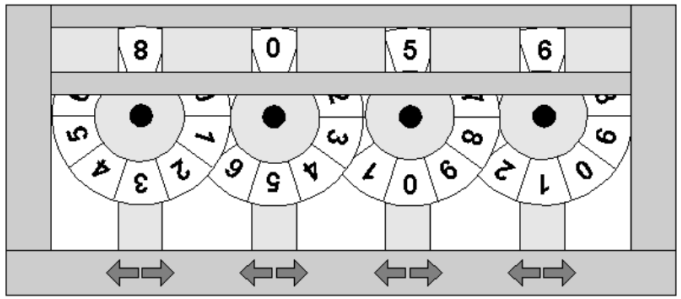
Modeling With Graphs

Things to decide –

- what the vertices represent
  - vertices represent things
- what the edges represent
  - edges represent connections (relationships) between pairs of things
    - does the relationship simply exist or not exist? – indicated by presence or absence of edge
    - does the relationship have a strength? – modeled by edge weights
    - is the relationship one-way? – indicated by directed or undirected edges
- what a solution to the original problem looks like in the graph
  - ideally it is a concept for which there is a known algorithm
    - e.g. reachability, shortest path, minimum spanning tree, cut vertices

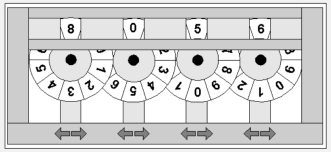
In this problem we will be considering a game played with four wheels. Digits ranging from 0 to 9 are printed consecutively (clockwise) on the periphery of each wheel. The topmost digits of the wheels form a four-digit integer. For example, in the following figure the wheels form the integer 8056. Each wheel has two buttons associated with it. Pressing the button marked with a *left arrow* rotates the wheel one digit in the clockwise direction and pressing the one marked with the *right arrow* rotates it by one digit in the opposite direction.



The game starts with an initial configuration of the wheels. Say, in the initial configuration the topmost digits form the integer $S_1S_2S_3S_4$. You will be given some (say, n) forbidden configurations $F_{i_1}F_{i_2}F_{i_3}F_{i_4}$ $(1 \leq i \leq n)$ and a target configuration $T_1T_2T_3T_4$. Your job will be to write a program that can calculate the minimum number of button presses required to transform the initial configuration to the target configuration by never passing through a forbidden one.

How to Model With Graphs

- the task is to find the minimum number of button presses to get from one configuration to another
  - a strategy is to find the shortest such sequence of configurations and then count the number of steps
    - → a sequence of configurations sounds like a path
    - vertices represent allowed configurations
      - no vertices for forbidden configurations
    - edges represent button presses
      - undirected because a button press is reversible



Building a particular software package requires compiling a number of modules. In order to compile a module, any other modules that it depends on must be compiled first. Determine an order of compilation for the modules so that the whole package can be built successfully.

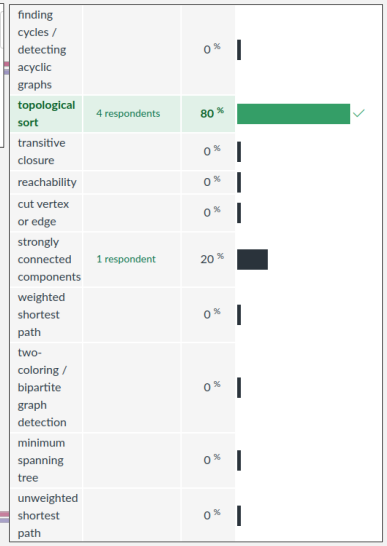
vertices?
→ modules

edges?
→ dependencies

directed or undirected?
→ directed – u → v means u depends on v (v must be compiled first)

edge weights?
→ no

what does a solution look like in the graph?
→ an ordering of vertices such that modules depended on are compiled before those that need them



Red light cameras have been placed at strategic intersections around a city. An intersection is "strategic" if the only route between two points passes through that intersection. For example, if there are 6 intersections (A, B, C, D, E, F) and 7 streets connecting intersections (B-C, A-B, C-A, D-C, D-E, E-F, F-C), then there is a camera at C because the only route from A to E goes through C. Given a set of intersections and routes, find the speed cameras.

