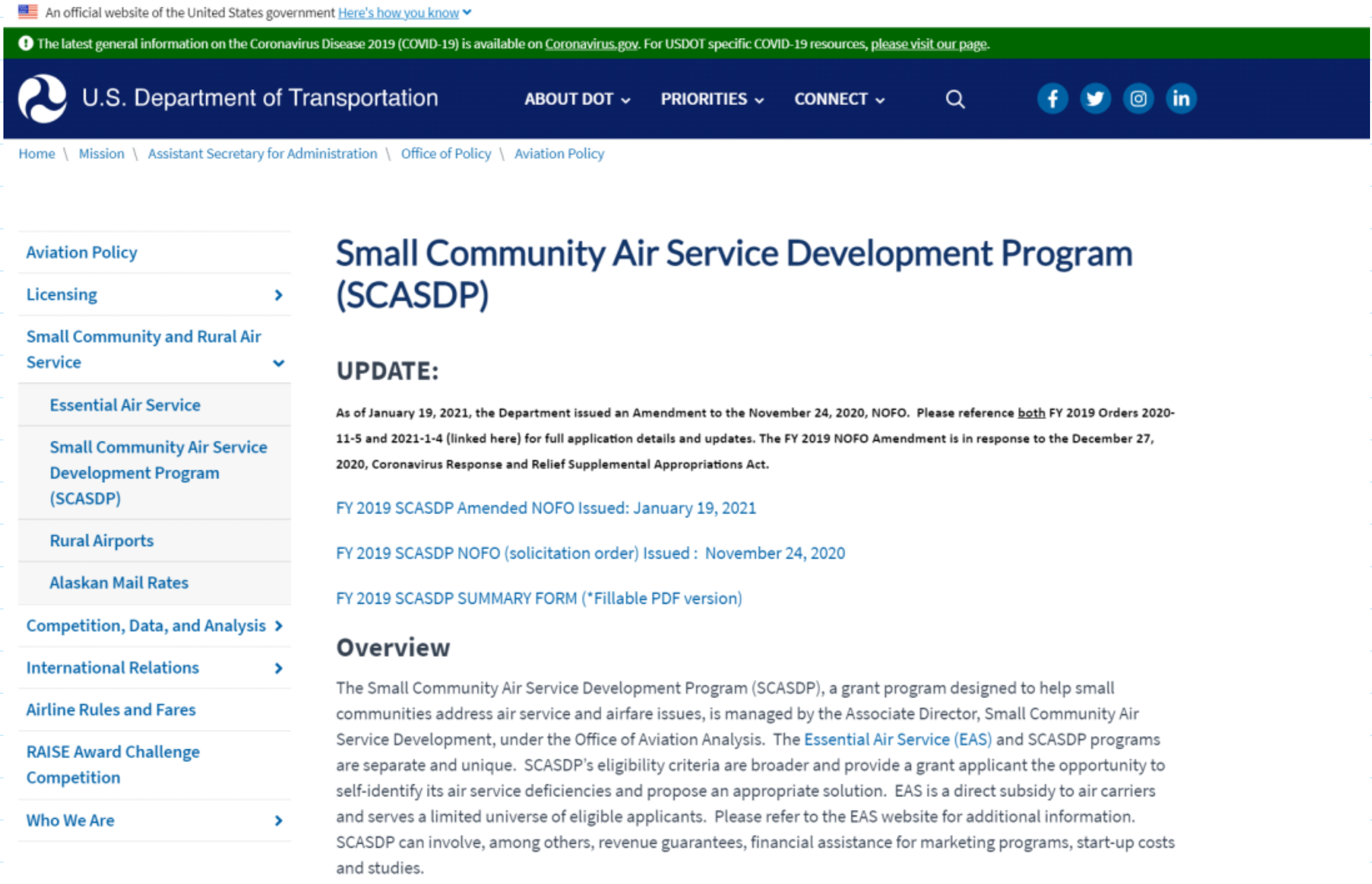
SCASDP



An official website of the United States government Here's how you know

The latest general information on the Coronavirus Disease 2019 (COVID-19) is available on Coronavirus.gov. For USDOT specific COVID-19 resources, please visit our page.

U.S. Department of Transportation
ABOUT DOT
PRIORITIES
CONNECT

Home \ Mission \ Assistant Secretary for Administration \ Office of Policy \ Aviation Policy

Aviation Policy
Licensing
Small Community and Rural Air Service
Essential Air Service
Small Community Air Service Development Program (SCASDP)
Rural Airports
Alaskan Mail Rates
Competition, Data, and Analysis
International Relations
Airline Rules and Fares
RAISE Award Challenge Competition
Who We Are

Small Community Air Service Development Program (SCASDP)

UPDATE:

As of January 19, 2021, the Department issued an Amendment to the November 24, 2020, NOFO. Please reference both FY 2019 Orders 2020-11-5 and 2021-1-4 (linked here) for full application details and updates. The FY 2019 NOFO Amendment is in response to the December 27, 2020, Coronavirus Response and Relief Supplemental Appropriations Act.

