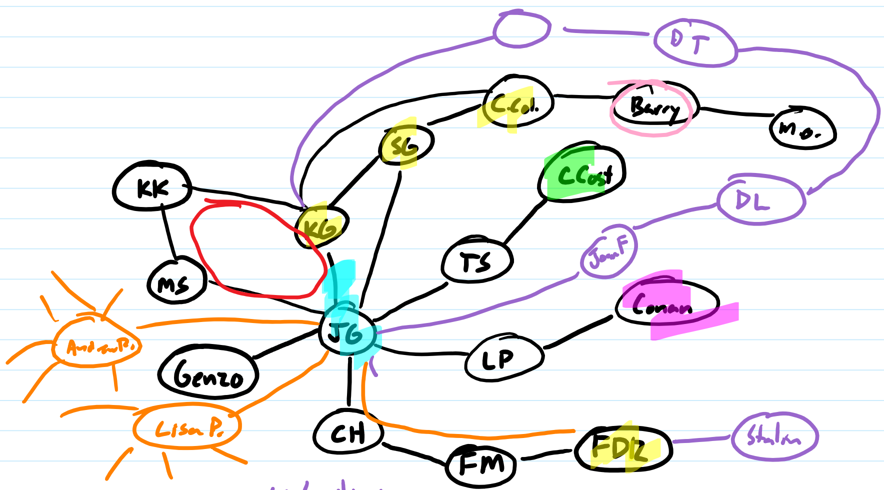
Graphs



for any two vertices, is there a path between of length ≤ 6

↓

edges on path

nodes/vertices

edges

Graph: represents things and relationships between them

people

relations

path: sequence of vertices w/ edges between JG, CH, FM, FDR

simple path: no repeats

cycle: starts/ends at same place JG, MS, KK, KG, JG

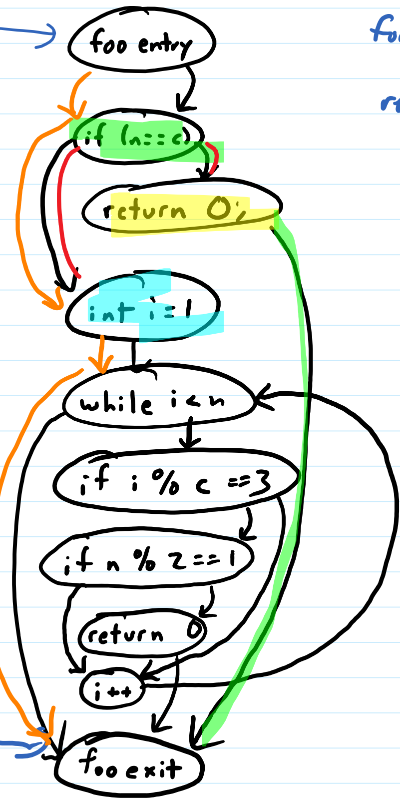
simple cycle: only repeat at beginning and end

```
int foo(int n, int c)
{
    if (n == c)
    {
        return 0;
    }
    int i = 1;
    while (i < n)
    {
        if (i % c == 3)
        {
            if (n % 2 == 1)
            {
                return 0;
            }
        }
        i++;
    }
}
```

vertices: lines of code

edges: control flow

directed



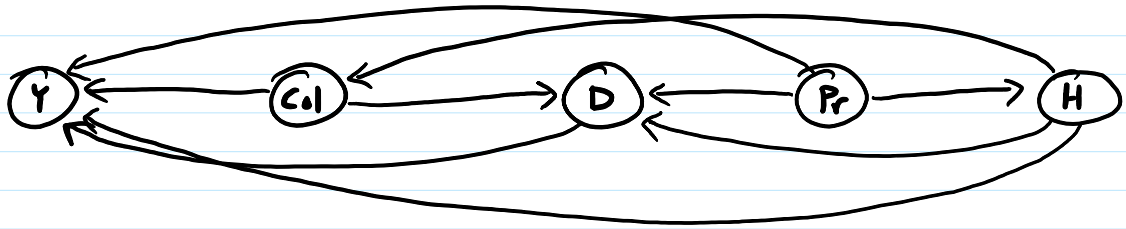
foo entry, if n==c, return 0

return 0, ~~if n==c, foo~~ entry

is there a path from entry to exit that doesn't go through return?