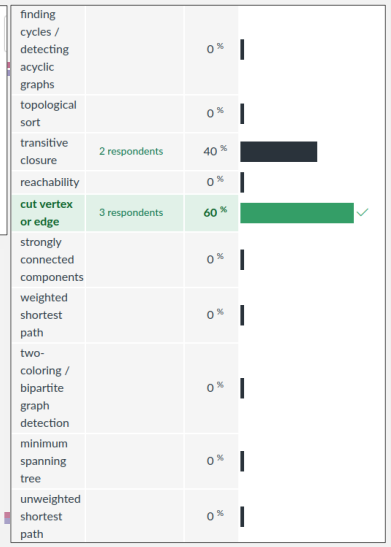
vertices?
→ intersections

edges?
→ roads between intersections

directed or undirected?
→ undirected

edge weights?
→ no

what does a solution look like in the graph?
→ a set of vertices such that the only route between some pair of vertices passes through that vertex



Electricity is finally coming to a (very) rural small town. Given the cost of stringing wire between every pair of houses in the town, determine how to connect the town with the smallest possible total cost.

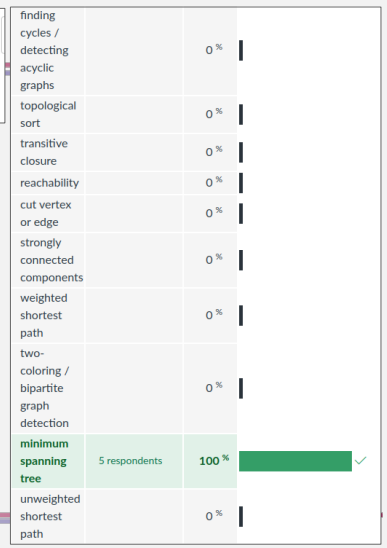
vertices?
→ houses

edges?
→ between every pair of vertices – possible connection for wiring

directed or undirected?
→ undirected

edge weights?
→ cost of stringing wire between that pair of houses

what does a solution look like in the graph?
→ a set of edges connecting every vertex with minimum cost



Within social networks, finding close-knit communities where everyone follows, directly or indirectly, everyone else in the group.

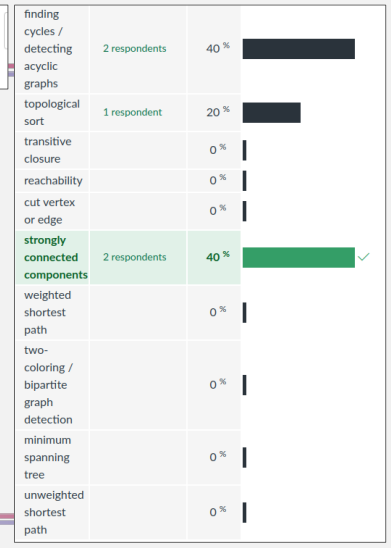
vertices?
→ members of the community

edges?
→ following relationship

directed or undirected?
→ directed – u → v means u follows v

edge weights?
→ no

what does a solution look like in the graph?
→ subsets of vertices where there is a path from every vertex to every other vertex within that subset



7-28. *[5]* Your job is to arrange n ill-behaved children in a straight line, facing front. You are given a list of m statements of the form "i hates j." If i hates j, then you do not want to put i somewhere behind j, because then i is capable of throwing something at j.

vertices?
→

edges?
→

directed or undirected?
→

edge weights?
→

what does a solution look like in the graph?
→

algorithm?
→

7-29. *[3]* A particular academic program has n required courses, certain pairs of which have prerequisite relations so that (x, y) means you must take course x before y. How would you analyze the prerequisite pairs to make sure it is possible for people to complete the program?

vertices?
→

edges?
→

directed or undirected?
→

edge weights?
→

what does a solution look like in the graph?
→

algorithm?
→

7-22. *[5]* Consider a set of movies $M_1, M_2, \ldots, M_k$. There is a set of customers, each one of which indicates the two movies they would like to see this weekend. Movies are shown on Saturday evening and Sunday evening. Multiple movies may be screened at the same time.