FY 2019 SCASDP Amended NOFO Issued: January 19, 2021

FY 2019 SCASDP NOFO (solicitation order) Issued : November 24, 2020

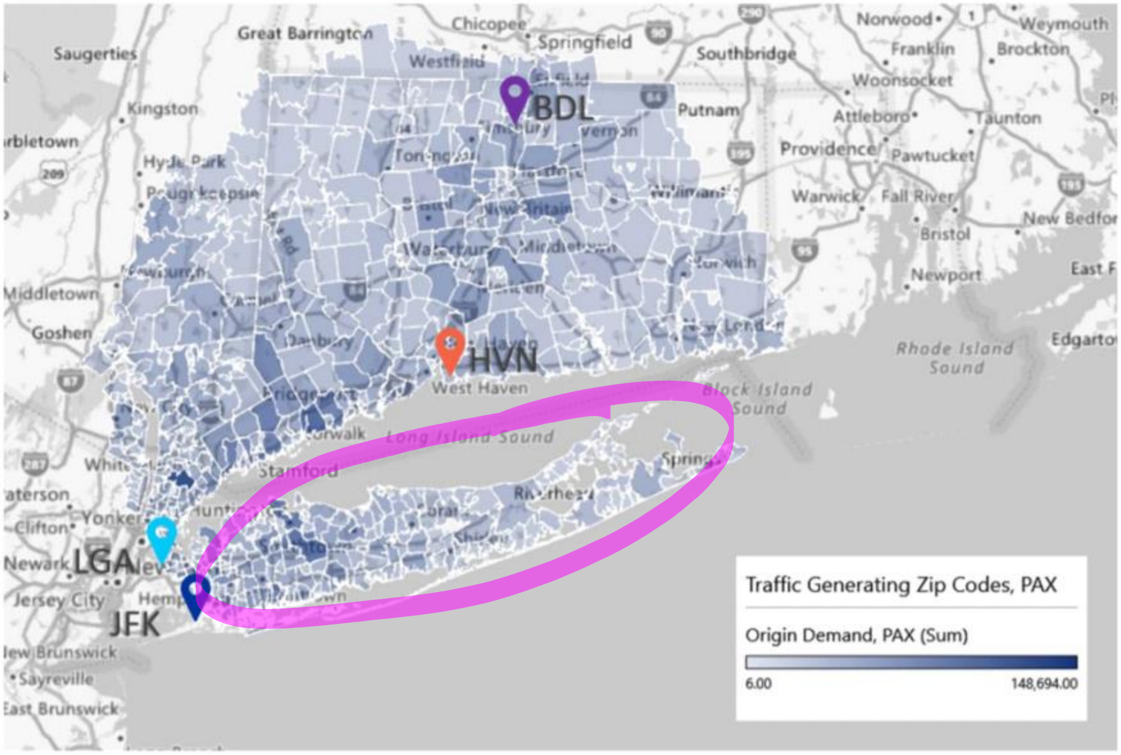
FY 2019 SCASDP SUMMARY FORM (*Fillable PDF version)

Overview

The Small Community Air Service Development Program (SCASDP), a grant program designed to help small communities address air service and airfare issues, is managed by the Associate Director, Small Community Air Service Development, under the Office of Aviation Analysis. The Essential Air Service (EAS) and SCASDP programs are separate and unique. SCASDP's eligibility criteria are broader and provide a grant applicant the opportunity to self-identify its air service deficiencies and propose an appropriate solution. EAS is a direct subsidy to air carriers and serves a limited universe of eligible applicants. Please refer to the EAS website for additional information. SCASDP can involve, among others, revenue guarantees, financial assistance for marketing programs, start-up costs and studies.

