GREEDY ALGORITHMS

MATH CIRCLE 4/27/2018

Introduction

(1) Normal coins in the U.S. come in denominations/values of $1, 5, 10, 25$ cents. What happens if we change these denominations?

Group	Denominations
A	$1, 5, 10, 25$
B	$4, 10, 25$
C	$2, 16, 20$
D	$3, 5, 11, 40$

a) For each group of denominations (A, B, C, and D) decide if it is possible to make change, and if so, the fewest number of coins required, for the following amounts of money:

i) 14

ii) 50

iii) 41

iv) 76

b) For each group of denominations, determine all amounts for which change can be made. For example, with U.S. denominations, **all** amounts can be made.

c) Try to come up with a method/algorithm for making change with group A. Bonus: Try to do so "locally", i.e. without any planning ahead.

2) Recall that a graph $G = (V, E)$ is a set of vertices (denoted by V) as well as a set of edges (denoted E) between the vertices.

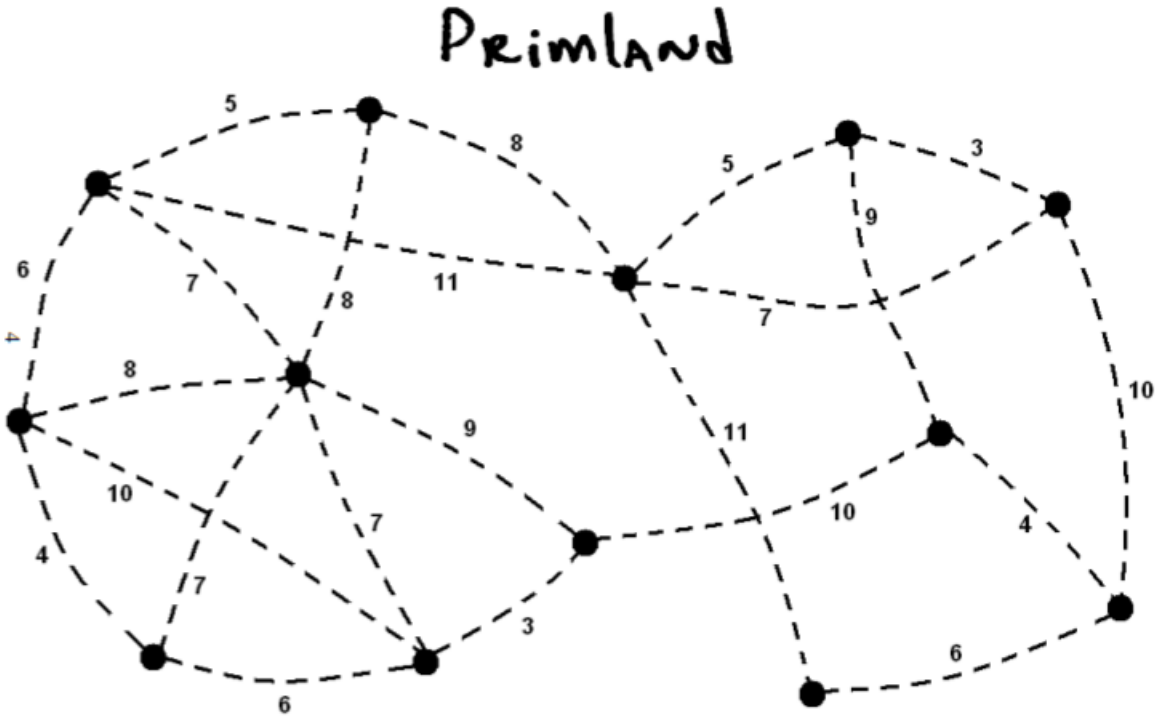
A graph is *connected* if it is possible to travel (along edges) between any two vertices.

A *subgraph* of G is another graph $H = (V', E')$ with $V' = V$ and $E' \subseteq E$; that is, H has the same vertices as G, every edge in H is an edge in G, but not every edge from G is in H. Even more simply, we get subgraphs of G by removing edges from G.

