)\in E} f(v, t).$$

Remark: This is a linear programming problem indeed.

Example: the softball example

For example, a softball team has 15 players, we need to choose 11 players to fill 11 playing positions.

The following table summarizes the positions each player can play.

Al	Bo	Che	Doug	Ella	Fay	Gene	Hal	Ian	John	Kit	Leo	Moe	Ned	Paul
$2, 8$	$1, 5, 7$	$2, 3$	$1, 4, 5, 6, 7$	$3, 8$	$10, 11$	$3, 8, 11$	$2, 4, 9$	$8, 9, 10$	$1, 5, 6, 7$	$8, 9$	$3, 9, 11$	$1, 4, 6, 7$		$9, 10$

Q: Is such assignment possible?

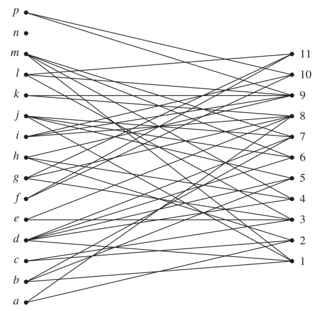
We can formulate this as a directed graph.

Each player and position is a vertex.

An arc, pointing from a player to a position, is formed if that player can play that position.

Al	Bo	Che	Doug	Ella	Fay	Gene	Hal	Ian	John	Kit	Leo	Moe	Ned	Paul
2, 8	1, 5, 7	2, 3	1, 4, 5, 6, 7	3, 8	10, 11	3, 8, 11	2, 4, 9	8, 9, 10	1, 5, 6, 7	8, 9	3, 9, 11	1, 4, 6, 7		9, 10

- Player vertices on left, position vertices on right.
- Inject one unit of water at each vertex on the left, and assume that each vertex on the right can almost receive one unit of water, then find the maximum flow.
- If the maximum flow is 11, then the assignment problem has a solution.



Example: matrix 0-1 problem

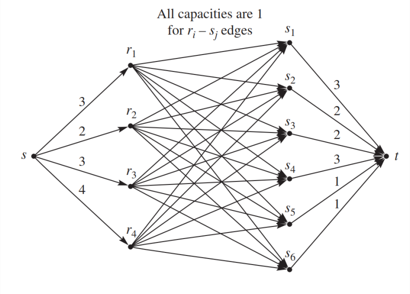
Consider a $m \times n$ $0-1$ matrix (each entry can either be 0 or 1).

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Then

- the row sums are $r_1 = 3$, $r_2 = 2$, $r_3 = 3$, and $r_4 = 4$;
- the column sums are $s_1 = 3$, $s_2 = 2$, $s_3 = 2$, $s_4 = 3$, $s_5 = 1$, and $s_6 = 1$.

The matrix $0-1$ problem is: given m, n and row and column sums, can you find a such $m \times n$ $0-1$ matrix?

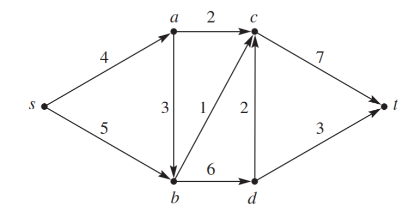


- Row sums are $r_1 = 3$, $r_2 = 2$, $r_3 = 3$, and $r_4 = 4$.
- Column sums are $s_1 = 3$, $s_2 = 2$, $s_3 = 2$, $s_4 = 3$, $s_5 = 1$, and $s_6 = 1$.
- All $r_i - s_j$ edges have capacity 1, the (i, j) entry is 0 or 1.
- If max flow is $3 + 2 + 2 + 3 + 1 + 1$, then solution exists.

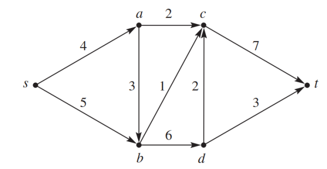
Remark: We will show you a code for solving matrix 0-1 problems on the tutorial section.

Ford and Fulkerson algorithm for maximum flow problems

We illustrate the algorithm by an example.

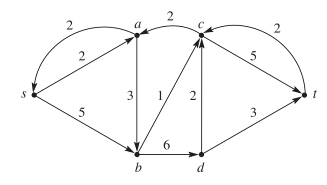


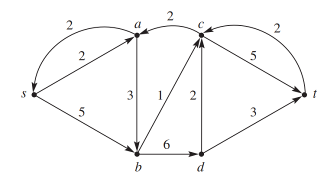
Here is the Ford and Fulkerson algorithm.