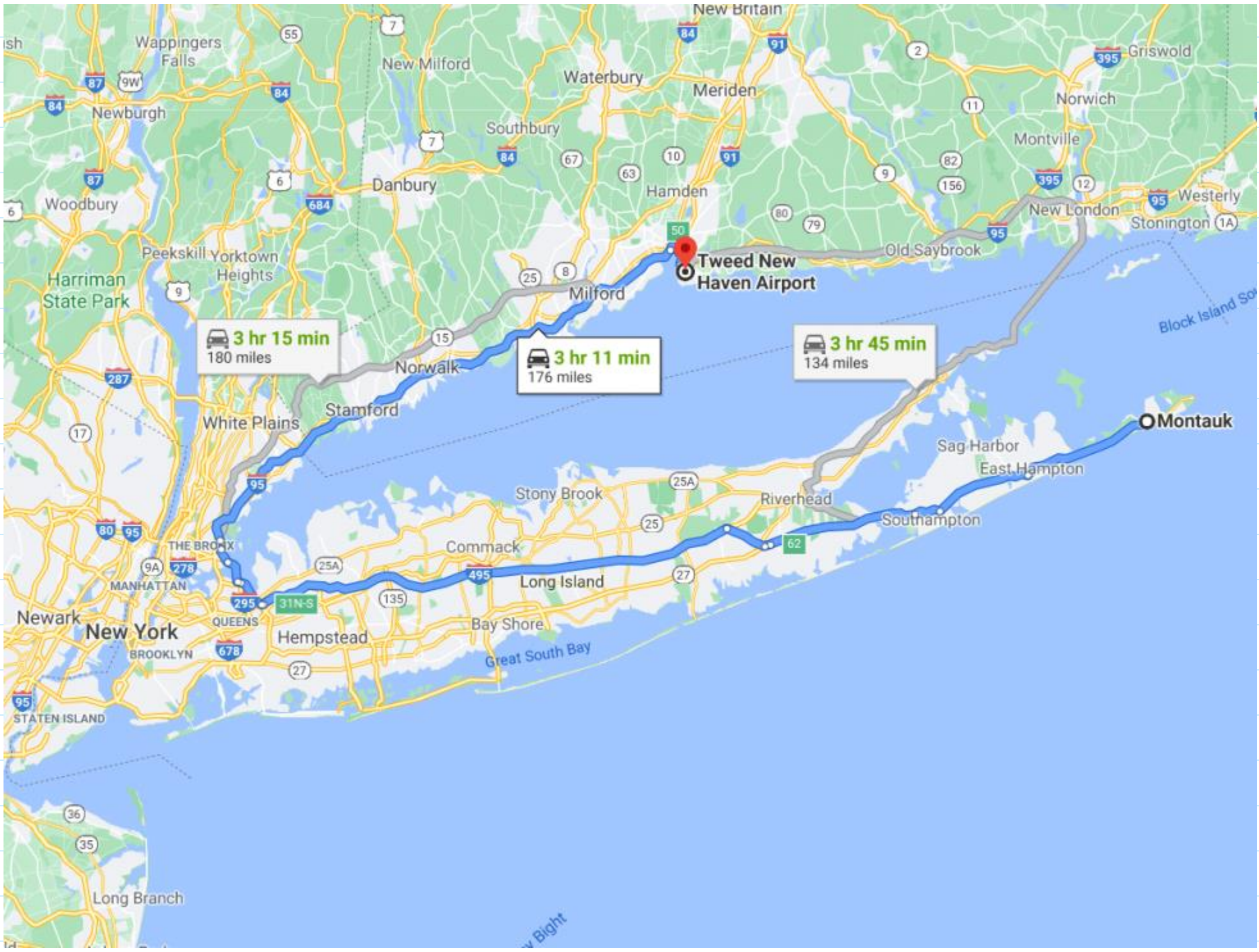
60-mile Radius

Figure 7. HVN's Catchment Traffic Generating Zip Codes within 60 mi of HVN (Resident/Origin Demand)

