Social Media Analytics

Graph representation and traversal
— SL02 —

Mourad Khayati

mourad.khayati@unifr.ch

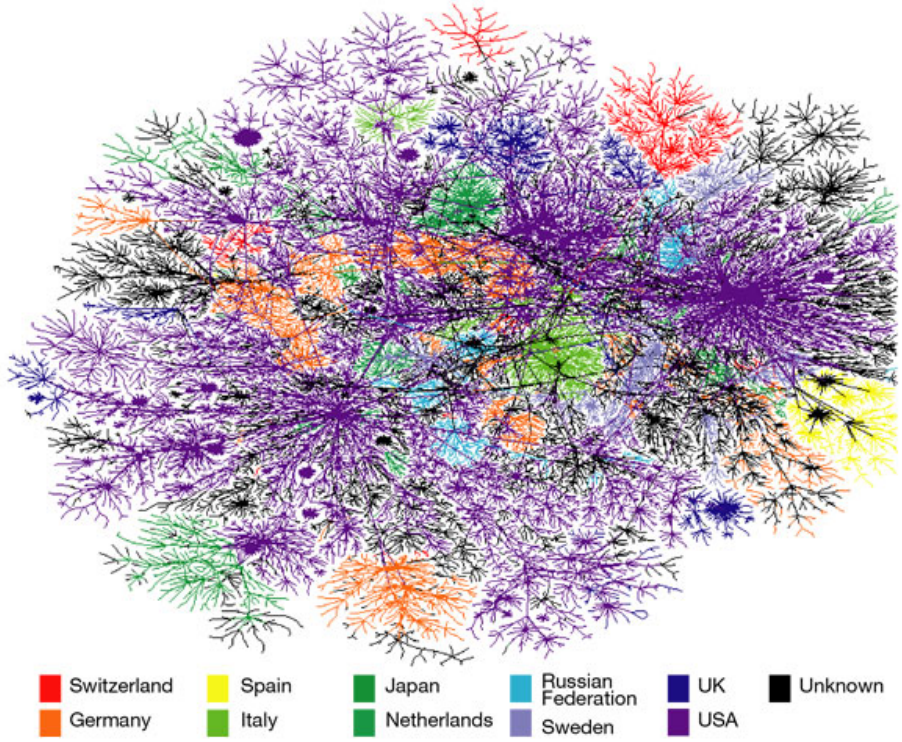
TABLE OF CONTENTS — SL02

What is a network?

- Networks exist in many fields, e.g.,
  - Water/electricity distribution networks.
  - Communication networks: disseminate information.
  - Transportation systems: help to reach a destination.
  - Social networks: individuals interact with each other via friendships, emails, blogposts.
- Networks = nodes, links and (possibly) weights.
- In this course, we will use the terms network and graph interchangeably.

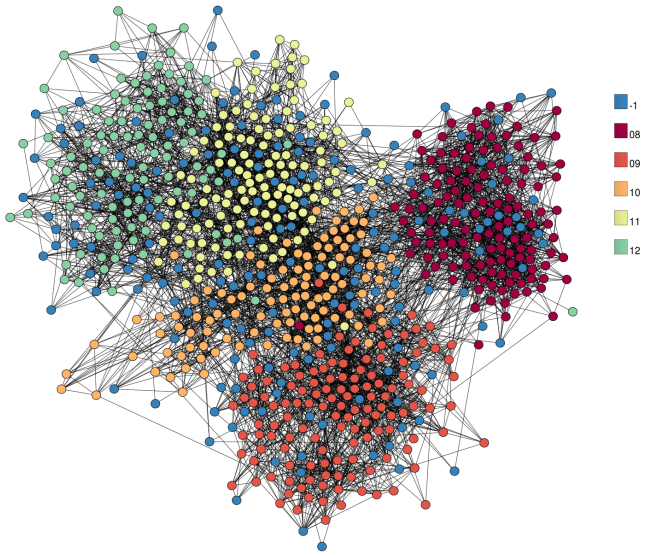
INTERNET

- Internet can be visualized as a graph.
- Network routers are represented as nodes and the paths an e-mail might take across some of the largest networks as edges.

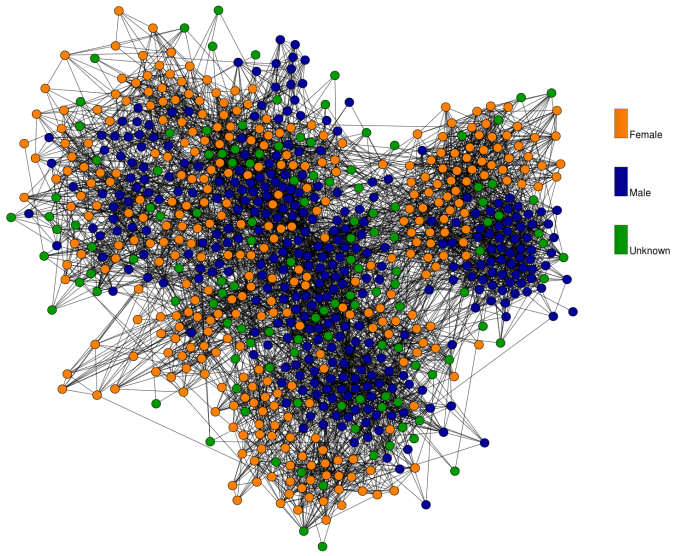


Friendship Network

- The friendship network can be visualized using node-link graphs.
- Each node is a person and each line indicates a friendship.



