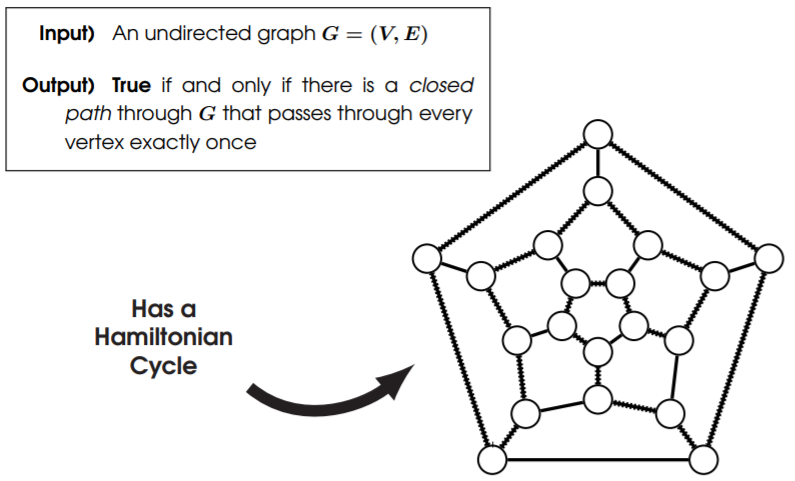
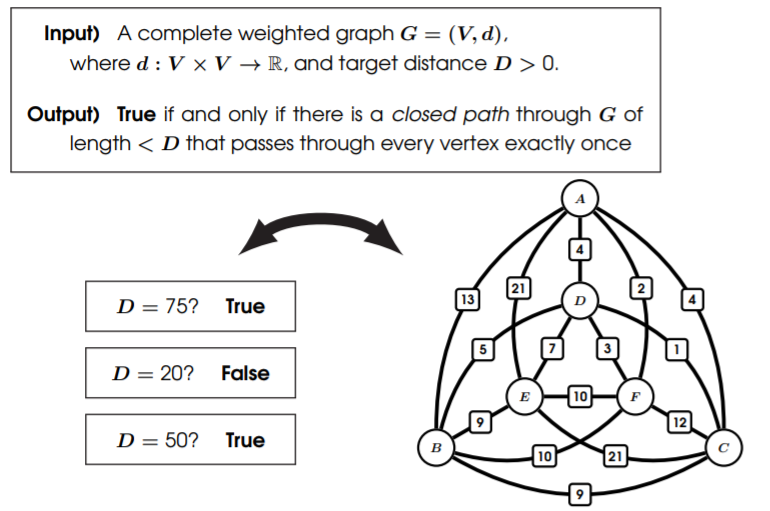
The Hamiltonian Cycle Problem

* That is there is a path that starts end ends at the same nodes, and it only visit one node once.
* Naively: Try every possibility, brute-force it. 20! Possible combination
* Non-deterministic Turing Machine can solve this quickly because each cycle can be split between the processors, so all 20! can run in parallel, which will be in polynomial time to check weather is a Hamiltonian cycle. Therefore, this is a problem that belongs to the class NP.

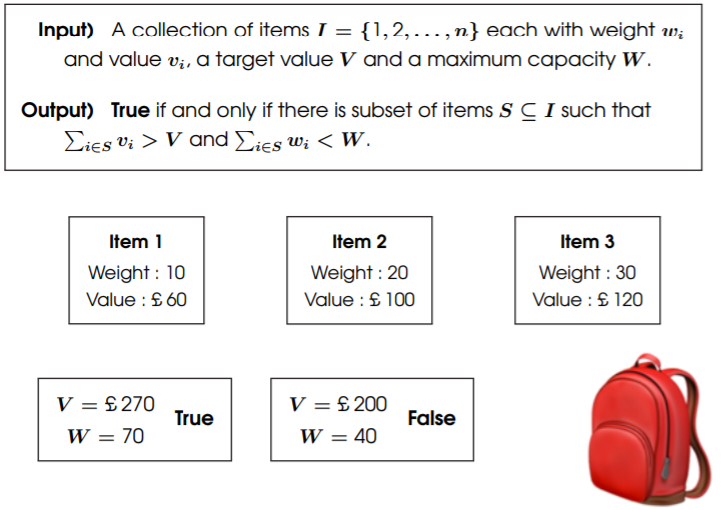
The Travelling Salesman Problem

* Given value **D** we must evaluate if it is possible to make on the graph.
* Non-deterministic Turing Machine can solve this problem quickly because it can split checking the paths in parallel on a different processor. Therefore, that’s why it is NP.