Step **1**: choose a path,

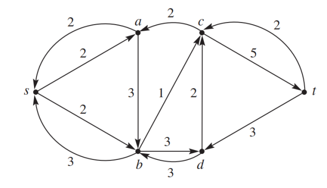
- we choose $s-a-c-t$, the amount of flow is $f=2$,
- no more flow is allowed on edge ac,
- capacities of other edges reduced by 2.

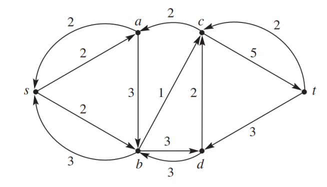




Step **2**: choose a path,

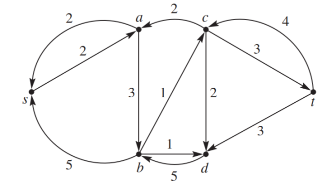
- we choose $s - b - d - t$, the amount of flow is $f = 2 + 3 = 5$,
- no more flow is allowed on edge dt,
- capacities of other edges reduced by 3.

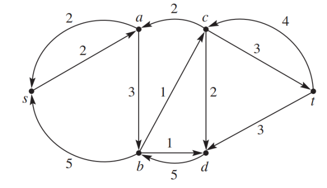




Step 3: choose a path,

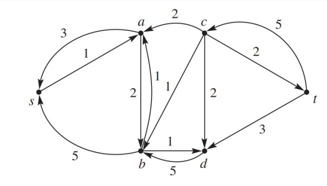
- we choose $s - b - d - c - t$, the amount of flow is $f = 5 + 2 = 7$,
- no more flow is allowed on edge dc and sb,
- capacities of other edges reduced by 2.

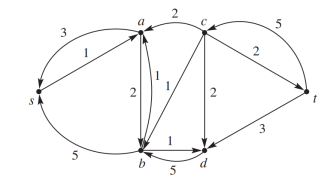




Step 4: choose a path,

- we choose $s - a - b - c - t$, the amount of flow is $f = 7 + 1 = 8$,
- no more flow is allowed on edge bc,
- capacities of other edges reduced by 1.

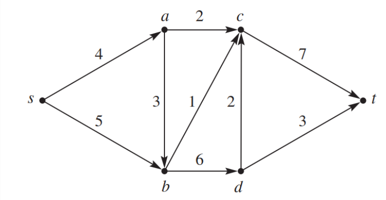




Current flow is $f = 8$.

Note that, no more flow is possible.

The maximum flow of the graph shown on the right is $f = 8$.



An important remark

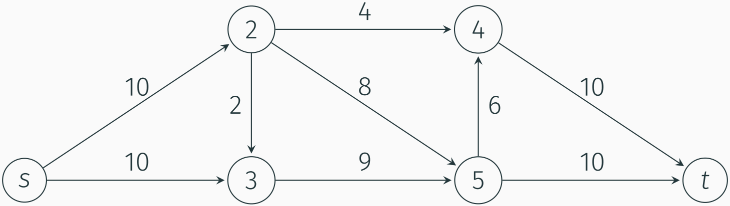
It is clear that, you get a wrong solution if you choose a wrong path.

Remedy: we allow water to flow in reverse direction, and the effect is to restore the flow capacity.

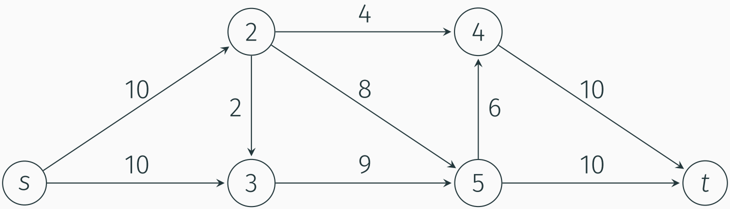
This is pretty confusing. However, just remember that we update constraints on all edges (including reverse arcs) in every step, and we can also use reverse arcs to find a path.

We illustrate this in the next example.