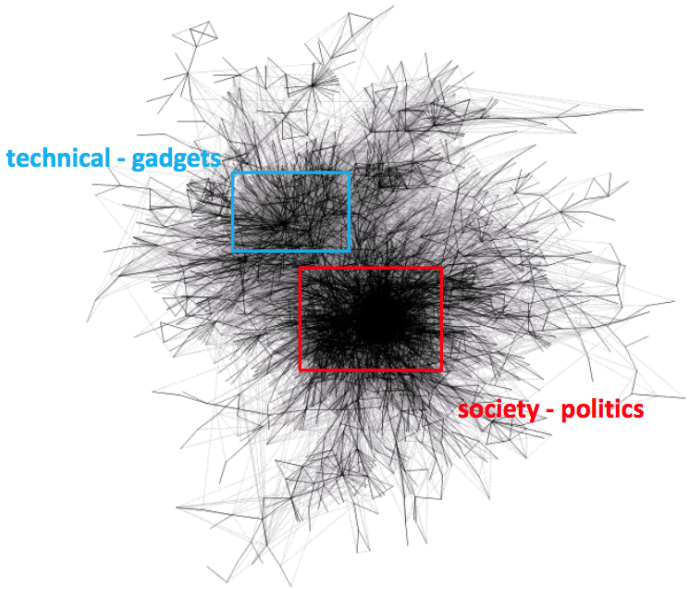
By grade



By gender

BLOGOSPHERE

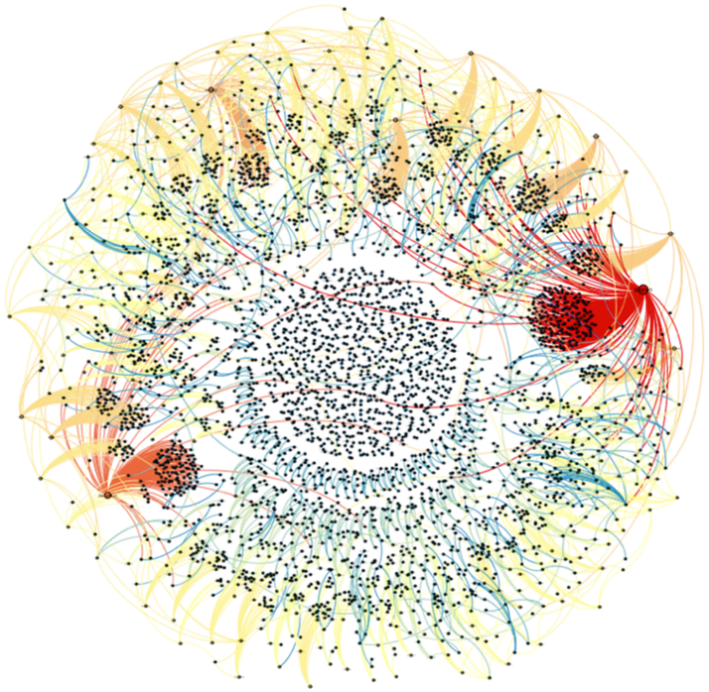
- The dark edges show the reciprocal links (where A has cited B and B has cited A).
- The larger, denser area of the graph is that part of the blogosphere generally characterized by socio-political discussion.



http://datamining.typepad.com/gallery/blog-map-gallery.html

TWITTER

- The following graph represents the number of tweets after the announcement of Mubarak's resignation in Egypt.
- The nodes represent the twitter users and the edges represent the retweets containing the #jan25 hashtag.



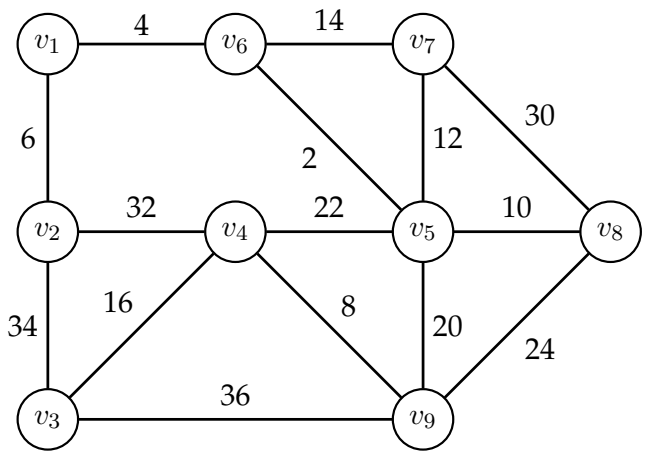
https://datavisualization.ch/showcases/egyptian-revolution

Network Problems

- Networks problems can be solved using graph representations.
- Representing a problem as a graph can make a problem much simpler.

Twitter example:

Given a piece of information, a network of individuals, and the cost to propagate information among any connected pair, find the minimum cost to disseminate the information to all individuals.

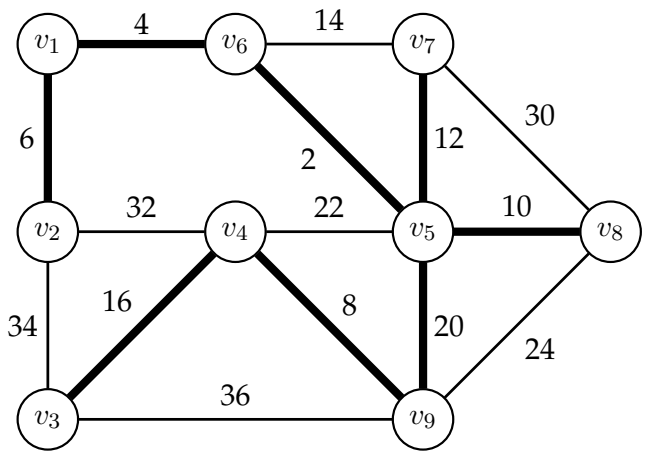


Network Problems

- Networks problems can be solved using graph representations.
- Representing a problem as a graph can make a problem much simpler.

Twitter example:

Given a piece of information, a network of individuals, and the cost to propagate information among any connected pair, find the minimum cost to disseminate the information to all individuals.



Network exploration: Application

Empirical and hypothesis-free observation of a network.

- Exploration is the first step of a scientific network analytics project.
- Exploration can lead to discover interesting research problems
- Many graphical tools have been proposed to perform network exploration

NETWORK EXPLORATION: TOOLS

- *yED* (available for free at yworks.com)
  - \+ Best suited for creating small networks and for layouting graphs
  - – slow
- *Gephi*: visualization and analysis of medium to large networks and of dynamic networks
  - \+ very good layout algorithms
  - – difficult to handle
- *R*: Open-source statistics environment with dedicated network analysis packages (igraph, statnet, sna)
  - \+ very good for mathematical analysis of large networks (not very large, though)
  - – bad visualization

Network exploration: Datasets

- **M.E. Newman**: `http://www-personal.umich.edu/~mejn/netdata/`
- **Jure Leskovec**: `http://snap.stanford.edu/data/`
- **Gephi data sets**: `https://github.com/gephi/gephi/wiki/Datasets`
- **Pajek**: `http://vlado.fmf.uni-lj.si/pub/networks/data/`

Table of contents — SL02

Graphs: Nodes & Edges

- **Graph**: a graph $\mathcal{G}$ consists of a set of vertices (nodes) V and a set of edges E i.e., $\mathcal{G} = (V, E)$.
- **Nodes**: A set of n vertices is denoted as $V = \{v_1, v_2, \ldots, v_n\}$ and $|V| = n$ is the *size* of the graph.
- **Edges**: A set of m edges is denoted as $E = \{e_1, e_2, \ldots, e_m\}$ and $|E| = m$ is the *order* of the graph.
- In an acyclic graph the following holds:

$$|V| \geq |E| + 1$$

Directed and undirected graphs

Directed graph:

- Edges have an orientation.
- Example: Twitter. If A follows B, then B does not have to follow A.
- The representation of a directed graph is not symmetric.

Undirected graph:

- Edges do not have any orientation.
- Example: Facebook. If A is friends with (in a relationship with or related to) B, then B must be friends with A.
- The representation of an undirected graph is symmetric.

Node Degree: definition

A degree of node v_i is denoted $d(v_i)$ and represents the number of edges connected to v_i. There exists two types of node degrees:

- $d^i(v_i)$: is the number of edges pointing towards v_i
- $d^o(v_i)$: is the number of edges pointing away from v_i

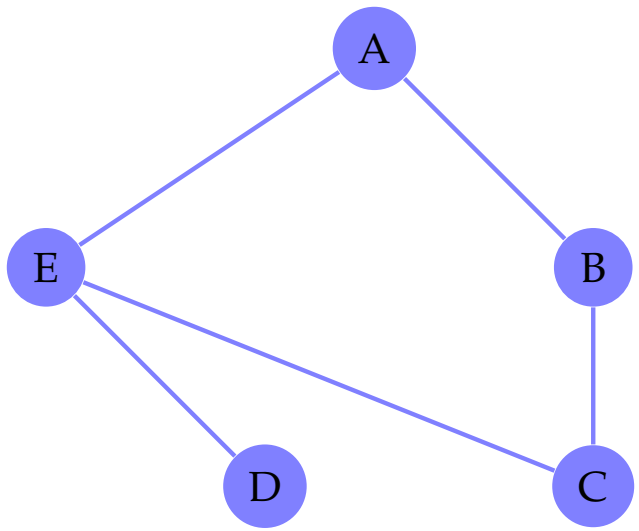
Theorem (Degree Summation)