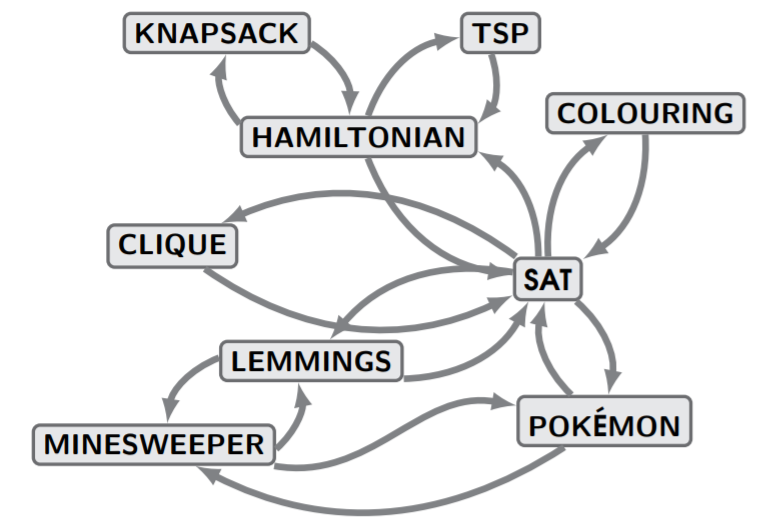


The Knapsack Problem

* So, suppose you are robbing into apple store, with limited capacity in the bag, this problem can evaluate to true, if there is a subset of items that are worth highest price and within the possible capacity.

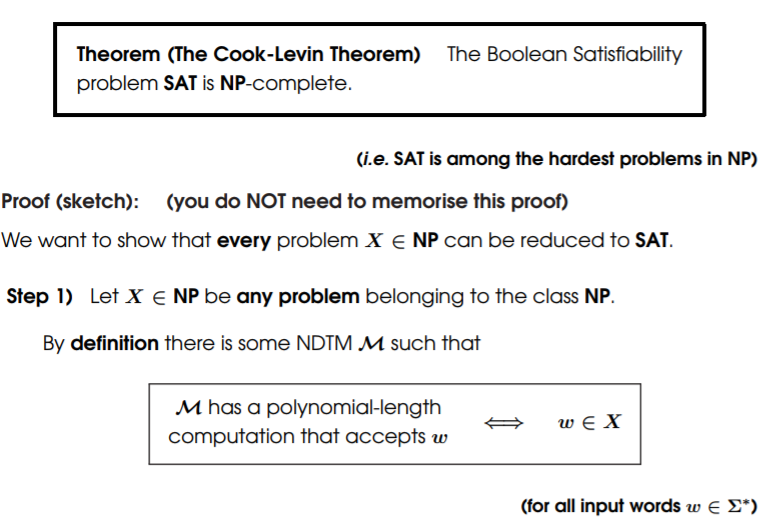


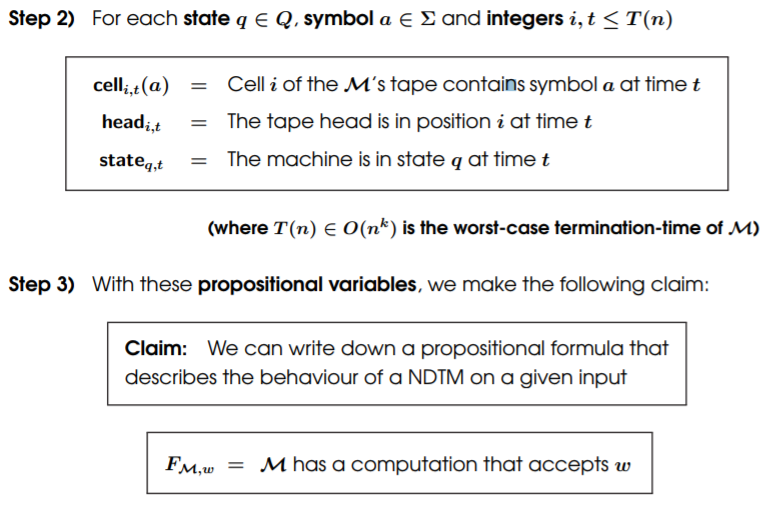
Relationships between NP-complete Problems

