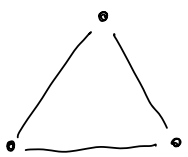
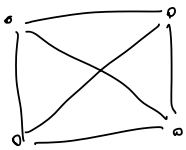
Traveling Salesman Problem TSP

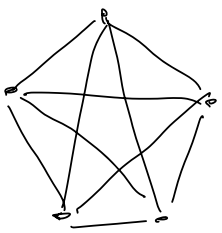
TSP is about Ham. circuits in <u>complete graphs</u>

A <u>complete</u> graph is one where all pairs of verts have an edge connecting them.

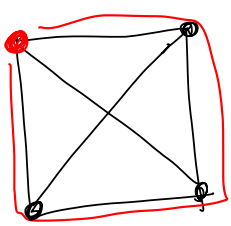


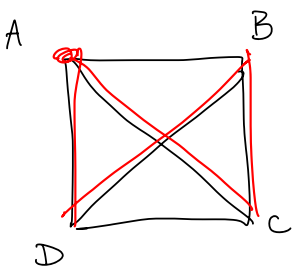
↑
The complete graph on 3 vertices.





A Ham circ. must visit every vertex with no repeats.



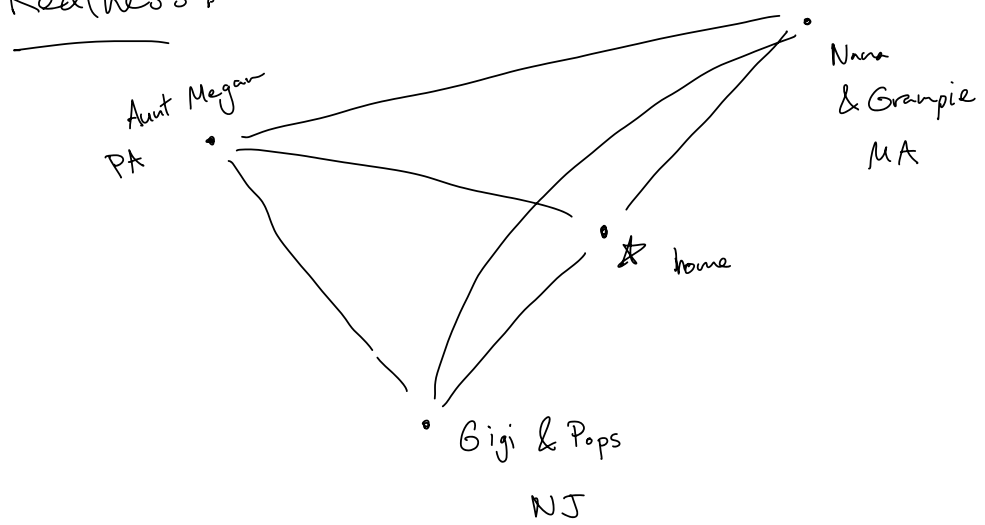


ACBDA
could do
ABCDA
or ACDBA

I can start & end at A, but ANY ordering of the letters will make a Ham. circ:

ABCDA
ABDCA
ACBDA
ACDBA
ADBCA
ADCBA

Realness:



What's the best route to visit them all, start & end at ☆.

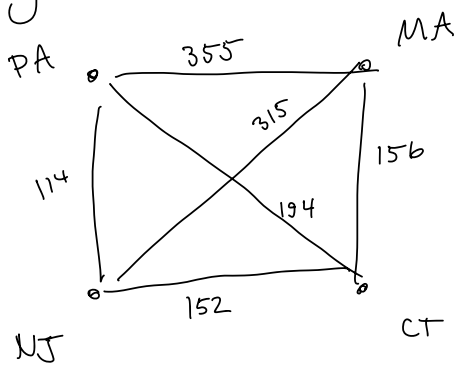
This will make a Hamiltonian circuit,
I want to minimize my total travel distance.

Similar to: A delivery truck with 100 packages to deliver to 100 addresses.

In what order should we deliver them?

(The Traveling Salesman Problem)

Driving:

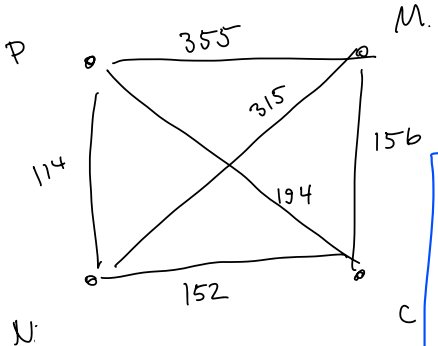


← this is a weighted graph

We want the Ham circ with minimum total weight.

Simplest idea: write down all Ham circs, find which one has smallest weight.