The summation of degrees in an (undirected) graph is twice the number of edges:

$$\sum_{v_i \in V} d(v_i) = 2 \times m$$

Node Degree: example

- Consider the following graph that has five nodes and five edges:



$$\sum_{v_i \in V} d(v_i) = 2 + 2 + 2 + 3 + 1 = 10$$

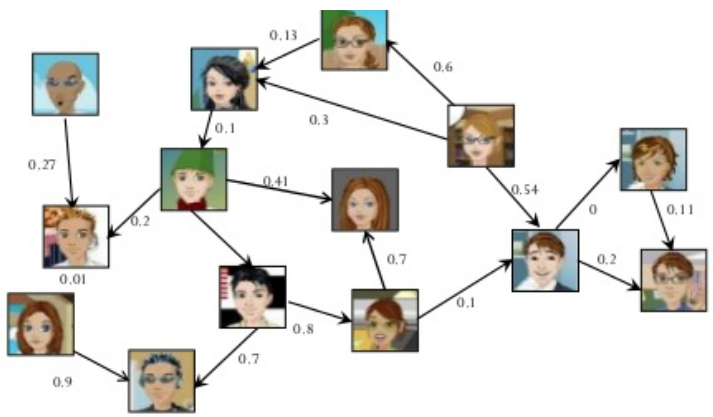
$$= 2 \times m$$

Null and Empty Graph

- A null graph is a graph where the node set is empty, i.e., $\mathcal{G}(V, E), V = E = \emptyset$.
- An empty or edgeless graph is one where the edge set is empty, i.e., $\mathcal{G}(V, E), E = \emptyset$.
- A null graph is an empty graph.

Weighted Graph

- In a weighted graph $\mathcal{G}(E, V, W)$ edges are associated with weights (W)
- Example: Facebook
  - Nodes are users, edges are relationships between users and
  - Weights are the degree of friendship between these users.

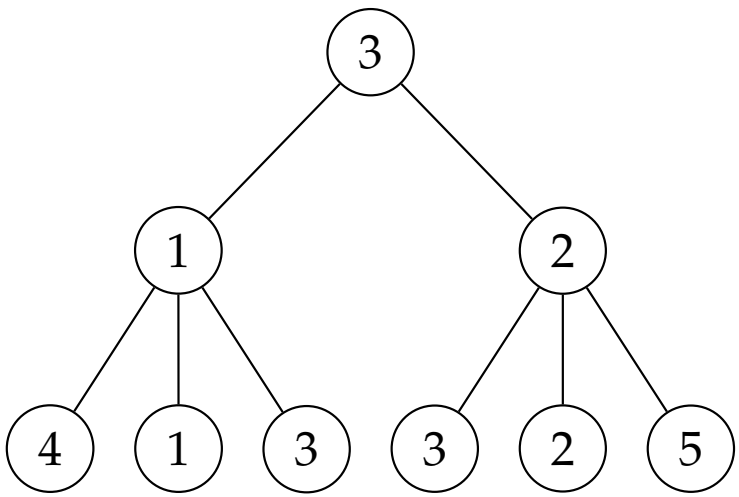


WEBGRAPH

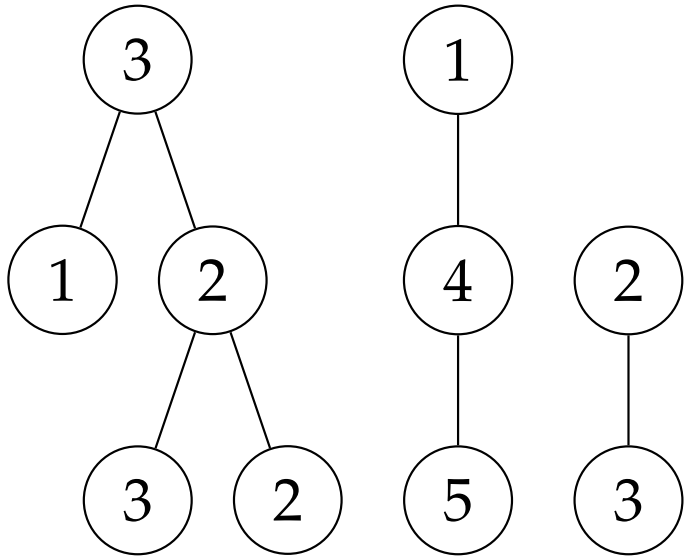
- A webgraph is a directed graph representing how Internet sites are connected on the web.
- Each node is a URL and an edge from x to y implies that page x contains a hyperlink toward page y.
- Two sites can have multiple links pointing to each other and can have loops (links pointing to themselves).

Trees and Forests

- A tree is an undirected graph structure that has no cycle: there is exactly one path between any pair of nodes
- In a tree we have $|V| = |E| + 1$.
- A graph consisting of set of disconnected trees is called a forest.



Tree



Forest

EXERCISE

1. Is it possible to construct an acyclic and non empty graph where $\forall v_i \in V : d(v_i) \geq k \geq 2$?
2. Prove your answer. Hint: in an acyclic graph we have: $|V| \geq |E| + 1$.

Representation of Social Graphs

- Social media networks involve a sheer number of users, e.g., Facebook (2.2B), WhatsApp(1.5B), Instagram (800M), Twitter (330M) (`https://www.statista.com`).
- What is the *best* way to *represent* social media networks: graphs? matrices? arrays? etc.
- How to quantify *best* and how to interpret *represent*?

Representation challenges

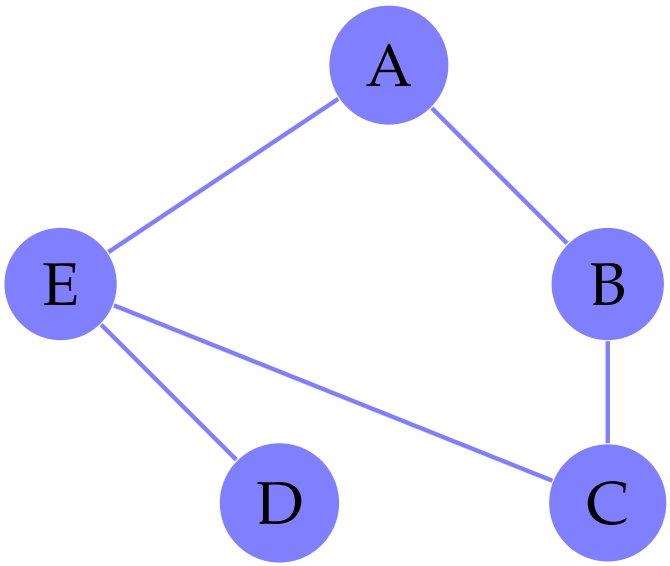
- Graph representation is intuitive but it can not be easily used by mathematical and computational tools.
- We seek for representations that can store the node and edge sets in a way that:
  - does not loose information
  - can be easily manipulated
  - can have mathematical methods applied easily

Representation Types

- Common graph search operations:
  - Retrieving all edges incident to a particular node
  - Checking if given two nodes are directly connected
- The most common graph representations are: i) edge list, ii) adjacency list and iii) adjacency matrix.

Edge list

- In this representation, each node v_i connected to a node v_j is represented as (v_i, v_j).