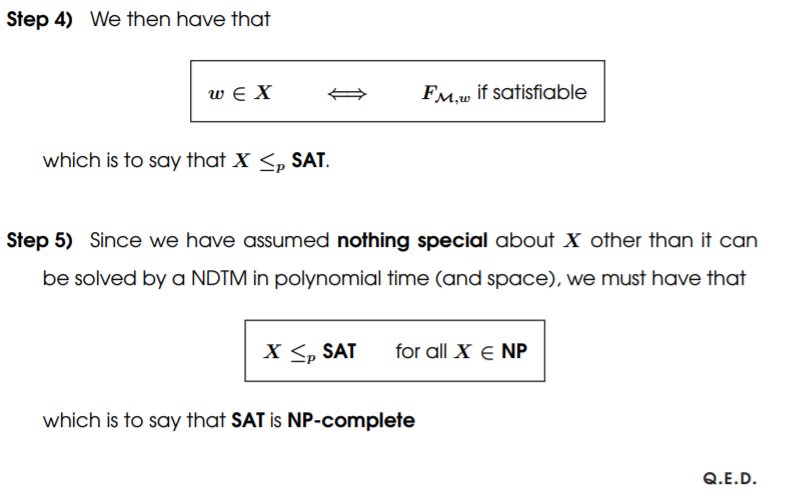


The Cook-Levin Theorem

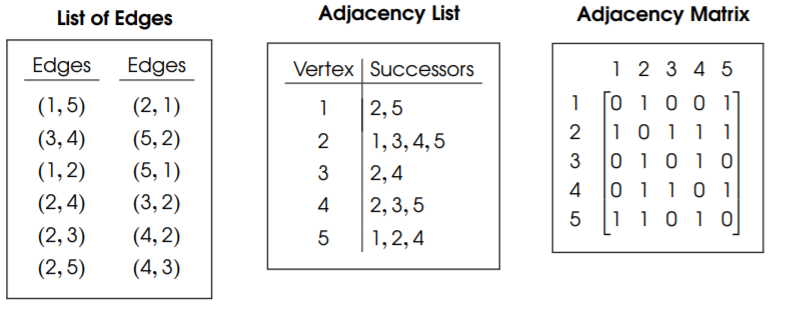
*(The Existence of NP-Complete Problems)*

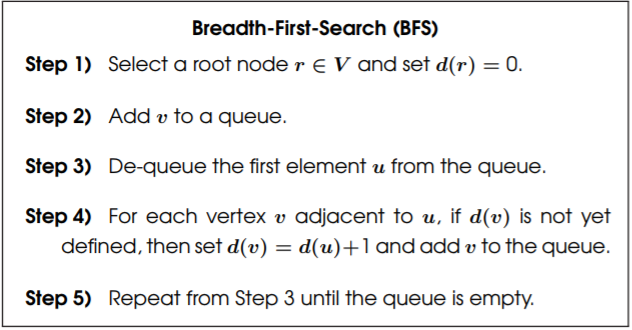






Graph Algorithms

* Data Structures for Graphs
  + The type of **data structure** used to store a graph can affect the efficiency of your algorithms, and memory requirements.

Breadth-First-Search

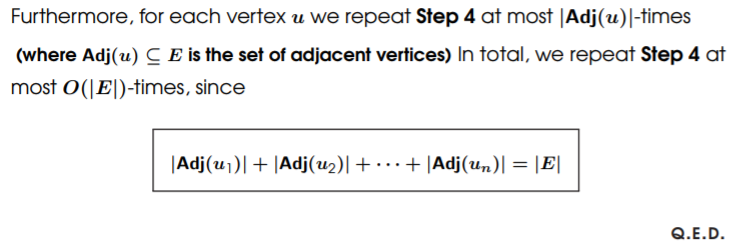
* Step 1) Pick a root note when search starts from
* Step 2) Add all vertices that can be accessed from to front node to the queue
* Step 3) remove element that is in front of queue
* Step 4) add all nodes to queue that was accessible from the removed node, no repetition
* Step 5) Repeat from Step 3 until the queue is empty.

### *The point is to measure the distance of all nodes from the root in some order.*

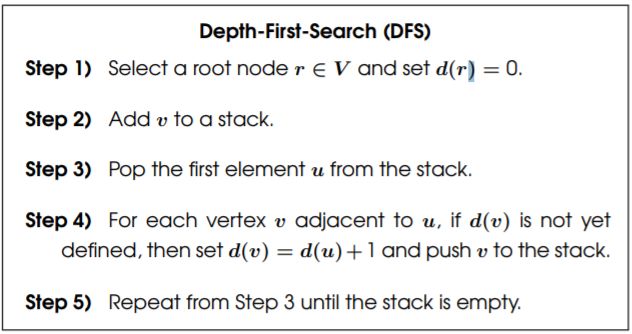
*If we don’t check If the node is already in the queue, we could end up in an infinite loop.*

***Theorem****: The worst-case termination time for Breadth-First-Search is* ***O(|v| + |e|)*** *which is Big oh of cardinality of vertices plus cardinality of the edges. Takes polynomial time.*

***Proof****:*

* Each vertex is enqueued at most once and so the number of times that **Step 3** and **Step 5** are repeated is at most **O(|v|)** times.

Depth-First-Search



***Theorem:*** The worst-case termination time for Depth-First-Search is **O(|V|+|E|)**

***Proof:*** The poof is almost identical to that of the **Breadth-First-Search,** replacing the **queue** for a **stack.**