YES - missing return!

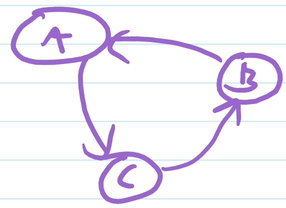
verts: teams   edge $u \to v$: u lost to v



can we order teams so edges go in same direction

vertices

edges



A B C

what ordering minimizes edges in wrong direction?

upsets

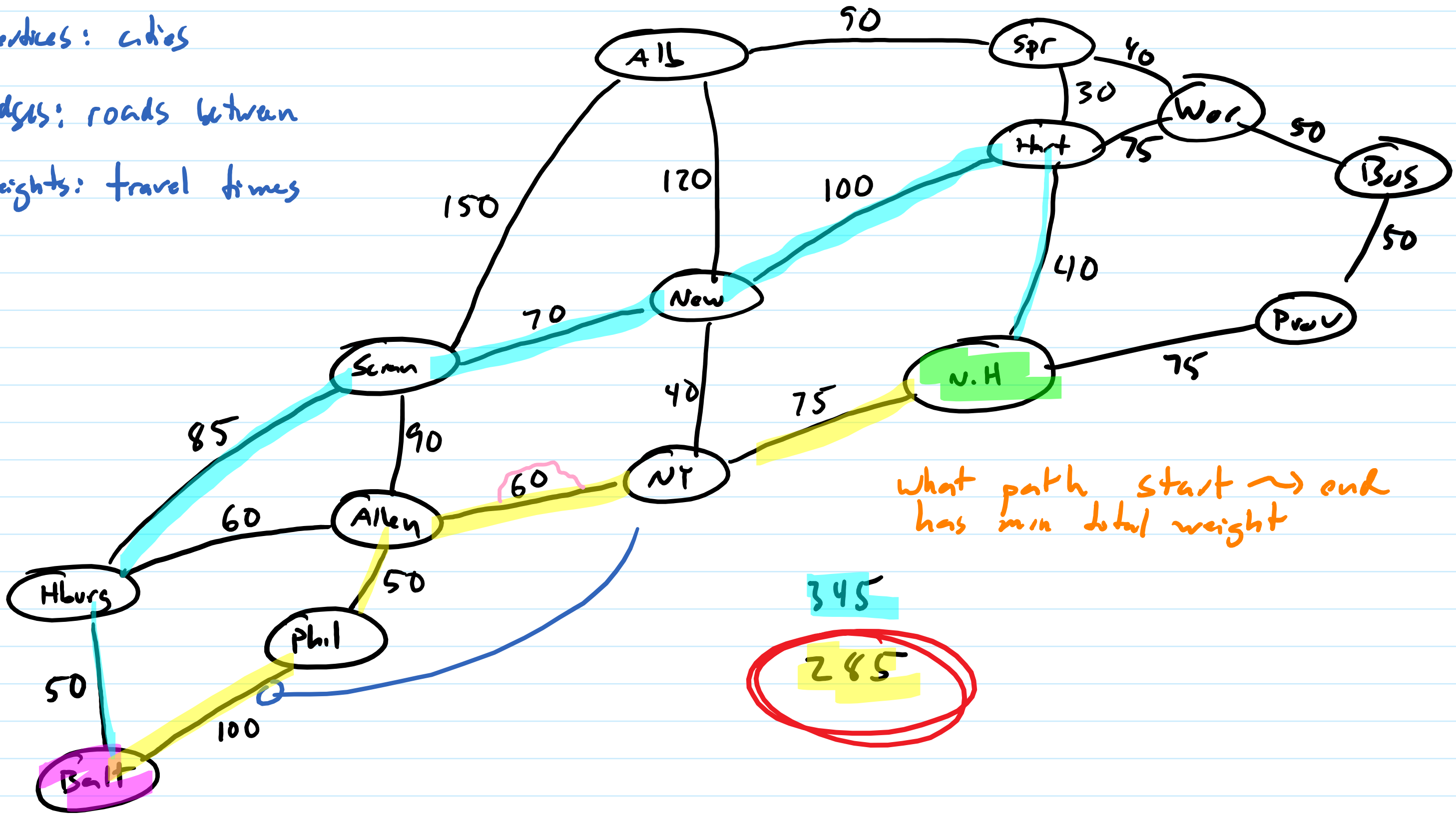
MIN-FEEDBACK-ARC-SET

NP-COMPLETE