(A,B)
(A,E)
(B,C)
(E,C)
(E,D)

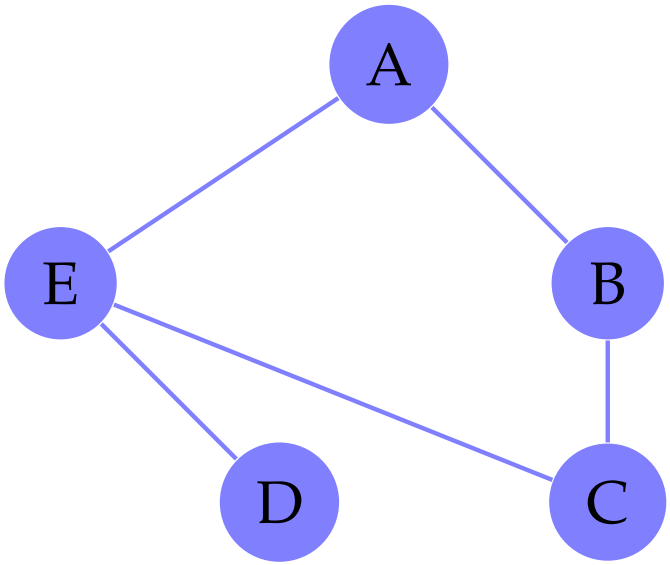
EDGE LIST / 2

- Memory efficient, space complexity is $O(m)$.
- Graph operations are fast because the number of edges is equal to the length of list.
- Slow in case we want to find if v_i and v_j are adjacent.

$\Rightarrow$ Better (i.e., fewer edges) for sparse graphs.

Adjacency list/1

- In adjacency list, we maintain for every node a list of all the nodes that it is connected to.
- The list is usually sorted based on the node order or other preferences.



Node	Connected to
A	B, E
B	A, C
C	B, E
D	E
E	A, C, D

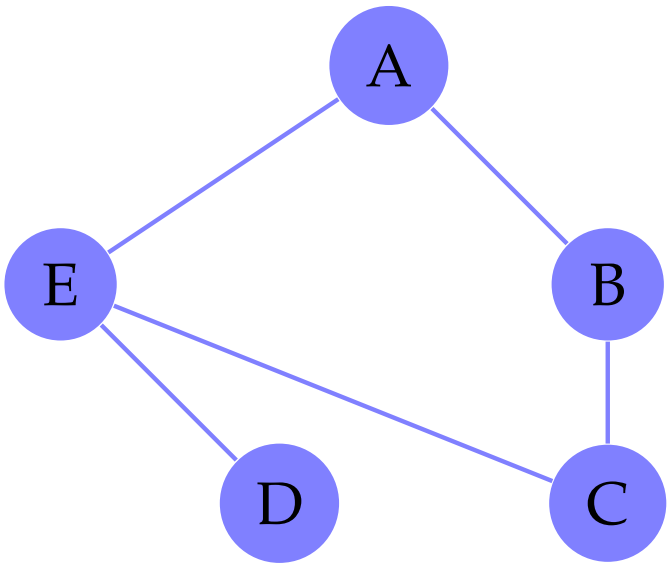
Adjacency list / 2

- More overhead than edge list, uses space $O(n + m)$.
- Very fast addition of edges in time $O(1)$.
- Deletion of edges in time $O(n)$.

$\Rightarrow$ Better for sparse graphs.

Adjacency matrix/1

- The adjacency matrix gives a natural mathematical representation for graphs.
- A value of 1 in the adjacency matrix indicates a connection between nodes v_i and v_j, 0 indicates no connection.



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Adjacency matrix/2

- High memory requirement, uses space $O(n^2)$.
- Iterating over all edges requires time $O(n^2)$.
- Fast random access to insert or delete or query edges: time complexity $O(1)$ in case v_i and v_j are adjacent.

$\Rightarrow$ Better for dense (i.e., lots of edges) graphs.

Exercise

- Write the adjacency matrix of the following graph.
- What's the most space-efficient representation to store this graph and why?

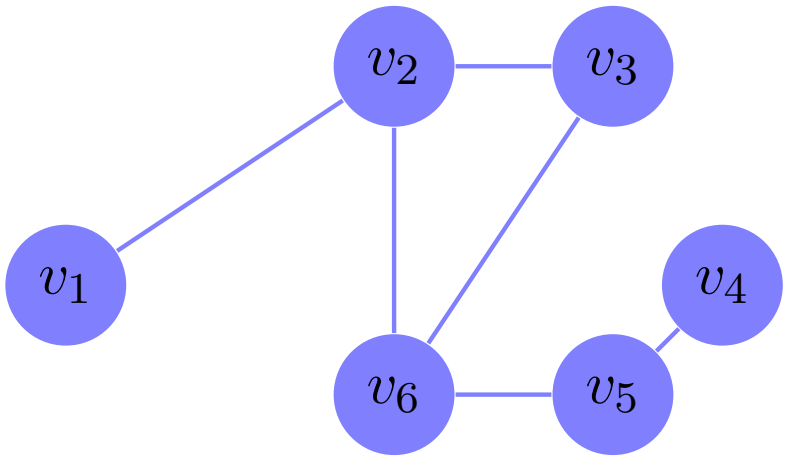


TABLE OF CONTENTS — SL02

Graph Traversal: Definition

- Graph traversal algorithms are used to visit the nodes of a graph such that:
  - all (or a set of) the nodes are visited
  - no node is visited more than once
- These algorithms specify an order to search through the nodes of the graph.
- They start at the source node and keep searching until either finding the target node or reaching a termination condition.

Graph Traversal: Application

- Web Crawlers: analyze which sites you can reach by following links on a particular website.
- Linkedin surveys: perform surveys starting from one user and then browsing his/her friends and then these friends' friends and so on.
- Disseminate (quickly) information on social media.
- Other: GPS Navigation systems (e.g., Google Maps).

Graph Traversal: Algorithms

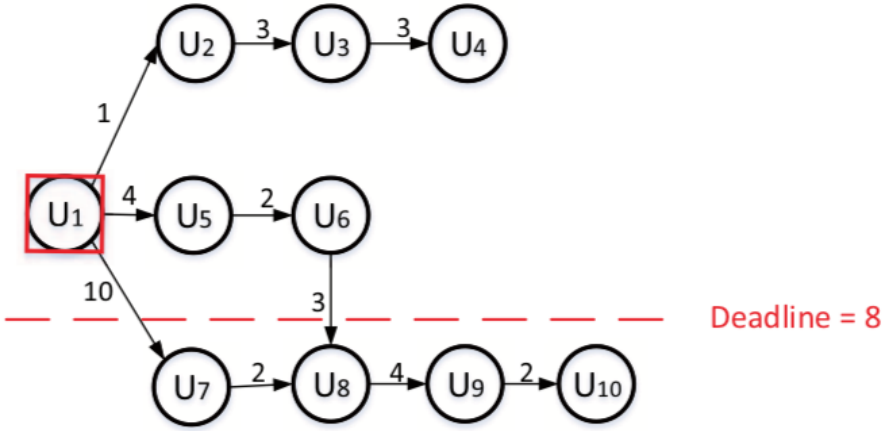
- There exist two of types of graph traversal algorithms:
  - Blind (uninformed) search:
    - Depth-First Search (DFS)
    - Breadth-First Search (BFS)
  - Informed search:
    - Greedy (Best First) Search
    - A/A^* Search

Depth-First Search (DFS)

- Depth-first search (DFS) strategy works as follows:
  1. start from a node v_i
  2. select one of v_i's neighbors $v_j \in N(v_i)$
  3. perform DFS on v_j before visiting other neighbors in $N(v_i)$
- DFS uses a stack structure to visit nodes in a depth-first fashion.
- The worst runtime complexity of DFS is $O(|V| + |E|)$.

