Graph Implementation Time/Space Complexity

n = # vertices
m = # edges

	Adj Matrix	Adj List	Adj Set (Hash)
Space	$O(n^2)$	$O(n^2)$ worst case $O(n+m)$	$O(n+m)$
has_edge	$O(1)$	$O(n)$ worst case $O(\text{outdegree}(u))$	$O(1)$ expected
add_edge precondition: edge doesn't exist	$O(1)$	$O(1)$ amortized	$O(1)$ expected
for_each_out_neighbor	$O(n)$	$O(n)$ worst case $O(\text{outdegree}(u))$	$O(n)$ worst-case $O(\text{outdegree}(u))$
for each vertex for_each_out_neighbor	$O(n^2)$	$O(n+m)$	$O(n+m)$

$0 \le m \le n \cdot (n-1)$

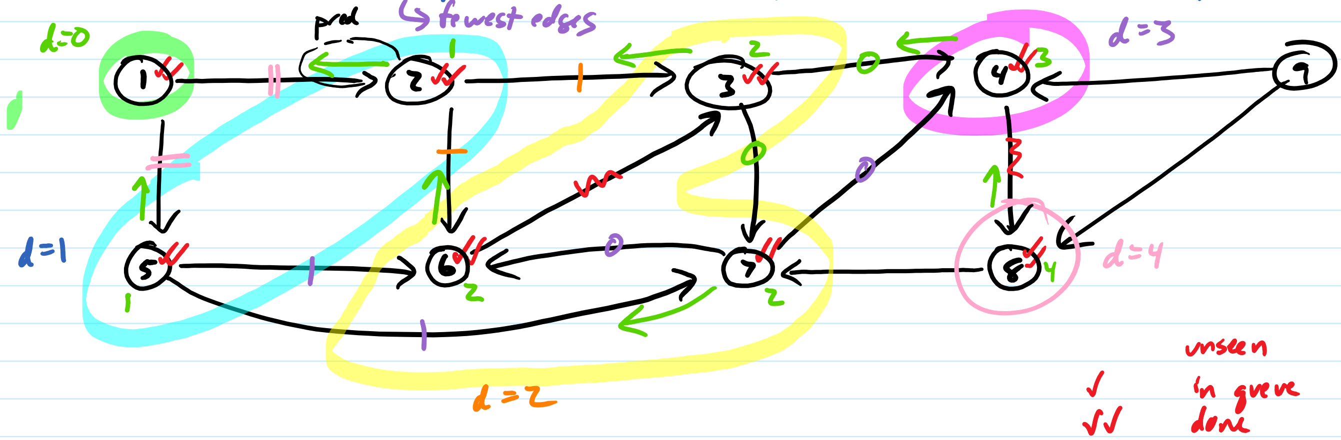
dense

sparse: m is $O(n)$

Breadth-First Search

↳ finds <u>shortest</u> paths from starting point in unweighted graph

↳ fewest edges



queue : ~~1~~ ~~2~~ ~~5~~ ~~3~~ ~~6~~ ~~7~~ ~~4~~ ~~8~~

curr : ~~1~~ ~~2~~ 5

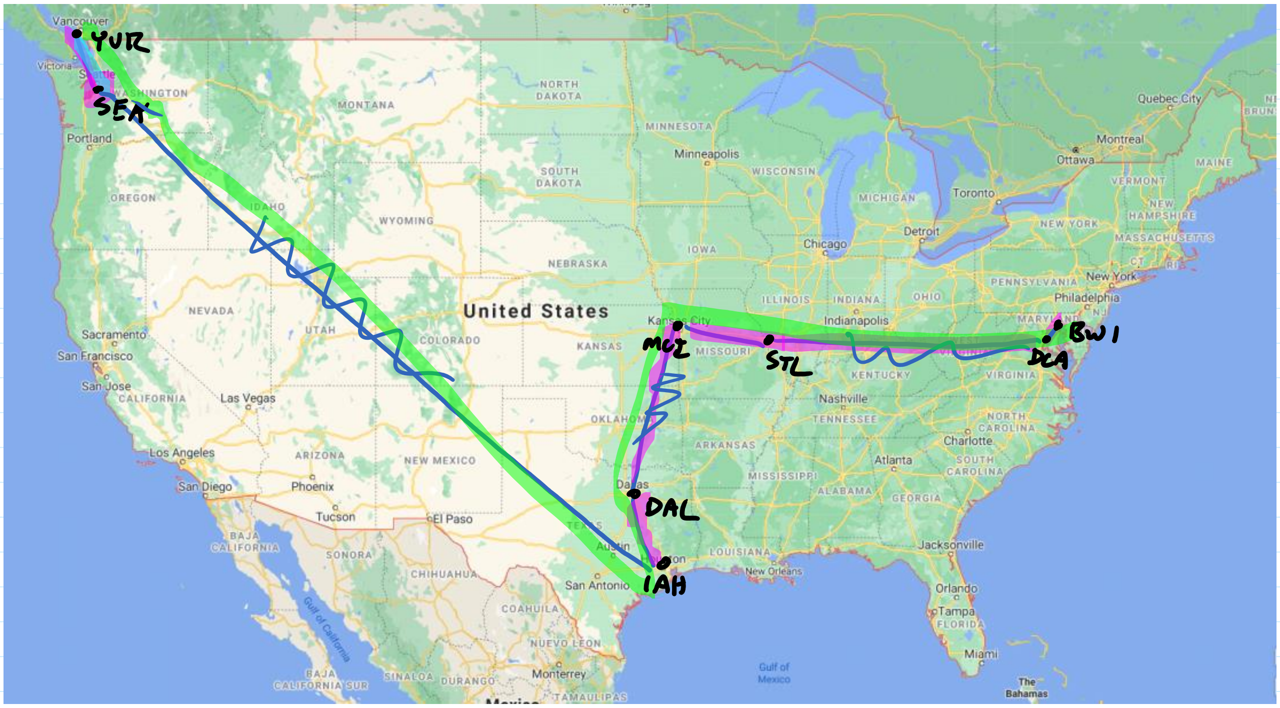
```
Q ← [start]
color[start] ← in queue
d[start] ← 0
pred[start] ← NULL
while Q not empty
    u ← dequeue(Q)
    for each outneighbor v of u
        if color[v] == unseen
            enqueue(Q, v)
            color[v] ← in queue
            π[v] ← u
            d[v] ← d[u] + 1
    color[u] ← done
```