You must decide which movies should be televised on Saturday and which on Sunday, so that every customer gets to see the two movies they desire. Is there a schedule where each movie is shown at most once? Design an efficient algorithm to find such a schedule if one exists.

vertices?
→

edges?
→

directed or undirected?
→

edge weights?
→

what does a solution look like in the graph?
→

algorithm?
→

Adapting Algorithms

If you don't have an exact algorithm for the problem you need to solve, there are two options –

Option #1 – adapt an existing algorithm to work with your situation

Option #2 – build an instance of a known problem so that the solution produced by a standard algorithm solves your problem

Adapting Algorithms

A concrete example –

Find the shortest path from s to every other vertex in a graph with vertex weights.

- cost of path = sum of weights of vertices on the path

Option #1 – adapt Dijkstra's algorithm to work with vertex weights instead of edge weights.

- requires proving that the new algorithm is correct

Option #2 – transform the vertex-weighted graph into an edge-weighted graph so that the shortest paths in the edge-weighted graph have the same cost as the shortest paths in the vertex weighted graph.

- make the graph directed
  - every undirected edge (u,v) in the original graph will turn into two directed edges (u,v) and (v,u)
- add new start vertex s' with edge s' → s
- give edge i → j the weight of vertex j
- run Dijkstra's algorithm starting from s'

Adapting Algorithms

Advantages of option #2 –

- can be easier to argue why a solution in the new graph is equivalent to a solution in the original graph than to argue correctness of a new algorithm
- can use a library implementation of the algorithm instead of having to code your own variation

7-14. *[3]* Plum blossom poles are a Kung Fu training technique, consisting of n large posts partially sunk into the ground, with each pole p_i at position (x_i, y_i). Students practice martial arts techniques by stepping from the top of one pole to the top of another pole. In order to keep balance, each step must be more than d meters but less than $2d$ meters. Give an efficient algorithm to find a safe path from pole p_s to p_t if it exists.

vertices?
→ poles

edges?
→ a step is possible between the two poles (between d and 2d meters apart)

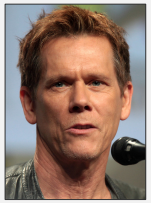
directed or undirected?
→ undirected

edge weights?
→ no

what does a solution look like in the graph?
→ a path from pole p_s to pole p_t

algorithm?
→ reachability or unweighted shortest path

Six Degrees of Kevin Bacon and Bacon Numbers



Six degrees of Kevin Bacon: find a way to link a particular actor to Kevin Bacon through movies they have in common.

The **Bacon number** of an actor is the number of degrees of separation he or she has from Bacon, as defined by the game. This is an application of the Erdős number concept to the Hollywood movie industry. The higher the Bacon number, the greater the separation from Kevin Bacon the actor is.

- Kevin Bacon himself has a Bacon number of 0.
- Those actors who have worked directly with Kevin Bacon have a Bacon number of 1.
- If the lowest Bacon number of any actor with whom X has appeared in any movie is N, X's Bacon number is N+1.

Examples [edit]

Elvis Presley:

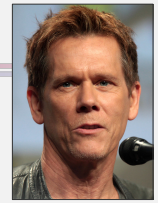
- Elvis Presley was in *Change of Habit* (1969) with Edward Asner
- Edward Asner was in *JFK* (1991) with Kevin Bacon

Therefore, Asner has a Bacon number of 1, and Presley (who never appeared in a film with Bacon) has a Bacon number of 2.