DFS Application

- *Temporal Influence Blocking: Minimizing the Effect of Misinformation in Social Networks*, ICDE'17
- The proposed technique tackles the problem of limiting the spread of rumor on social networks.



- DFS algorithm is applied to find the nodes that could be reached by a rumor starter R within t hops.

BREADTH-FIRST SEARCH (BFS)

- Breadth-first search (BFS) strategy works as follows
  1. start from a node v_i
  2. visit all v_i's direct neighbors
  3. move to the second level by traversing their neighbors
- BFS uses a queue structure to perform a breadth traversal.
- The worst runtime complexity of BFS is $O(|V| + |E|)$.

BFS Application

- The `https://oracleofbacon.org` website finds the degrees of separation between an actor A and an actor B.

- The website maintains an undirected graph on which each vertex corresponds to an actor, then links two vertices through an edge if two people appeared in the same film.
- BFS from the vertex for an actor A finds the shortest path to all other actors and actresses.

Greedy (Best First) Search

- The Greedy (Best First) Search uses a heuristic $h(n)$ to determine which nodes to visit. $h(n)$ approximates the cost to go from a node v_i to the goal.
- Greedy Search works as follows
  1. start from a node v_i
  2. visit v_i's direct neighbor having the minimum cost $h(n)$
  3. move to the second level by traversing their neighbors
- The worst runtime complexity is $O(b^m)$ where b is the branching factor and m the maximal depth.

Greedy Search Algorithm

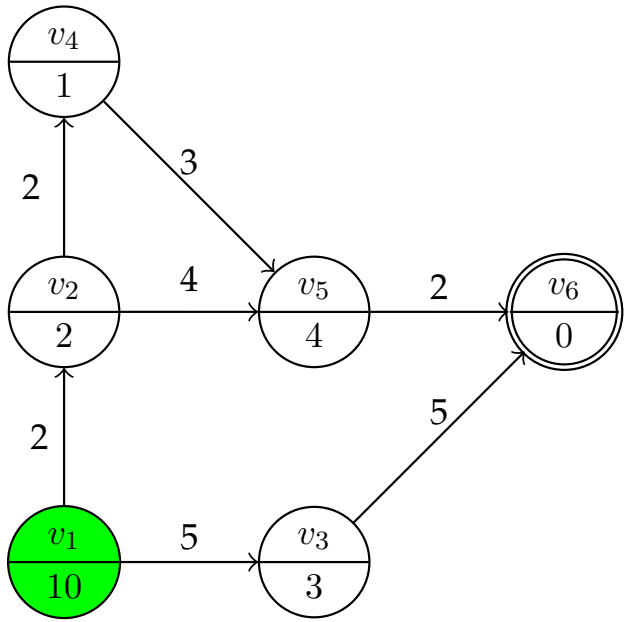
Algo: Greedy Search

input : v_i, // initial node
$\mathcal{G}(V, E)$, // input graph/tree
Q // Queue
output: An ordering set of visited nodes
Enqueue v_i into Q;
while $Q \neq \emptyset$ **do**
  Pick the path P with minimum $h(head(P))$ from Q;
  if $head(P)$= goal **then**
    return P;
  else
    foreach v_j such that $(head(P), v_j) \in E$ **do**
      Enqueue (v_j, P) into Q;
return FAILURE

Greedy Search Example/1

Q:

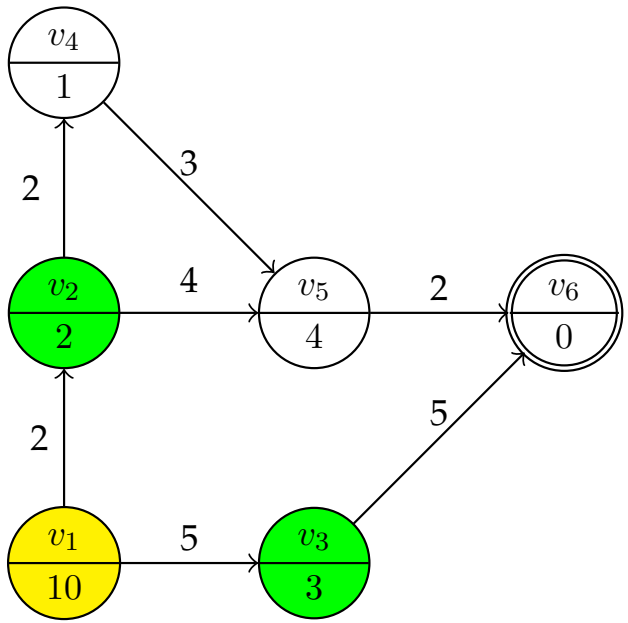
$path$	$w(P)$	h
$\langle v_1 \rangle$	0	10



Greedy Search Example/2

Q:

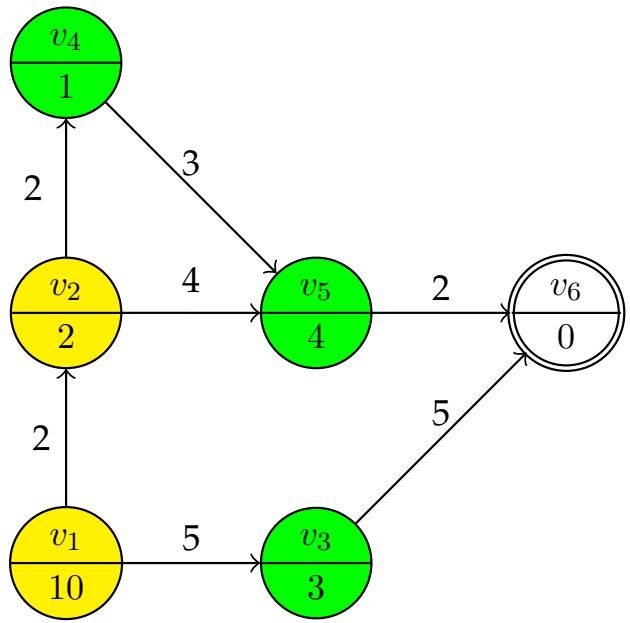
$path$	$w(P)$	h
$\langle v_2, v_1 \rangle$	2	2
$\langle v_3, v_1 \rangle$	5	3



Greedy Search Example/3

Q:

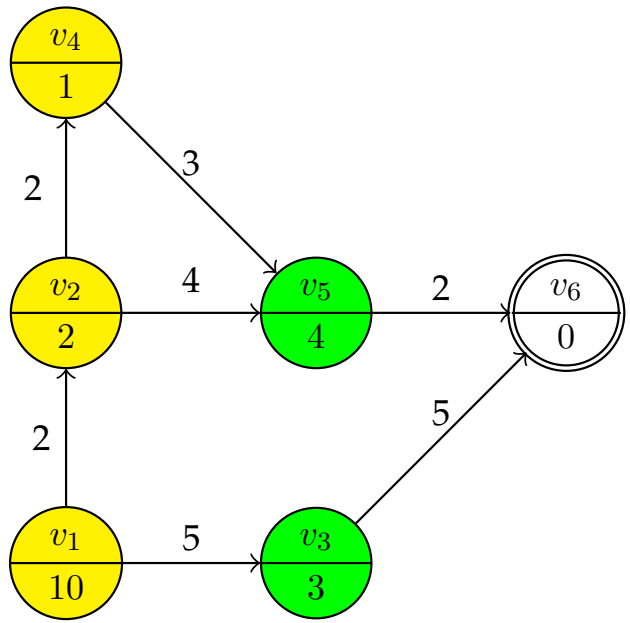
$path$	$w(P)$	h
$\langle v_4, v_2, v_1 \rangle$	4	1
$\langle v_3, v_1 \rangle$	5	3
$\langle v_5, v_2, v_1 \rangle$	6	4