A *minimal spanning tree* (MST) of a connected graph G is a connected subgraph H of G with the smallest possible number of edges.

a) Find a MST for the graph on the next page.

b) Try to find a method/algorithm for coming up with MST for arbitrary connected graphs. Hint: Start at an arbitrary vertex, and build up a minimal spanning tree one edge at a time.


Primland
5
8
5
3
9
6
7
11
8
7
8
10
9
11
10
10
4
7
4
7
3
6
6

Definition: A greedy algorithm is an algorithm that makes "locally" optimal choices at each stage, in the hope of ending up with a globally optimal solution.

Greedy Algorithm For Making Change

Making Change Greedy Algorithm: Continually pick the largest denomination possible until you either succeed in making change or you cannot pick any more coins.

3) For each group of denominations from Problem 1 (A, B, C, D), decide which statement below is correct (and give a reason for your answer):

i) If it is possible to make change, the algorithm always gives a way to make change optimally (with the fewest coins).

ii) If it is possible to make change, the algorithm always gives a way to make change, but not always optimally.

iii) If it is possible to make change, the algorithm can fail to make change.

4) Suppose we want to add a fifth coin (worth more than 25 cents) to group A. Find a value for this coin so that statement i) is still true. Bonus: Find the smallest value possible.

5) Challenge: a) Suppose that $\{a_1, \ldots, a_n\}$ is a set of denominations so that statement i) is true. Show that statement i) is still true for $\{a_1, \ldots, a_n, 2a_n\}$.

Solution: For values below $2a_n$ the algorithm is the same for both coin sets, so suppose $X \geq 2a_n$. Thus step one of the algorithm will be to choose the largest possible coin, $2a_n$. If this value is still at least $2a_n$, this process will repeat until we get a value $X' < 2a_n$. Changing X' with the new coin set is the same as changing it with the old coin set since it is below $2a_n$, so we know it can be done optimally. This uses the least possible number of coins since the changing for X' is minimal and we use the largest possible coin to get from X' to X.

b) Use a) to build a set of denominations so that you can give change for any value less than a dollar using ≤ 7 coins.

Solution: Answers vary, but an example is {1,2,3,4,6}.

Greedy Algorithm For Finding Minimal Weight Spanning Trees

A *weighted* graph is a graph such that each edge has a weight (i.e. positive number) associated with it. The weight of a (weighted) graph is the sum of all its edge's weights.