vertices: cities

edges: roads between

weights: travel times



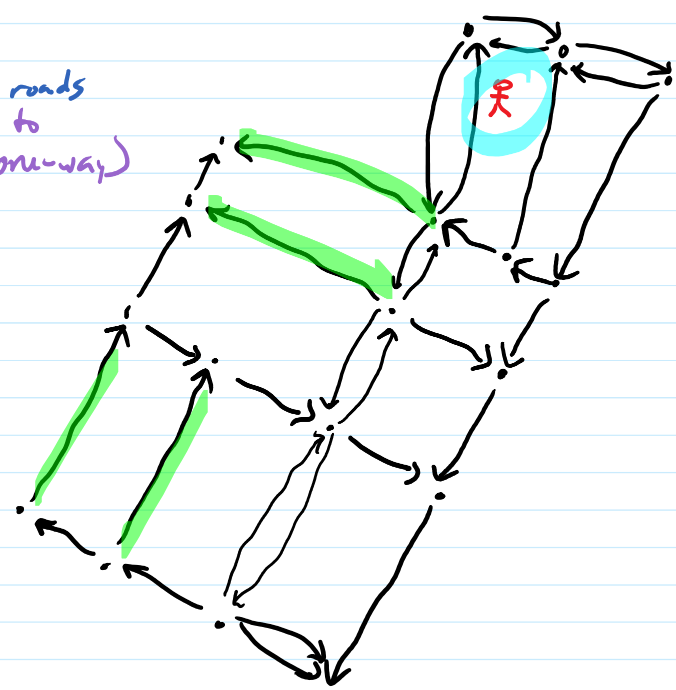
what path start ⇝ end has min total weight

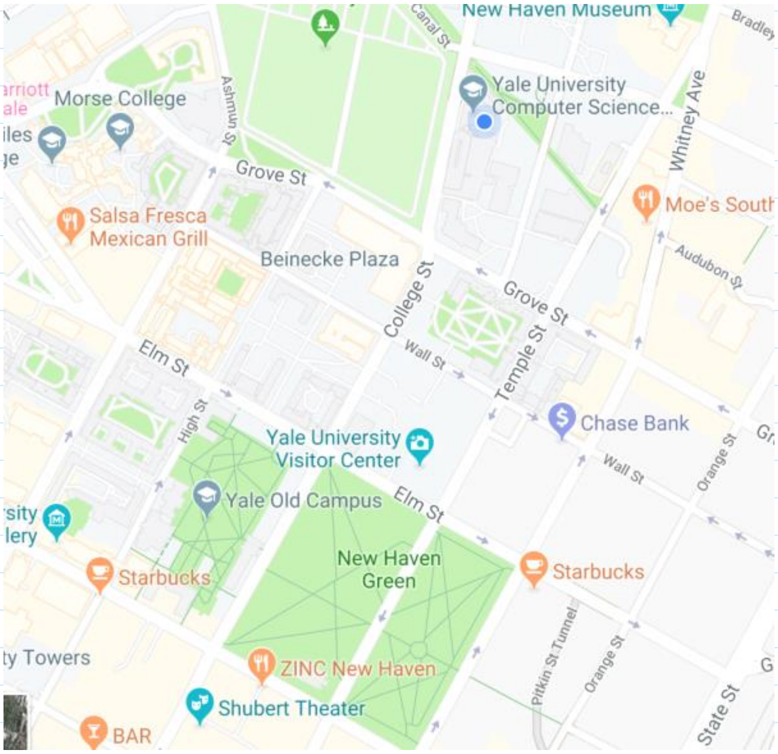
345

245

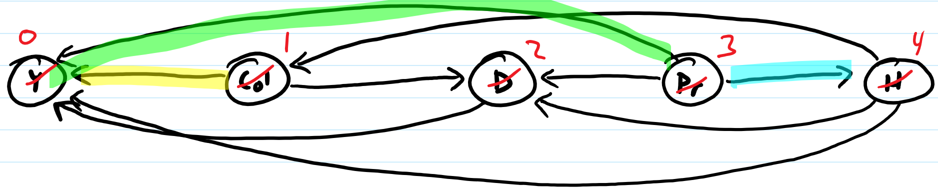
vertices: intersections

edges: segments of roads (w/ direction to represent one-way)