Greedy Search Example/4

Q:

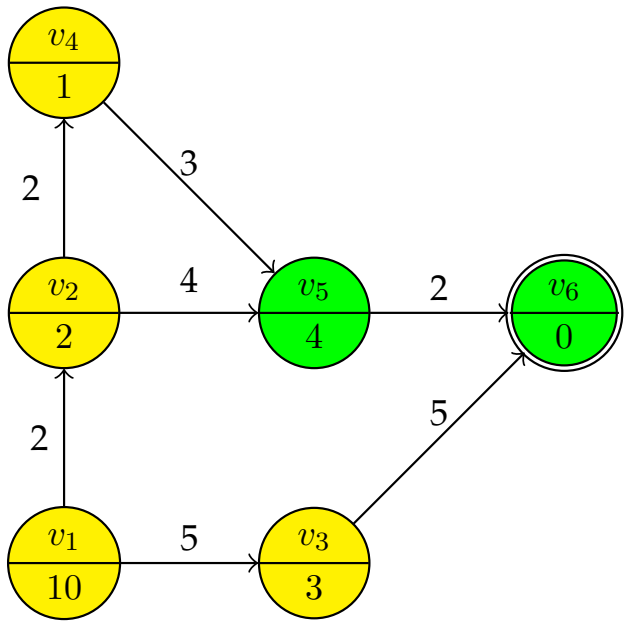
$path$	$w(P)$	h
$\langle v_3, v_1 \rangle$	5	3
$\langle v_5, v_2, v_1 \rangle$	6	4
$\langle v_5, v_4, v_2, v_1 \rangle$	7	4



Greedy Search Example/5

Q:

$path$	$w(P)$	h
$\langle v_6, v_3, v_1 \rangle$	10	0
$\langle v_5, v_2, v_1 \rangle$	6	4
$\langle v_5, v_4, v_2, v_1 \rangle$	7	4



A Search: Idea/1

- Greedy search is not optimal as it is heavily biased towards moving towards the goal.
- The non-optimality comes from neglecting "the past".

Idea:

- Keep track both of the cost of the partial path to get to a node, i.e., $g(v)$, and of the heuristic function estimating the cost to reach the goal from a node, $h(v)$.

A Search: Idea/2

- A search combines the strengths of Breadth First Search and Greedy Best First.
- Like BFS, it finds the **shortest path**, and like Greedy Best First, it is **fast**.
- Each iteration, A search chooses the node on the frontier which minimizes:

$$\underbrace{\text{steps from source}}_{\text{Like BFS, nodes close to source first}} + \underbrace{\text{approximate steps to target}}_{\text{Like Greedy, uses heuristic to prioritize nodes closer to target}}$$

A Search: Procedure

- We choose as a "ranking" function the sum of the two costs:

$$f(v) = g(v) + h(v)$$

  - $g(v)$: cost-to-come (from the start to v).
  - $h(v)$: cost-to-go estimate (from v to the goal).
  - $f(v)$: estimated cost of the path (from the start to v and then to the goal).

A Search Algorithm

Algo: A Search

input : v_i, // initial node
$\mathcal{G}(V, E)$, // input graph/tree
Q // Queue

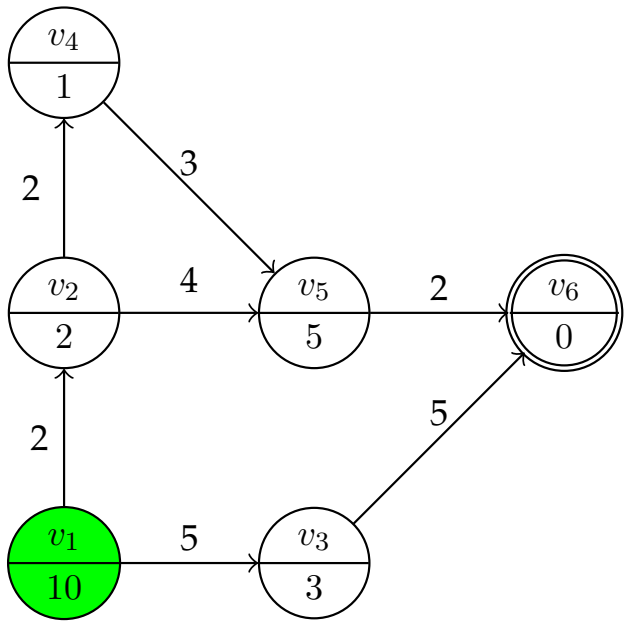
output: An ordering set of visited nodes

```
Enqueue v_i into Q;
while Q ≠ ∅ do
    Pick the path P with minimum cost
     f(P) = g(P) + h(head(P)) from Q;
    if head(P)= goal then
        return P;
    else
        foreach v_j such that (head(P), v_j) ∈ E do
            Enqueue (v_j, P) into Q;
return FAILURE;
```

A Search Example/1

Q:

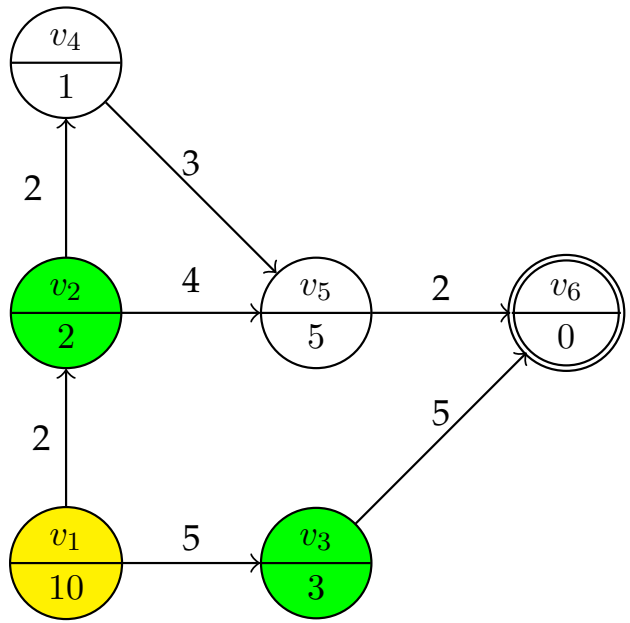
$path$	g	h	f
$\langle v_1 \rangle$	0	10	10



A Search Example/2

Q:

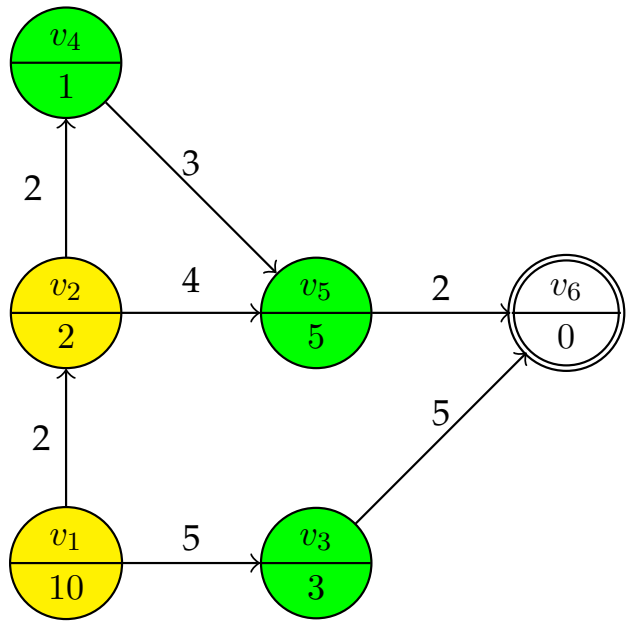
$path$	g	h	f
$\langle v_2, v_1 \rangle$	2	2	4
$\langle v_3, v_1 \rangle$	5	3	8



