Algorithms

DATE: 05/04/2022

Minimum spanning tree problem

NETWORK DESIGN PROBLEM

How can we solve the problem if we only have 1 digit? $\boxed{1}$

- $n_1, n_2, \dots, n_k, \dots$

With two digits we can hit the 1 as many time as needed to represent the value of a digit, then we press 0 before starting to encode the next digit.

$\boxed{0}$ $\boxed{1}$

$\underbrace{\boxed{1} \dots \boxed{1}}_{n_1}$ $\boxed{0}$ $\underbrace{\boxed{1} \dots \boxed{1}}_{n_2}$ $\boxed{0}$

THM: IF $n \geq 2$ is an integer, then there exists a unique sequence of prime numbers $2 \leq p_1 < p_2 < p_3 < \dots < p_k$ and a unique sequence of natural numbers $n_1 \geq 1,\ n_2 \geq 1, \dots, n_k \geq 1$ S.T.

$$n = \prod_{i=1}^{k} p_i^{n_i}$$

Fundamental theorem of algebra.

- $15 = 3 \cdot 5 = 3^1 \cdot 5^1$

  $p_1 = 3 \quad p_2 = 5$
  $n_1 = 1 \quad n_2 = 1$

- $20 = 2^2 \cdot 5^1$

  $p_1 = 2 \quad p_2 = 5$
  $n_1 = 2 \quad n_2 = 1$

Let's say that P1 is the 1st prime number.

There are infinitely many prime numbers.

$2 = P_1, \quad 3 = P_2, \quad 5 = P_3 \dots$

P_i is the iTH prime number

$n_1, n_2, n_3, \dots, n_k \Rightarrow \boxed{1} \dots \boxed{1}$ $\quad P_1^{n_1} P_2^{n_2} \dots P_k^{n_k} = N_k$

ESEMPIO: ADDS $n_{k+1} \Rightarrow P_1^{n_1} P_2^{n_2} \dots P_k^{n_k} P_{k+1}^{n_{k+1}} = N_{k+1}$

$2, 3 = 4 \cdot 27 = 108$

108	2
54	2
27	3
9	3
3	3
1	

$108 = 2^2 \cdot 3^3$

$N_{k+1} \geq N_k$

HIT $\boxed{1}$ $N_{k+1} - N_k$ TIMES

If we want to codify these 2 numbers, we have to press 1 108 times. (inefficient)

$\boxed{0}$ $\boxed{1}$

$\underbrace{\boxed{1} \dots \boxed{1}}_{n_1}$ $\boxed{0}$ $\underbrace{\boxed{1} \dots \boxed{1}}_{n_2}$ $\boxed{0}$

If we'd like to add another integer to a sequence, then we'd have to find a number that added to a previous number, after the decomposition it returns back prime factors whose exponents represent the initial sequence of integers.

How many digits to represent $n_1, n_2, \dots, n_k$? in binary

$$\sum_{i=1}^{k} (n_i + 1) = \sum_{i=1}^{k} n_k + k$$

"$n_i + 1$ bits to represent n_i"

"in binary we use $O(\log n_i)$ bits to represent n_i"

$n_i \xrightarrow{\text{base } 2} b_1, b_2, \dots, b_k$

$0b_1\ 0b_2\ 0b_3\ 0b_k\ \boxed{1}\ 1$

$2\log_2 n_i = 2k$
$3\log_2 n_i = 3k$

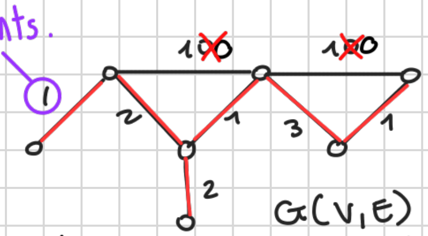
$2\log_2 n_i$
$\log_2 n_i + O(\log \log n_i)$

$11111 = 5$

$\boxed{11}\boxed{10} = 5$

NETWORK DESIGN PROBLEM

#Weights.



With weights
$c : E \to \mathbb{R}_{\geq 0}$

$G(V,E)$ is a weighted, connected, graph.

#Here the goal is to take/consider all the edges of minimum length in such a way to still have all the nodes of the given graph connected, but paying attention that these nodes doesn't form cycles between them.

#A property of this graph is that at least 1 of the edges will have length 1.

We have $V = \{v_1, v_2, \dots, v_n\}$ locations.

Some pairs of locations, those in E, can be directly linked. Directly linking $\{v_i, v_j\} \in E$ costs $c(v_i, v_j) \geq 0$.

Assuming G is connected, what is the minimum price for indirectly connecting each pair of nodes of V?

We aim to find subset $T \subseteq E$ of the edges so that:

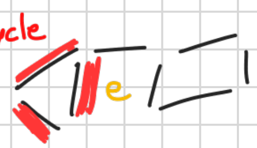
- $G(V,T)$ is connected, and
- The cost of T, $cost(T) = \sum_{e \in T} c(e)$, is minimum.