Ian McKellen:

- Ian McKellen was in *X-Men: Days of Future Past* (2014) with Michael Fassbender and James McAvoy
- McAvoy and Fassbender were in *X-Men: First Class* (2011) with Kevin Bacon

Therefore, McAvoy and Fassbender have Bacon numbers of 1, and McKellen has a Bacon number of 2.



Compute the Bacon number for an actor / determine if their Bacon number ≤ 6.

vertices?
→

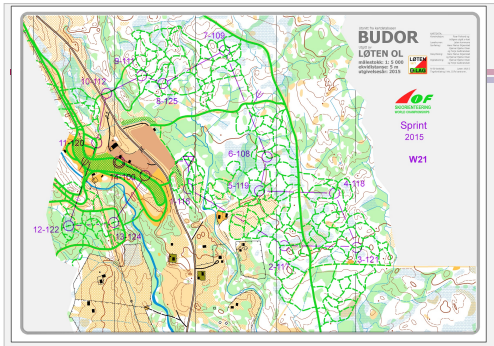
edges?
→

directed or undirected?
→

edge weights?
→

what does a solution look like in the graph?
→

algorithm?
→



vertices?

edges?

directed or undirected?

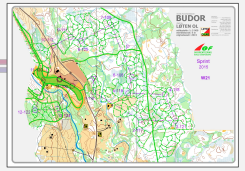
edge weights?

what does a solution look like in the graph?

algorithm?

- ski orienteering – find the fastest route from the start (triangle) to the finish (double circle), visiting each of the control locations (purple circles) in the order indicated by the first number in the label (e.g. 1-116, 2-117, 3-121, etc)
  - it is OK to visit controls more than once / out of order as long as there is a sequence in the correct order
- the start and finish are at track junctions, but all controls are along tracks (assume in the middle of a track segment)
- all tracks are two-way but you may only change directions at a junction or a control (not the case in actual competition)
- you must stay on a track and not cut between tracks (not the case in actual competition)
- the speed of travel on a track may be different in different directions

Ski orienteering.



vertices?
→

edges?
→

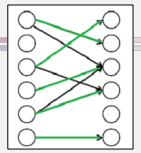
directed or undirected?
→

edge weights?
→

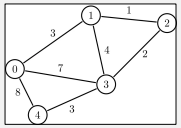
what does a solution look like in the graph?
→

algorithm?
→

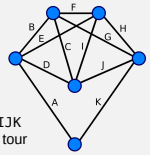
More Graph Algorithms



bipartite matching – subset of edges such that no two edges share a vertex



min cut – cheapest set of edges whose removal partitions the graph into two disjoint subsets



postman tour – min-length cycle traversing every edge at least once

euler cycle (tour) – cycle traversing every edge exactly once

ABCDEFGHIJK
is an euler tour

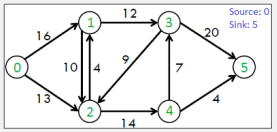
https://www.geeksforgeeks.org/maximum-bipartite-matching/
https://hyperskill.org/learn/step/5645
https://en.wikipedia.org/wiki/Eulerian_path

More Graph Algorithms

maximum spanning tree – spanning tree that maximizes the sum of the edge weights

minimum bottleneck spanning tree – spanning tree with minimum max edge weight (of all the spanning trees, this one has the smallest max edge weight)

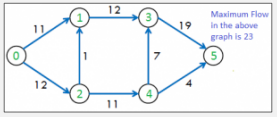
minimum product spanning tree – spanning tree that minimizes the product of the edge weights



flow network – edge weights denote max capacity of the edge

max flow – max flow that can be sent from *s* to *t* while respecting max capacities