A Search Example/3

Q:

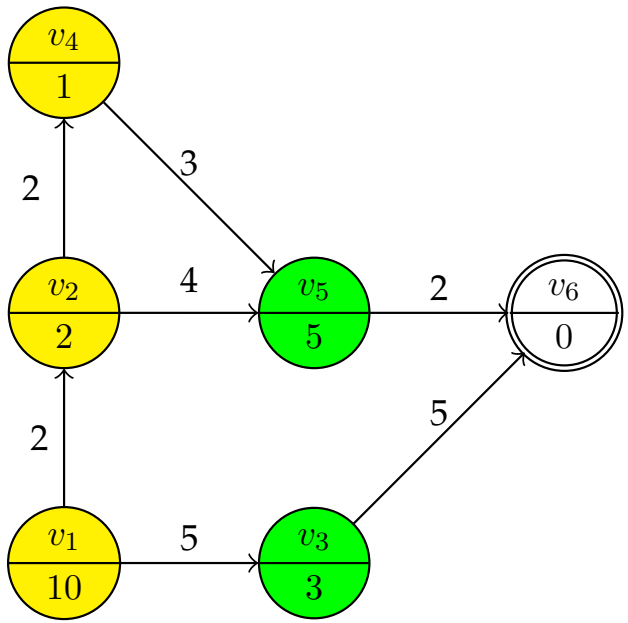
$path$	g	h	f
$\langle v_4, v_2, v_1 \rangle$	4	1	5
$\langle v_3, v_1 \rangle$	5	3	8
$\langle v_5, v_2, v_1 \rangle$	6	5	11



A Search Example/4

Q:

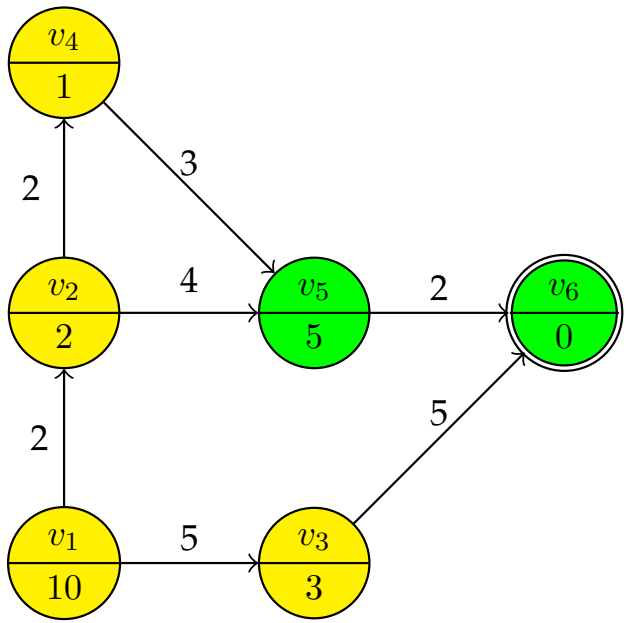
$path$	g	h	f
$\langle v_3, v_1 \rangle$	5	3	8
$\langle v_5, v_2, v_1 \rangle$	6	5	11
$\langle v_5, v_4, v_2, v_1 \rangle$	7	5	12



A Search Example/5

Q:

$path$	g	h	f
$\langle v_6, v_3, v_1 \rangle$	10	0	10
$\langle v_5, v_2, v_1 \rangle$	6	5	11
$\langle v_5, v_4, v_2, v_1 \rangle$	7	5	12



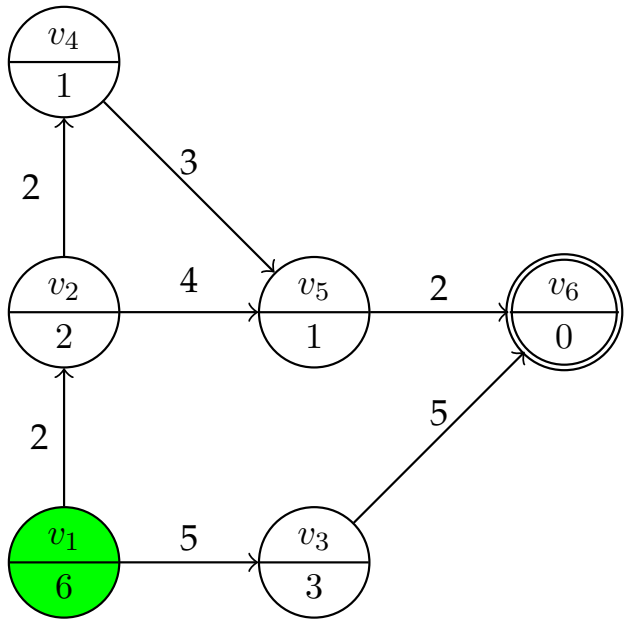
$A*$ Search: Idea

- A search is not optimal because of the random heuristic.
- The optimal solution requires the use of an admissible heuristic. An admissible heuristic does not overestimate the cost-to-go.
- The distance (Euclidean, physical, etc.) between nodes is often used as an admissible heuristic, i.e., $h(v) = dist(v, g)$.
- The A search with an admissible heuristic is called A^* search, which is guaranteed to be optimal.

A^* Search Example/1

Q:

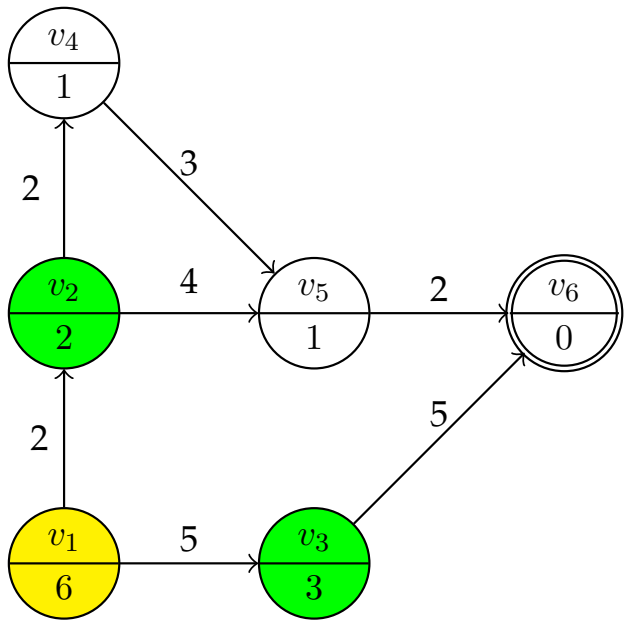
$path$	g	h	f
$\langle v_1 \rangle$	0	6	6



A^* Search Example/2

Q:

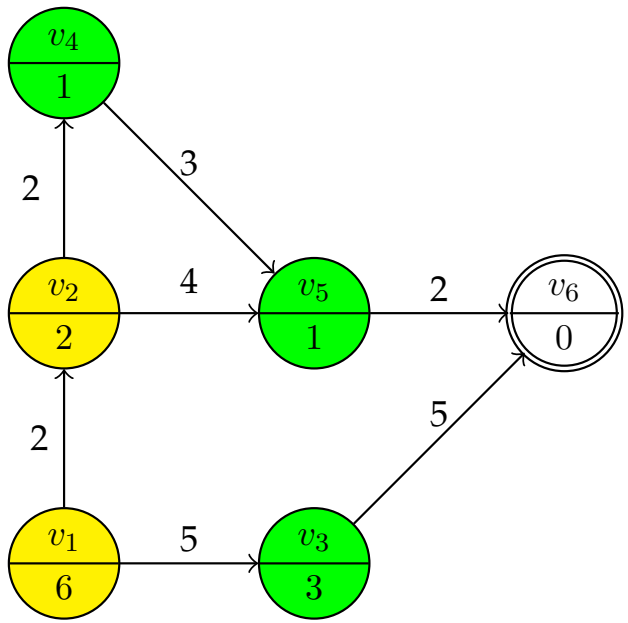
$path$	g	h	f
$\langle v_2, v_1 \rangle$	2	2	4
$\langle v_3, v_1 \rangle$	5	3	8



A^* Search Example/3

Q:

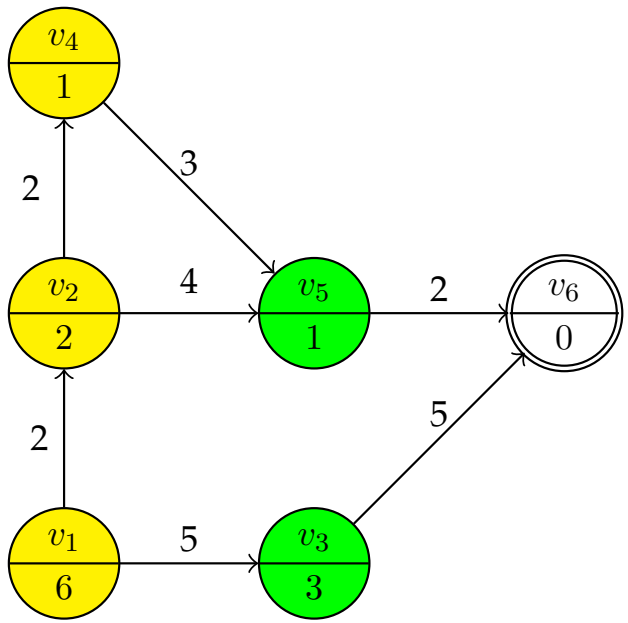
$path$	g	h	f
$\langle v_4, v_2, v_1 \rangle$	4	1	5
$\langle v_5, v_2, v_1 \rangle$	6	1	7
$\langle v_3, v_1 \rangle$	5	3	8



A^* Search Example/4

Q:

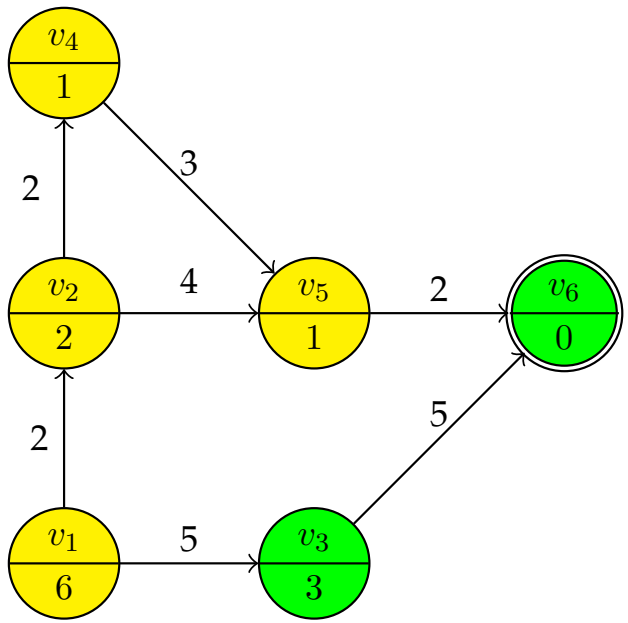
$path$	g	h	f
$\langle v_5, v_2, v_1 \rangle$	6	1	7
$\langle v_3, v_1 \rangle$	5	3	8
$\langle v_5, v_4, v_2, v_1 \rangle$	7	1	8



A^* Search Example/5

Q:

$path$	g	h	f
$\langle v_6, v_5, v_2, v_1 \rangle$	8	0	8
$\langle v_3, v_1 \rangle$	5	3	8
$\langle v_5, v_4, v_2, v_1 \rangle$	7	1	8



Random Walk: Definition

- A walking graph algorithm takes as input a starting node and selects a neighbor at random.
- Then, the algorithm moves to the selected neighbor and repeats the same process until a termination condition is verified.
- The random sequence of nodes selected in this way is called a random walk of the graph.

Random Walk in Weighted Graph

- In a weighted graph, the weight of an edge can be used to define the probability of visiting it.
- For all edges that start at v_i the following equality holds

$$\sum_x w_{i,x} = 1, \forall i, j \text{ with } w_{ij} \geq 0$$

Random Walk: Algorithm

Algo: Random Walk

input : v_0, // initial node
$\mathcal{G}(V, E)$, // Weighted graph
t // Steps

output: Random Walk P

state =0;

$v_t = v_0$;

$P = \{v_0\}$;

while $state < t$ **do**

  $state = state + 1$;

  Select a random v_j adjacent go v_t with probability $w_{t,j}$;

  $v_t = v_j$;

  $P = P \cup \{v_j\}$;

return P

Random Walk: Application/1

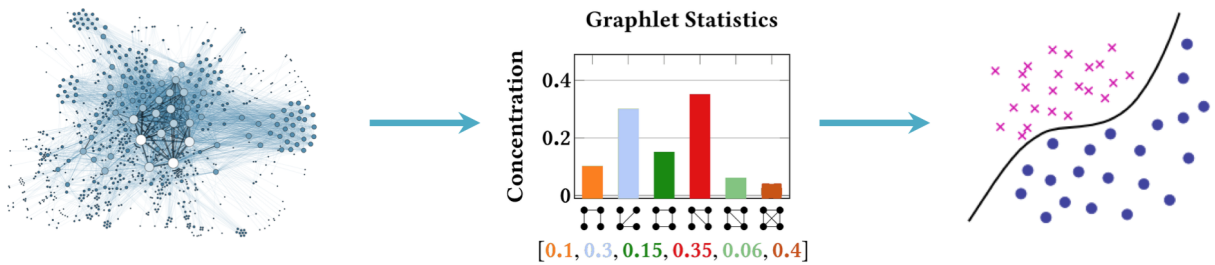
Random walks are applied in social networks to perform:

- Sampling: Obtain a representative sample of users by exploration of the social graph.
- Link prediction: Predict what new edges (friendships) a user will create between timestamp t and some future timestamp t' with $t' > t$.
- Connection recommendation: suggest to each user a list of people that the user is likely to create new connections to.

Random Walk: Application/2

- *A General Framework for Estimating Graphlet Statistics via Random Walk*, published in the Very Large Databases conference (VLDB'16).

- The proposed technique counts the number of subgraph patterns (Graphlet).



- Random walk is applied as a sampling algorithm to estimate the number of Graphlets in a social network.