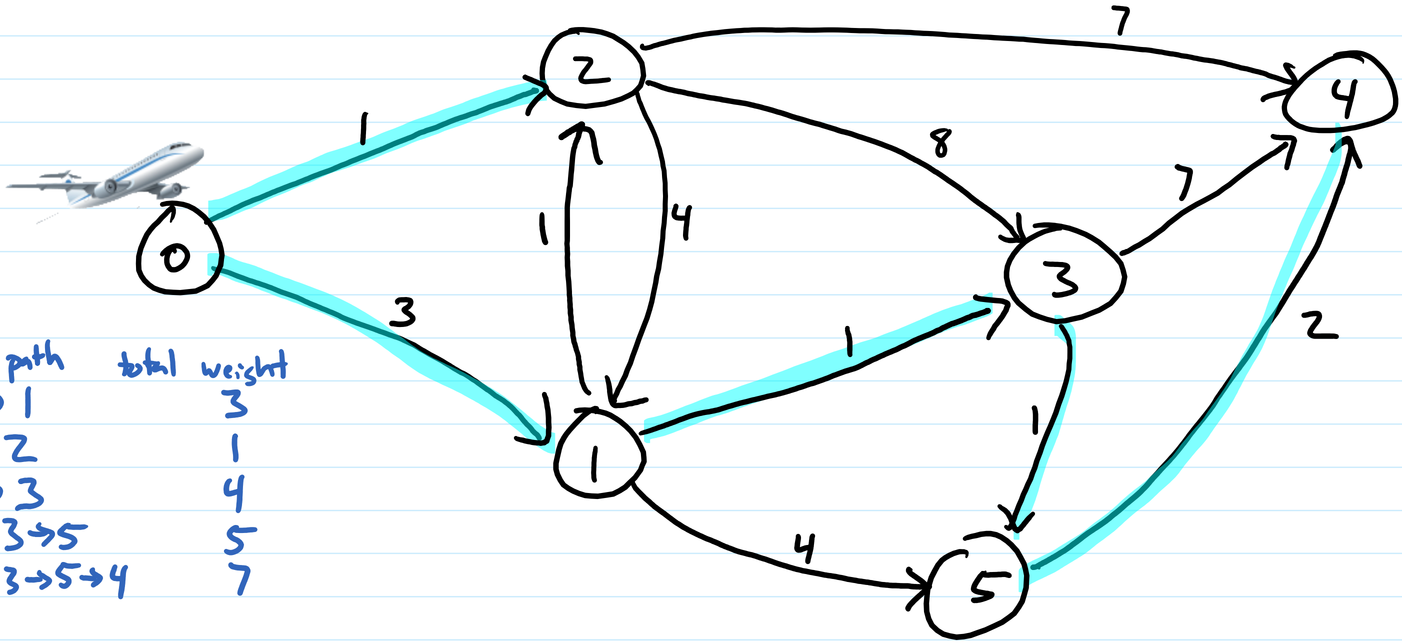


Source: ASM's HVN Catchment/Leakage Analysis, YE1Q2018, Airline Data Inc.