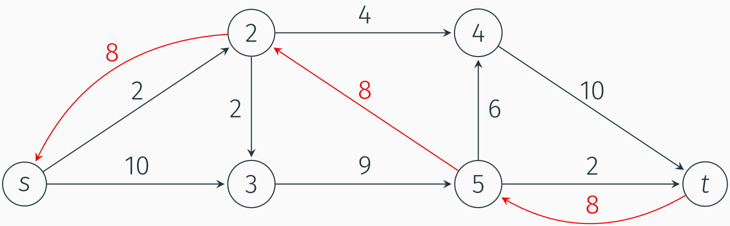
An example with reverse flow

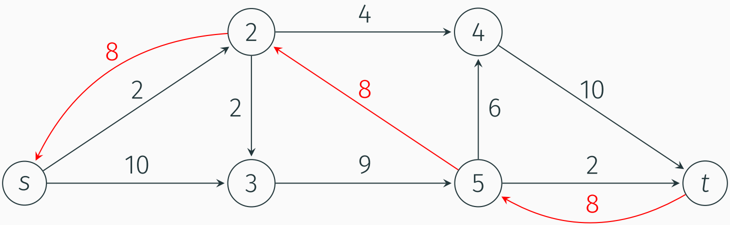


Initially, we set the flow to zero.

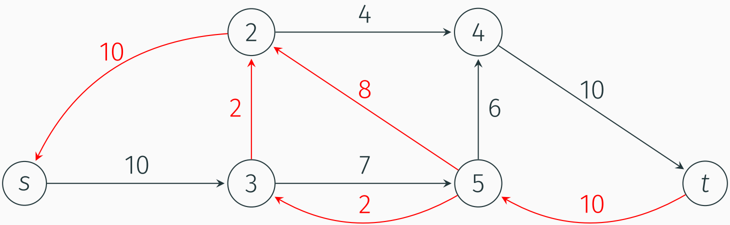


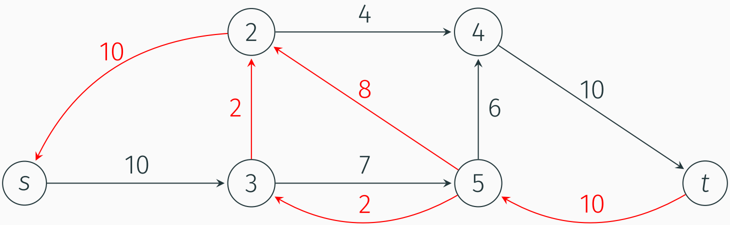
We choose the path $s - 2 - 5 - t$, giving a flow of 8.



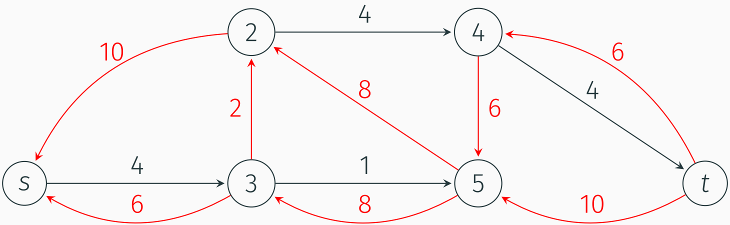


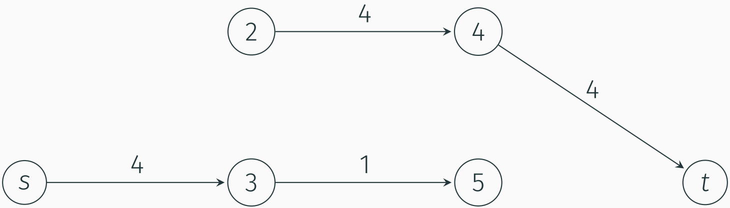
We choose the path $s - 2 - 3 - 5 - t$, giving a flow of 2.



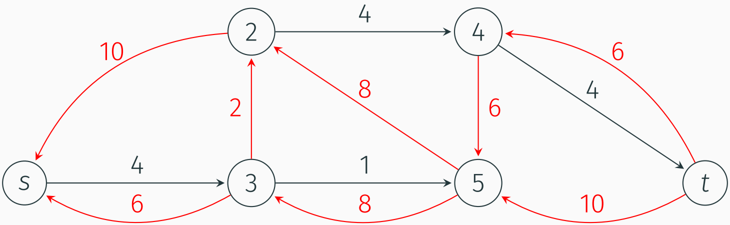


We choose the path $s - 3 - 5 - 4 - t$, giving a flow of 6.