for each vertex (in order of increasing d)
for each outneighbor

$O(n+m)$ using adj list

minimum spanning tree (MST)

SEA YVR MCI STL DCA BWI DAL IAH



DCA BWI
SEA YVR
MCI STL
DAL IAH
MCI DAL
STL DCA
SEA IAH

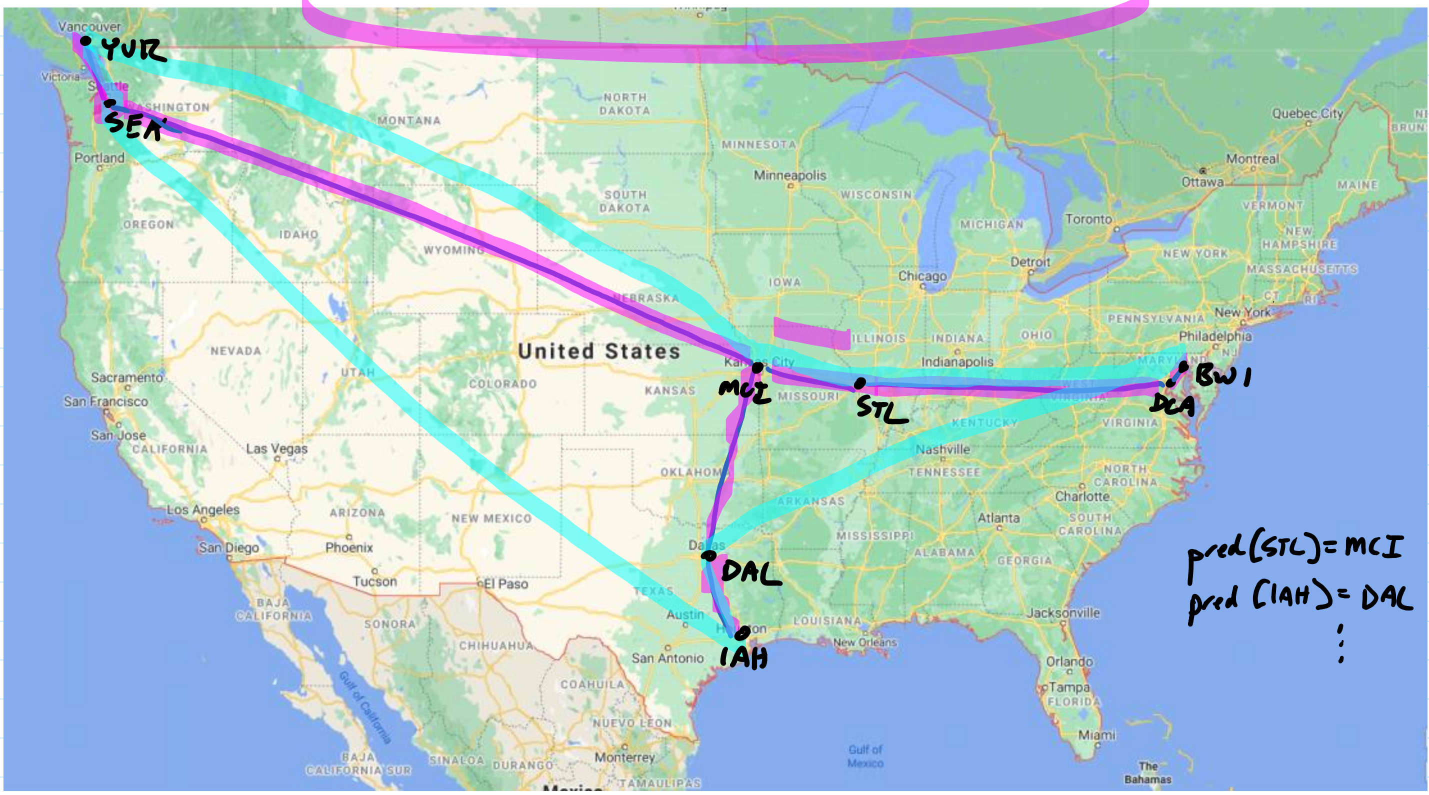
use DFS to order edges into tour

Depth-first search: backtracking graph exploration

Thursday, November 12, 2020 1:13 PM

minimum spanning tree (MST)

SEA YVR MCI STL DCA BWI DAL IAH



Depth-first search: backtracking graph exploration