The Brute Force Method



Write down all Hams: find total weight

C M N P C → 156 + 315 + 114 + 194 = 779

C M P N C → 156 + 355 + 114 + 152 = 777

C N M P C → 152 + 315 + 355 + 194 = 1016

C N P M C ⟶ 777

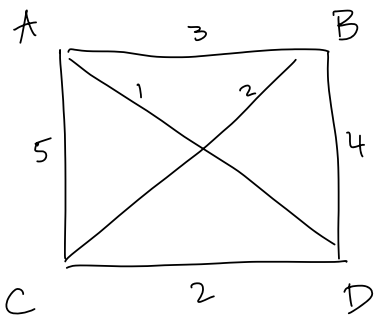
C P M N C ⟶ 1016

C P N M C ⟶ 779

Some are reversals, so we don't need to add it up twice.

Our best route is CT → NJ → PA → MA → CT

(or the reverse) total dist = 777.



Solve the TSP here by brute force starting at A.

A B C D A → 3+2+2+1 = 8

A B D C A → 3+4+2+5 = 14

A C B D A → 5+2+4+1 = 12

A C D B A → 14

A D B C A → 12

A D C B A → 8

Best route is ABDA or ADCBA for 8.

If we have many verts, there are MANY MANY routes to check.

even checking w/10 verts is not practical.

The Package guy needs to solve this problem every day with ~ 100 locations.

No known algorithm can find the best route in a reasonable amount of time.

So, brute-force works, but at large scales it's not practical to use brute-force.

Instead, we use tricks to find pretty-good routes, even if they're not the best.

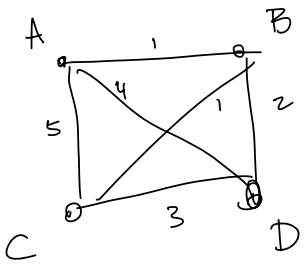
"approximate algorithms"

↑ not nec. best, but good enough.

1st approx. alg for TSP

The Nearest-Neighbor Algorithm.

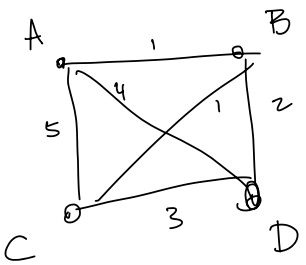
Much faster than brute-force, although may not always give the best answer.

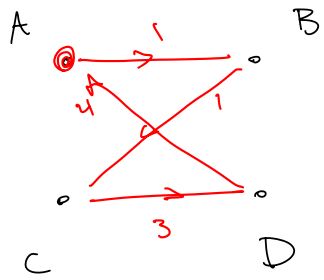


Find a solution starting at A with NN

From the starting point, choose edges as you go, each time use the cheapest legal edge.

↑ don't revisit any verts until the end.

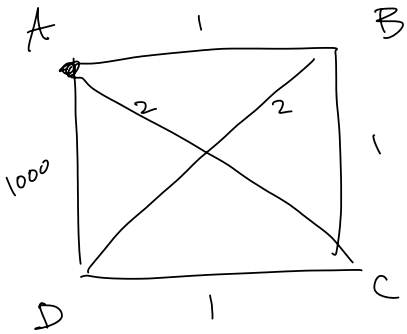




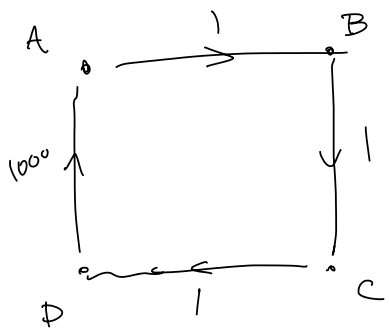
ABCDA

weight: 1+1+3+4 = 9

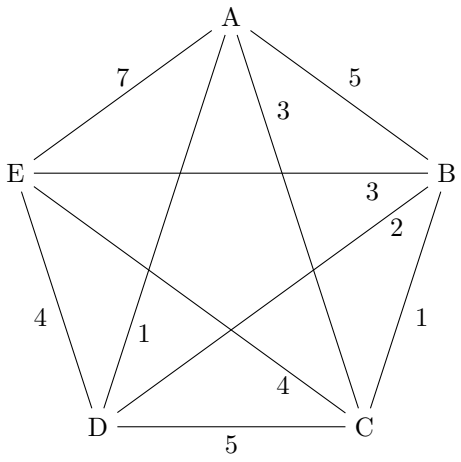
Much faster than brute force, but doesn't always pick the best:



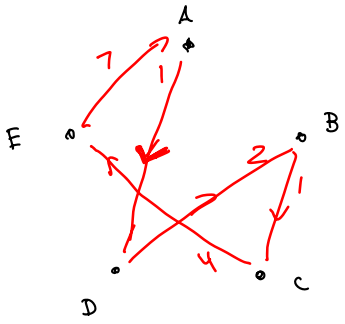
NN starting at A:



Here, NN chooses a bad answer.



Find the circ using NN at A.



ADBCEA → 15