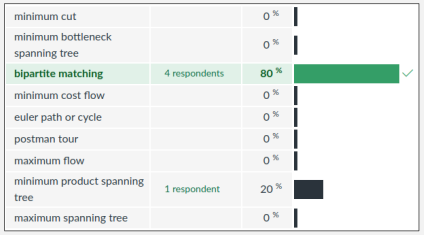
min cost flow – cheapest way to send a given flow volume from *s* to *t*



https://www.geeksforgeeks.org/max-flow-problem-introduction/

Assign projects to teams with the skill set to complete the assigned project, such that each team is assigned at most one project, each project is assigned to at most one team, and the maximum number of projects are completed.

spanning trees connect all vertices with no restrictions on the degree of the vertex in the tree



vertices?
→ projects, teams

edges?
→ connect projects to teams with the necessary skill set

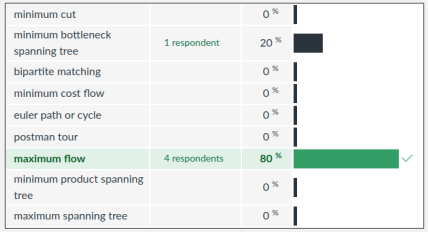
directed or undirected?
→ undirected

edge weights?
→ no

what does a solution look like in the graph?
→ a matching – a maximum-size collection of edges such that each vertex (project and team) is included at most once

Each connection in a computer network has a maximum bandwidth. Find the maximum bandwidth that can be achieved between computer A to computer B.

minimum bottleneck spanning tree is the spanning tree whose max edge weight is lowest over all spanning trees but max bandwidth along a path is determined by the lowest max bandwidth edge on that path



vertices?
→ computers

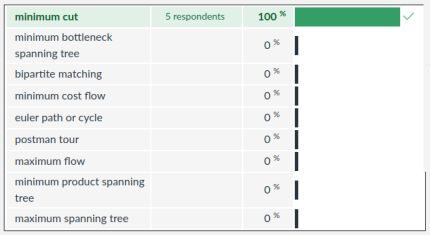
edges?
→ between computers with a direct connection

directed or undirected?
→ undirected

edge weights?
→ yes (max bandwidth)

what does a solution look like in the graph?
→ how much data can be sent from A to B subject to max bandwidth constraints

How vulnerable is an electrical grid? Determine how many transmission lines would need to be interrupted in order to cut off at least one community from the power plant.



vertices?
→ communities, the power plant

edges?
→ transmission lines between communities

directed or undirected?
→ undirected

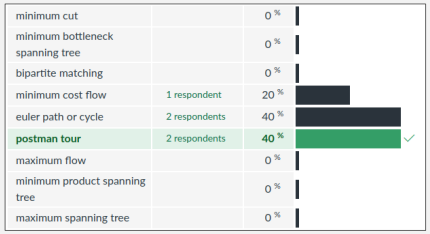
edge weights?
→ no

what does a solution look like in the graph?
→ the smallest cut that separates at least one community from the power plant

Find the optimal route for plowing all of the roads in a city.

flow is about getting things from one place to another

euler cycle is the right idea, but not all graphs have an euler cycle – it may be necessary to repeat some edges to visit all of them



vertices?
→ intersections

edges?
→ roads

directed or undirected?
→ undirected

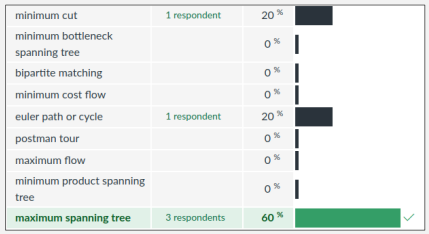
edge weights?
→ no

what does a solution look like in the graph?
→ a route that covers all of the roads at least once – ideally no more than once, but that is not always possible

Finding the core backbone of key influencers within a social network.