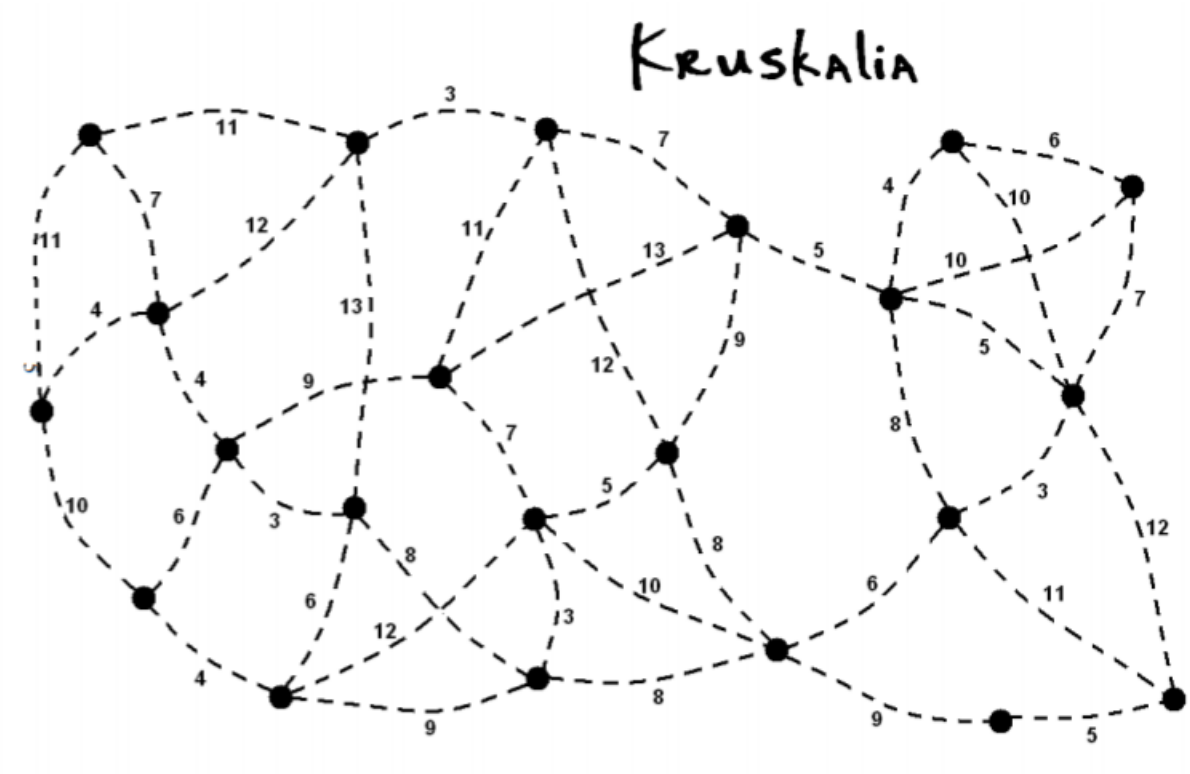
A *minimal weight spanning tree* (MWST) of a connected graph G is a connected subgraph H of G with the smallest possible weight.

Example: If every edge has weight 1, then MWST are identical to MST.

6) For the graph on the next page, find a MWST.

7) a) Modify the algorithm from Problem 2b) to get a greedy algorithm for trying to generate MWST. This is called Prim's algorithm.

b) Use your algorithm on the graph on the next page. Do you think that the algorithm always gives us MWST?


Kruskalia

8) Come up with another greedy algorithm for generating MWST. Rough idea: At each stage, add an edge (from anywhere in the graph) with minimal weight. This is called Kruskal's algorithm. Test your algorithm using the graph on the next page.

9) Use both Prim's algorithm and Kruskal's algorithm on Primland and Kruskalia. Show that they may generate the same MWST, but don't always need to. If they don't generate the same MWST, do you notice any similarities between the MWST?

10) Challenge: Try to prove Kruskal's algorithm always generates MWST. Hint: Prove by induction that at every stage, the edges chosen are part of *some* MWST.

Solution: First we easily show that the output is a spanning tree. First, because the output has no cycles (by design, we never add an edge which creates a cycle). Second, the output is spanning because if there were a vertex not included, we would be able to include its minimum weight incident edge without creating any cycles. Third, the output is connected. If not, since G is connected, there must be a potential bridge between connected components that can be added. The minimum weight edge among these would have been added as it would not have created a cycle.

Now we prove that the output is minimum weight. We prove by induction that each chosen edge is a part of some MWST. Let $T*$ be a minimum weight spanning tree. The first edge $e = (u, v)$, the overall minimum weight edge of G, certainly has to be in $T*$. Suppose not. Then there is some other path from u to v in the MWST. Any of the edges along this path can be exchanged with our edge e without increasing the weight of the tree. Thus, e is in some MWST. Now, suppose we have added k edges and we have a tree T, not spanning yet, and suppose T is minimum weight. Let $f = (s, t)$ be the minimum weight edge that has not been included which would not produce a cycle. Again, suppose f is not in $T*$. Then there is some other path from s to t in $T*$. There must be some edge along this path that is **not** included in $T*$, because if

not, the entire path plus the edge f would form a cycle. This non-included edge can be exchanged with f without increasing the weight and without creating any cycles, so f is a part of some MWST. Thus, by induction, every edge chosen by Kruskal's algorithm is part of some MWST, so the output will always be an MWST.