Long Islang is a long drive from HVN



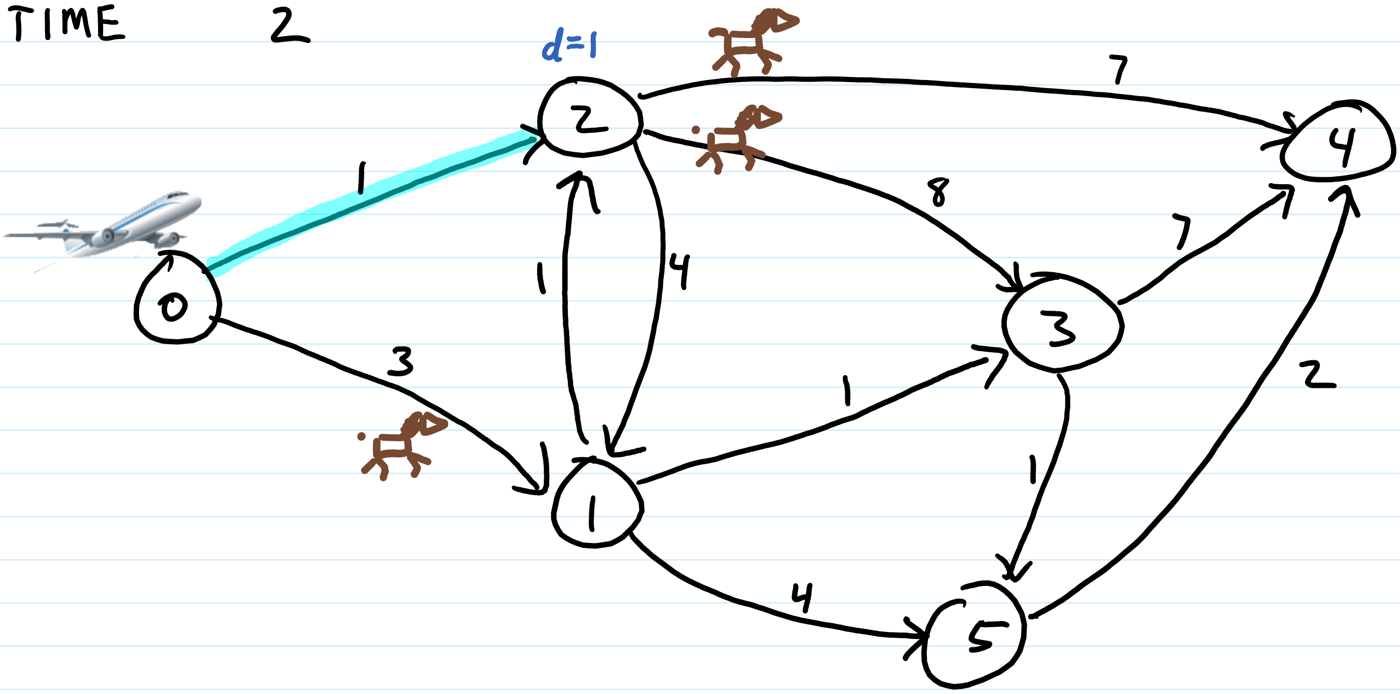
Shortest Paths



shortest path	total weight
0 → 1	3
0 → 2	1
0→1→3	4
0→1→3→5	5
0→1→3→5→4	7