#The set of minimum length edges that we select.

L: Let T be an optimal solution to the network design problem. Then, $G(V,T)$ is a tree.

P: By definition, $G(V,T)$ has to be connected. We will show that $G(V,T)$ cannot contain cycles - thus, it has to be a tree (a connected graph with no cycles is a tree).

By contradiction, suppose that $G(V,T)$ contains a cycle C. Let "e" be any edge of the cycle.



$G(V, T-\{e\})$ is also connected,

since any path that went through the edge "e" can be rerouted through $C - \{e\}$.
Thus, $T-\{e\}$ is a valid (FEASIBLE) solution to the network design problem. (you can go from any node to any other node — that is, $G(V, T-\{e\})$ is connected).

$\boxed{\text{The cost of this new solution is:}}$

$$COST(T-\{e\}) = \sum_{e' \in T-\{e\}} c(e') = \left(\sum_{e' \in T} c(e')\right) - c(e) = COST(T) - c(e)$$

Recall that $c(e) > 0$, thus the $cost(T-\{e\}) = COST(T) - c(e) < COST(T)$. Thus, T is not an opt. solution.
CONTRADICTION ■

NOTE: This is also known as minimum spanning tree algo.

Thus, the network design problem is actually asking to find a subtree of $G(V,E)$ of minimum cost, and that connects each pair of vertices.

The latter problem is known as MINIMUM SPANNING TREE (or MST).

GREEDY APPROACHES?

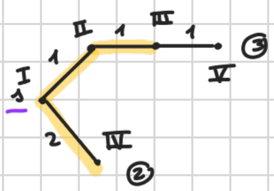
① PRIM'S ALGORITHM.

#The set of visited nodes

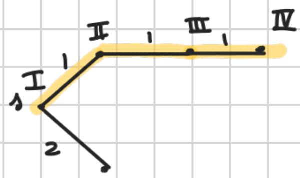
You start from an arbitrary node s. Let $S \leftarrow \{s\}$.

To select a new edge, we pick one having smallest cost, among those that take us from some node in S to some node in $V-S$.

s: source



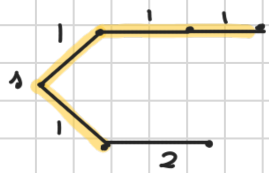
DIJKSTRA



PRIM

chooses edges that don't create cycles.

#In dijkstra, to select a new node, to visit, we picked the closest one to the source node.



#In prim's algorithm we pick all the edges having smallest (dist.) from the actual node and that doesn't create a cycle.

② KRUSKAL'S ALGORITHM

Sort the edges increasingly by cost. Let $T \leftarrow \emptyset$.
Scan the list of edges e:

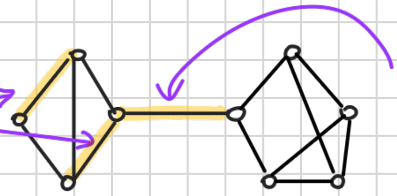
If "e" can be added to T w/o (without) creating cycles, add "e" to T.

3 REVERSE - KRUSKAL

Sort the edges decreasingly by cost. Scan the list of edges "e": IF "e" is part of a cycle (in the current graph), throw "e" away, o/w (otherwise) ADD "e" to T.

When is it "safe" to add an edge to a spanning tree T?

But what about these ones?



Even If this edge costs 1M €, It must be part of the MST.

Starting from an assumption w.l.o.G. that all the edges costs are different.

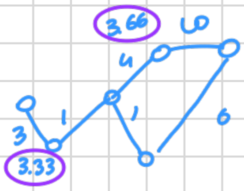
L: Assume that edge cost are pairwise distinct.

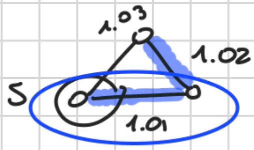
Otherwise this property might not hold.

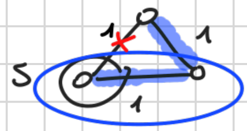
Let $\emptyset \subset S \subset V$ be a set of Nodes of $G(V, E)$.

Let $e \in E$ be an edge having smallest cost among the edges having one endpoint in S, and one in $V - S$.

Then, EACH MST of $G(V, E)$ contains "e".







Noise

δ is the minimum difference between distinct edges costs in $G(V,E)$

With the assumption there exists a single minimum from S to V-S.

Without the assumption not all the edges are part of a MST...

$-\frac{\delta}{3} \leq \text{noise} \leq +\frac{\delta}{3}$. If the noise that we add is small enough, then, It doesn't change the cost of any MST.

In other words, It's always safe to add e, if e has this property.

This is a pretty strong property, because, no matter which set of nodes we've reached, if we pick the one edge having smallest cost, from S to the rest of the graph, then, we're sure that we did the right choice, because each MST has to contain that edge, and therefore, if we take It, we make no mistake (It will always be a solution compatible to the edges we took so far, so the greedy property would be sastisfied).