New Haven Museum
Yale University
Computer Science...
Whitney Ave
Morse College
Ashmun St
Grove St
Moe's South
Salsa Fresca
Mexican Grill
Beinecke Plaza
Audubon St
College St
Grove St
Wall St
Temple St
Elm St
Chase Bank
High St
Yale University
Visitor Center
Wall St
Orange St
Yale Old Campus
Elm St
New Haven
Green
Starbucks
Starbucks
ty Towers
ZINC New Haven
Pitkin St Tunnel
Orange St
State St
Shubert Theater
BAR

Graph Representation



Adjacency Matrix — 2D array s.t. adj[r][c] is T if edge $r \to c$

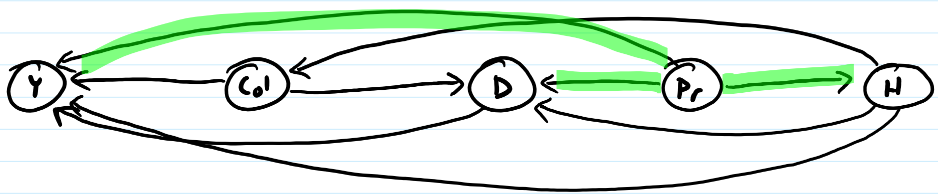
to

from		Y 0	Col 1	D 2	Pr 3	H 4
Y	0	F	F	F	F	F
Col	1	T	F	T	F	F
D	2	T	F	F	F	F
Pr	3	T	F	T	F	T
H	4	F	T	T	F	F

has_edge(H, Col)?

map

Y	0
Col	1
D	2
Pr	3
H	4



Adjacency List — list of lists s.t. Adj[v] is list of verts u s.t. $v \to u$ is an edge

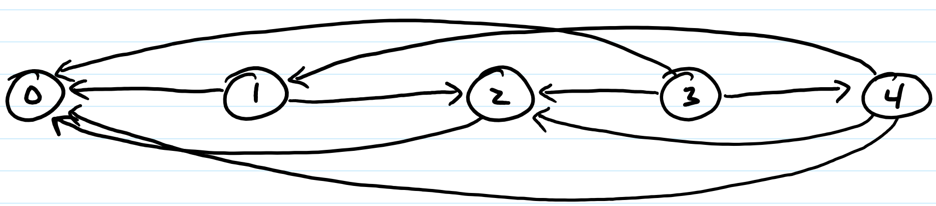
Y :

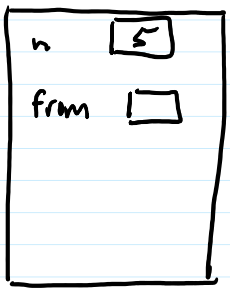
Col : Y D

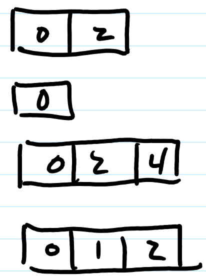
D : Y

Pr : Y D H

H : Y Col D




n
5
from


0 2
0
0 2 4
0 1 2

Graph Implementation Time/Space Complexity

n = # vertices
m = # edges

	Adj Matrix	Adj List
space	$O(n^2)$	~~$O(n^2)$~~ $O(n+m)$ better for sparse graphs
has_edge	$O(1)$	$O(n)$ (seq. search) $O(\text{degree}(n))$ $O(\text{outdegree}(n))$
add_edge	$O(1)$	$O(n)$ (has_edge to avoid duplicates)
for_each_neighbor	$O(n)$	$O(n)$ $O(\text{degree}(n))$
for each vertex v for each neighbor of v	$O(n^2)$	~~$O(n^2)$~~ worst case $O(n+m)$ → better for sparse graphs

$0 \le m \le n^2$

sparse → $m \ll n^2$ (ex. m is $O(n)$)

$$\sum_v \sum 1 + \text{degree}(v)$$
$$= \sum_v 1 + \sum_v \text{degree}(v)$$
$$= n + m$$