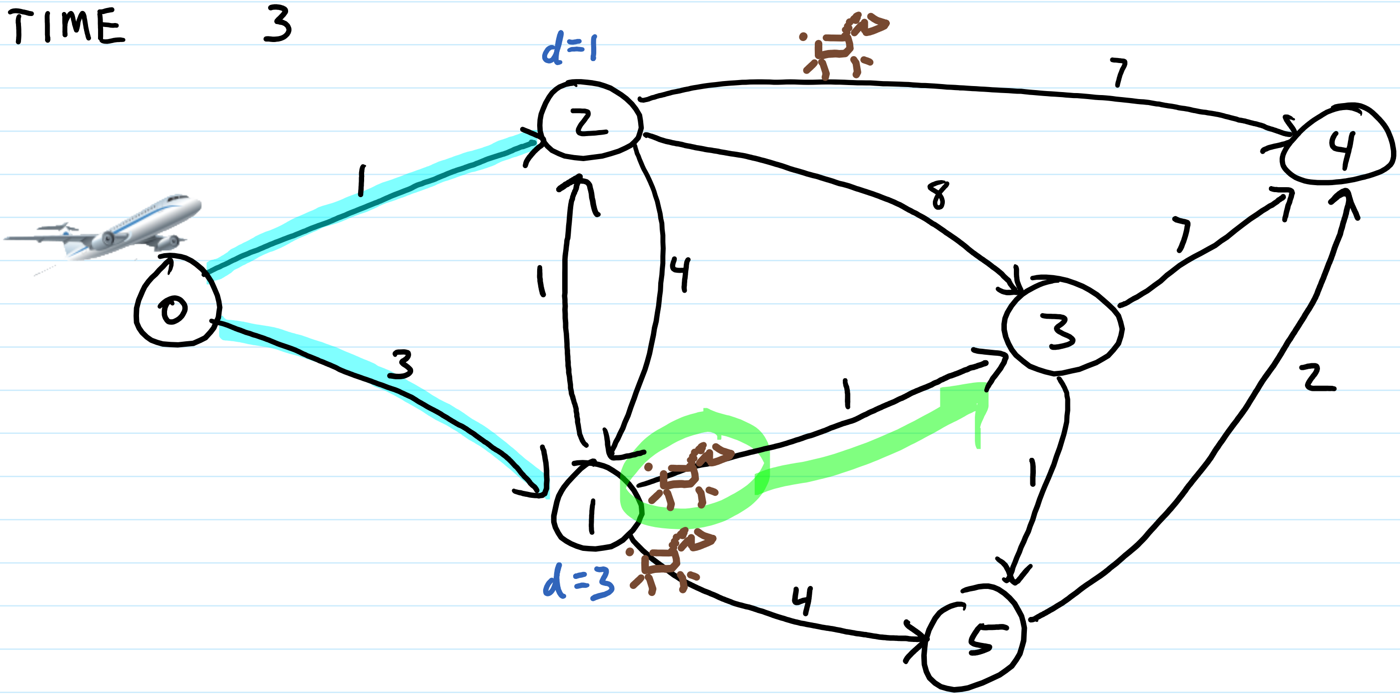
Shortest Paths

TIME 2



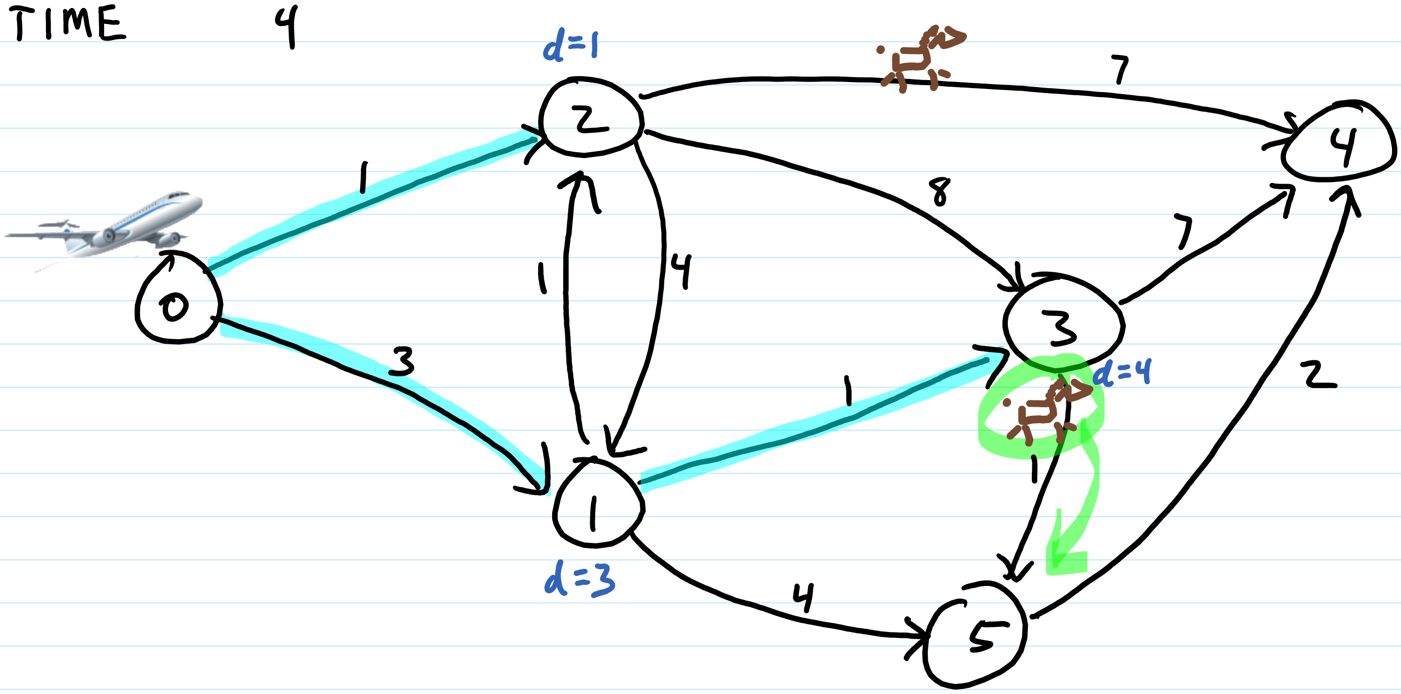
Shortest Paths

TIME 3



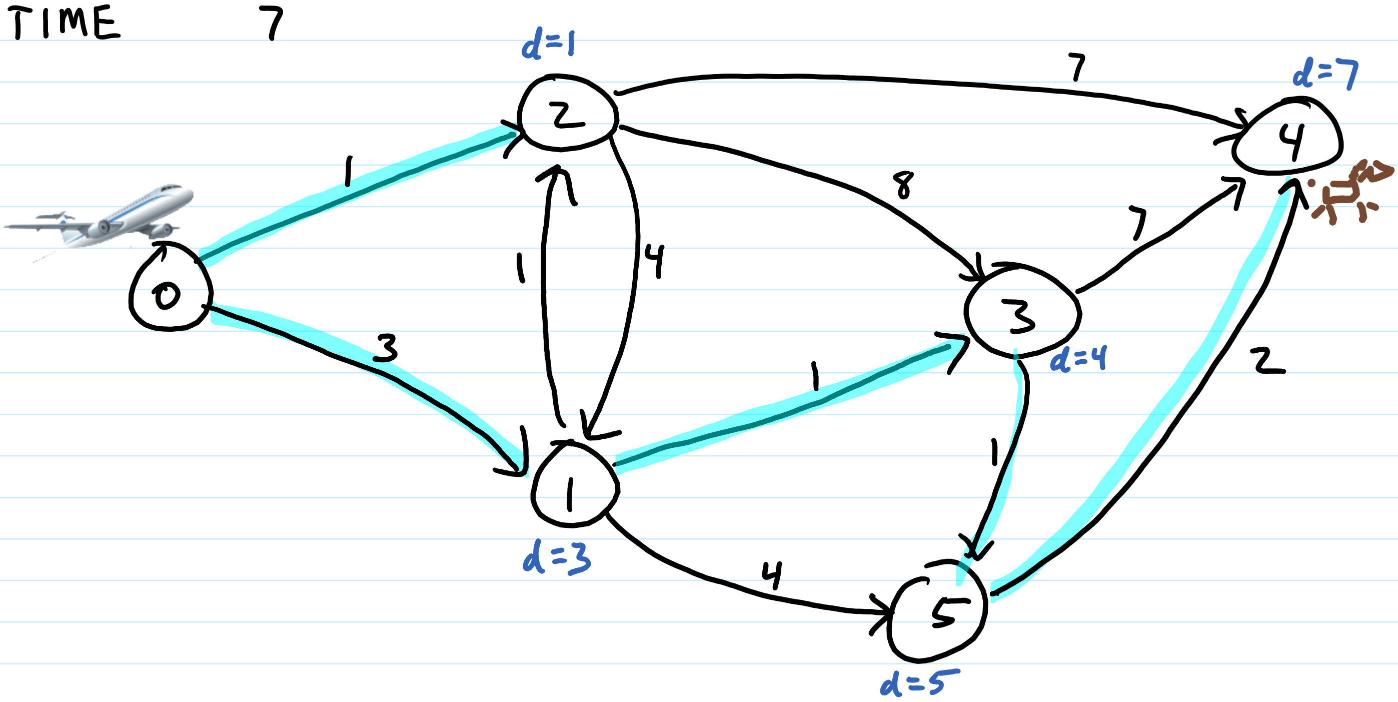
Shortest Paths