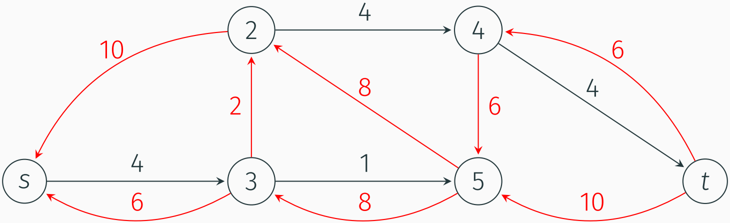
At this stage, it seems we cannot proceed anymore.



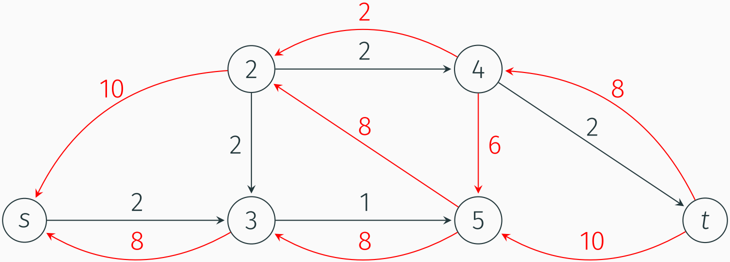
At this stage, it seems we cannot proceed anymore.

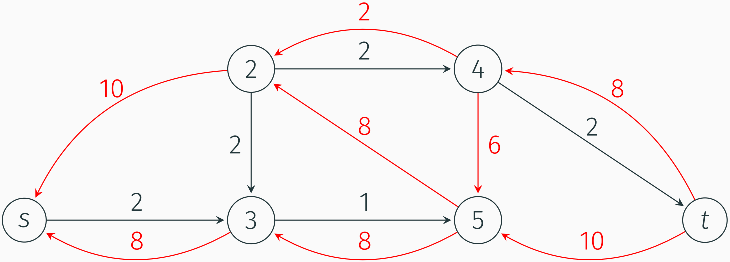
However, if we look at the path $s - 3 - 2 - 4 - t$:

- the arcs $s - 3$, $2 - 4$ and $4 - t$ allow 4 units of flow
- the arc $3 - 2$ allows a flow of 2 units, restoring the flow capacity
- the flow for the path $s - 3 - 2 - 4 - t$ is then 2

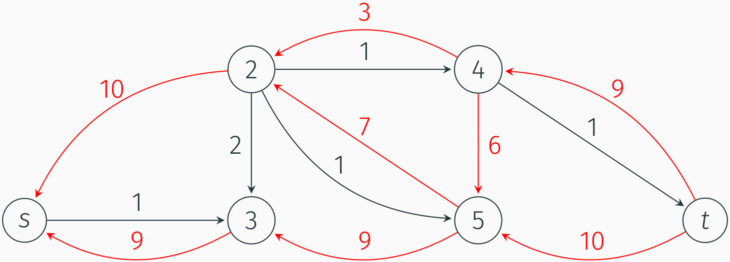


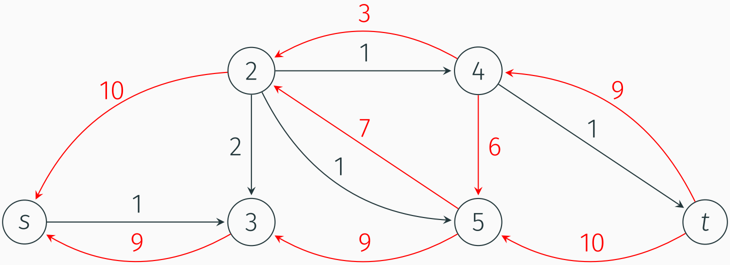
We choose the path $s - 3 - 2 - 4 - t$, giving a flow of 2.





We choose the path $s - 3 - 5 - 2 - 4 - t$, giving a flow of 1.





Finally, we see no more flow is available.

There is arc from vertex s to vertex 3, but no arc from vertex 3 to any other vertex.

Disclaimer

All figures, tables, and data appearing in the slides are only used for teaching under guidelines of Fair Use.