cuts are about disconnecting a graph

euler path/cycle is about a route covering every edge



vertices?
→ members of the social network

edges?
→ who follows who

directed or undirected?
→ directed

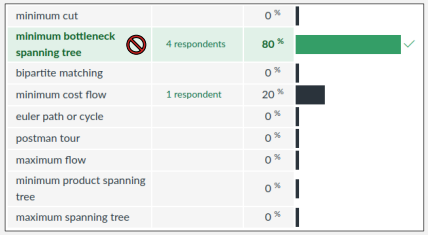
edge weights?
→ yes (by number of followers)

what does a solution look like in the graph?
→ key influencers = most followers
→ "core backbone" suggests ensuring that everyone is connected

Determine if there is sufficient bandwidth between every pair of computers within a network.

minimum bottleneck spanning tree is the spanning tree whose max edge weight is lowest over all spanning trees

max flow could be used to determine the bandwidth between a given pair of computers – about how much can be moved, not how cheaply (not min cost flow)



vertices?
→ computers

edges?
→ direct connections between computers

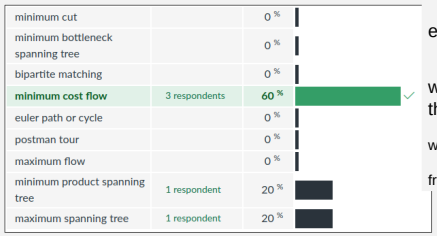
directed or undirected?
→ undirected

edge weights?
→ yes (available bandwidth for the connection)

what does a solution look like in the graph?
→ the max bandwidth of a path is the min weight edge along that path
→ for each pair of computers, want the path with the highest max bandwidth

Determine the most cost-effective way to ship products from a collection of warehouses to a collection of retailers while satisfying the retailers' demand for product.

flow involves getting things from one place to another; spanning trees are about connectivity



vertices?
→ warehouses, retailers

edges?
→ between warehouses and retailers where there is a direct route (likely all pairs)

directed or undirected?
→ undirected

edge weights?
→ yes (cost of shipping)

what does a solution look like in the graph?
→ also vertex weights (supply at a warehouse, demand at a retailer)
→ flow network – getting something from one place to another

Mr. G. works as a tourist guide. His current assignment is to take some tourists from one city to another. Some two-way roads connect the cities. For each pair of neighboring cities there is a bus service that runs only between those two cities and uses the road that directly connects them. Each bus service has a limit on the maximum number of passengers it can carry. Mr. G. has a map showing the cities and the roads connecting them. He also has the information regarding each bus service. He understands that it may not always be possible for him to take all the tourists to the destination city in a single trip.

vertices?
→ cities

edges?
→ existence of a bus service between cities

directed or undirected?
→ undirected (two-way roads)

edge weights?
→ yes (bus capacity)

what does a solution look like in the graph?
→ routes from one city to another which together get all of the passengers to their destination without exceeding any bus capacities

algorithm?
→ flow network / max flow – add a source vertex *s* connected to the start city with edge capacity equal to the number of tourists, *t* is the destination city

An *edit step* is a transformation from one word x to another word y such that x and y are words in the dictionary, and x can be transformed to y by adding, deleting, or changing one letter. So the transformation from *dig* to *dog* or from *dog* to *do* are both edit steps. An *edit step ladder* is a lexicographically ordered sequence of words $w_1, w_2, \ldots, w_n$ such that the transformation from w_i to w_{i+1} is an edit step for all i from 1 to $n - 1$.

For a given dictionary, you are to compute the length of the longest edit step ladder.

vertices?
→

edges?
→

directed or undirected?
→

edge weights?
→

what does a solution look like in the graph?
→

algorithm?
→

In this problem you are given N colorful cubes each having a distinct weight. Each face of a cube is colored with one color. Your job is to build a tower using the cubes you have subject to the following restrictions:

- Never put a heavier cube on a lighter one.
- The bottom face of every cube (except the bottom cube, which is lying on the floor) must have the same color as the top face of the cube below it.
- Construct the tallest tower possible.

vertices?
→

edges?
→

directed or undirected?
→

edge weights?
→

what does a solution look like in the graph?
→

algorithm?
→