TIME 4



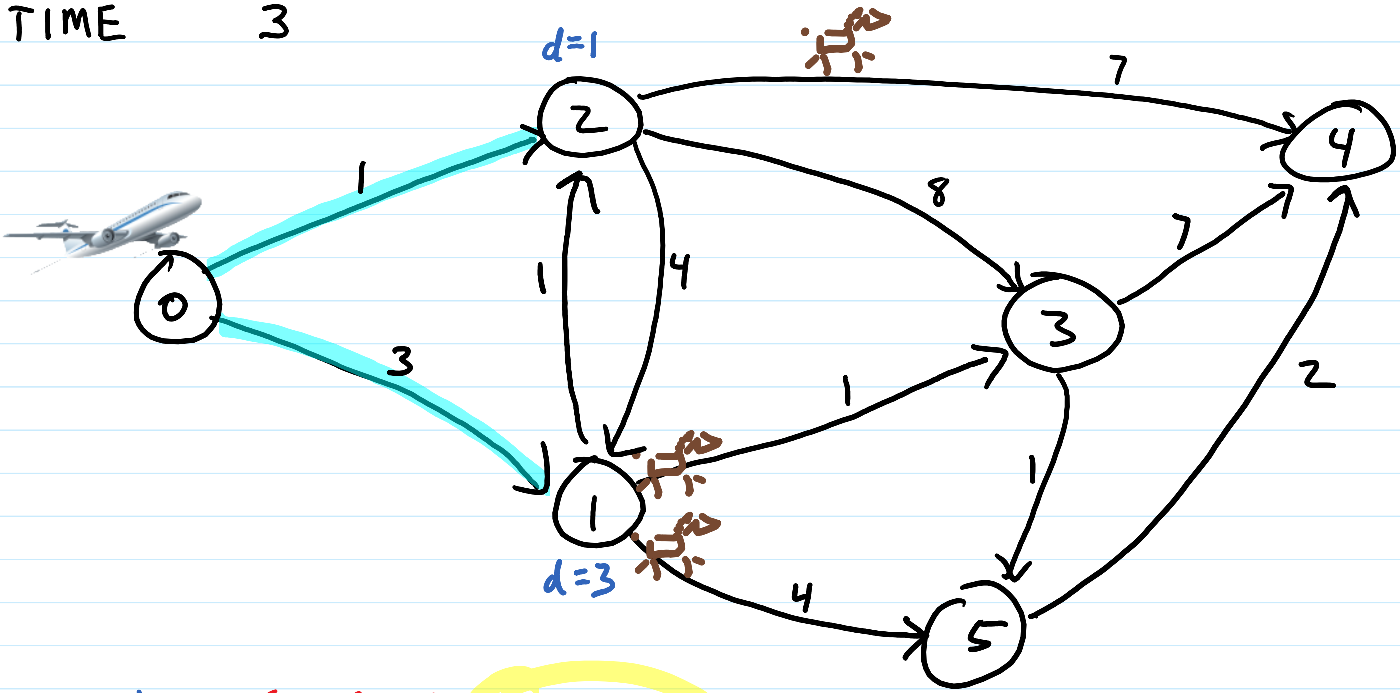
Shortest Paths

TIME 7



Shortest Paths

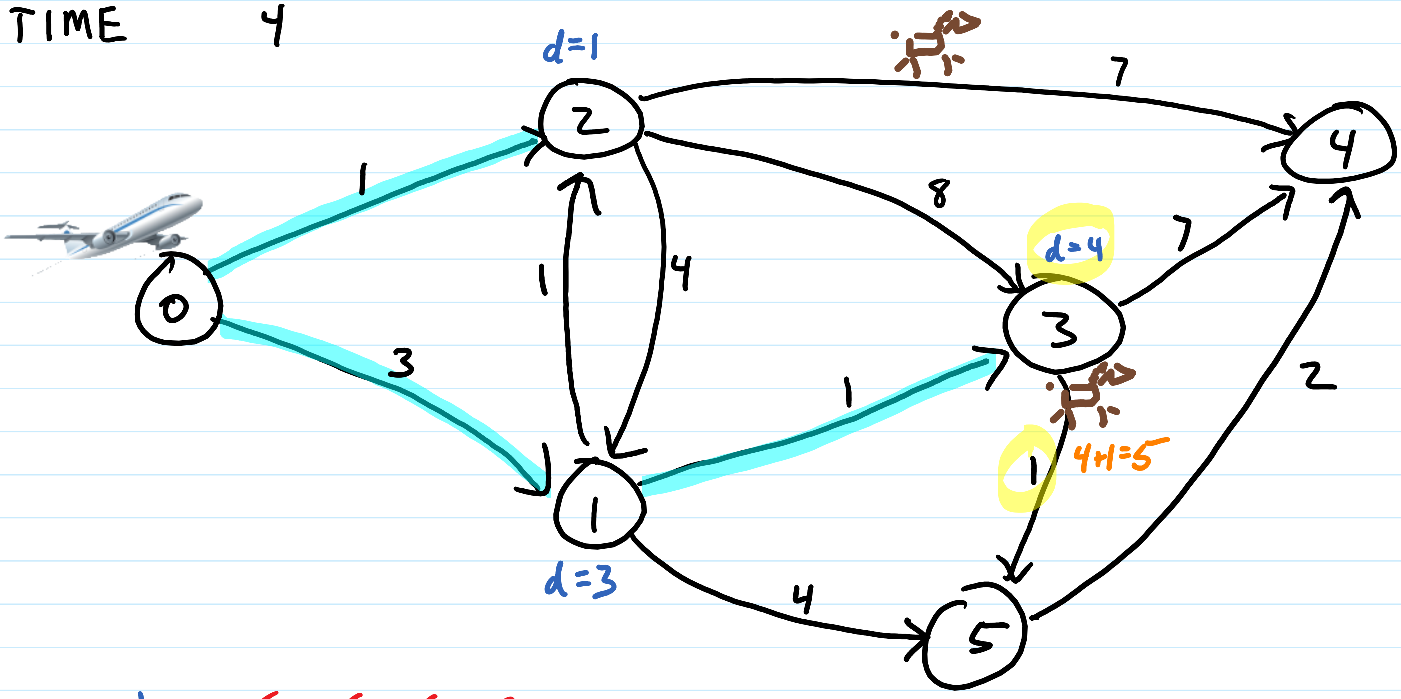
TIME 3



vertex	~~0~~	~~1~~	~~2~~	3	4	5
d	0	3	1	4	8	7
pred	–	0	0	1	2	1

Shortest Paths

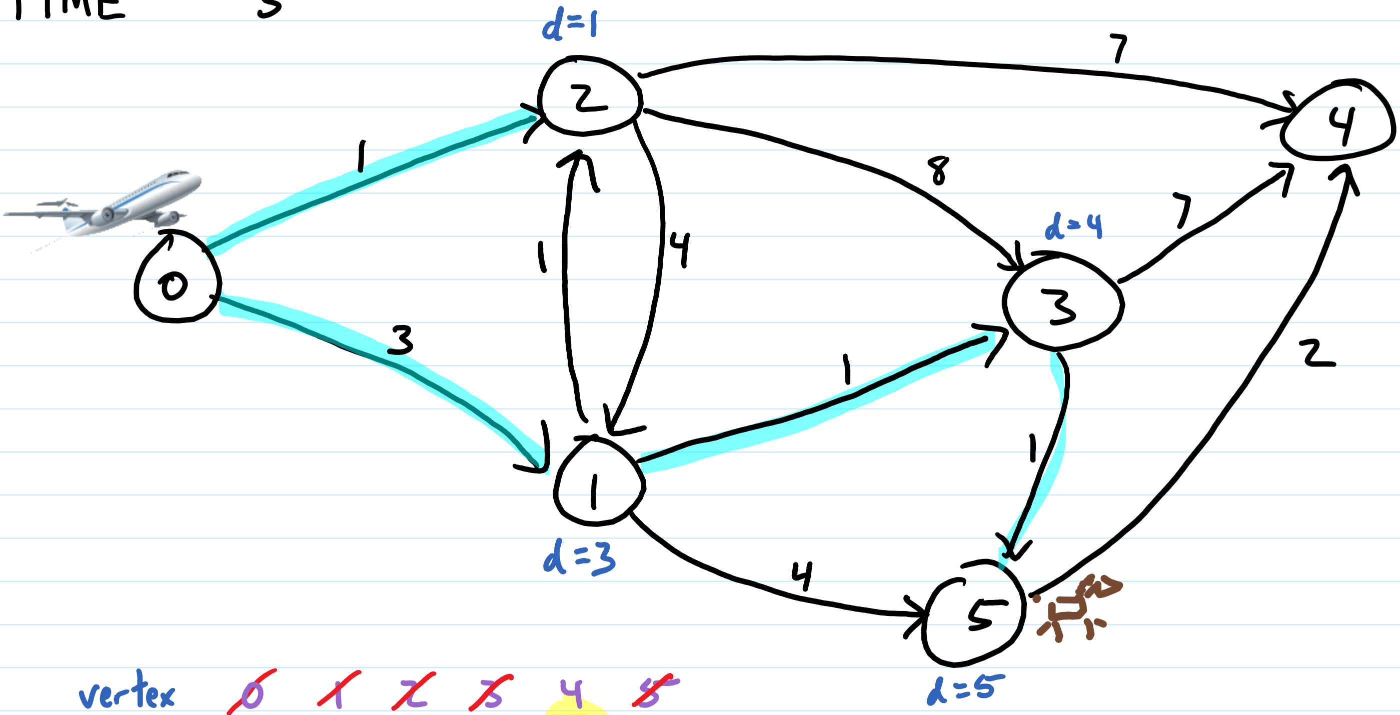
TIME 4



vertex	~~0~~	~~1~~	~~2~~	~~3~~	4	5
d	0	3	1	4	8	5
pred	–	0	0	1	2	3

Shortest Paths

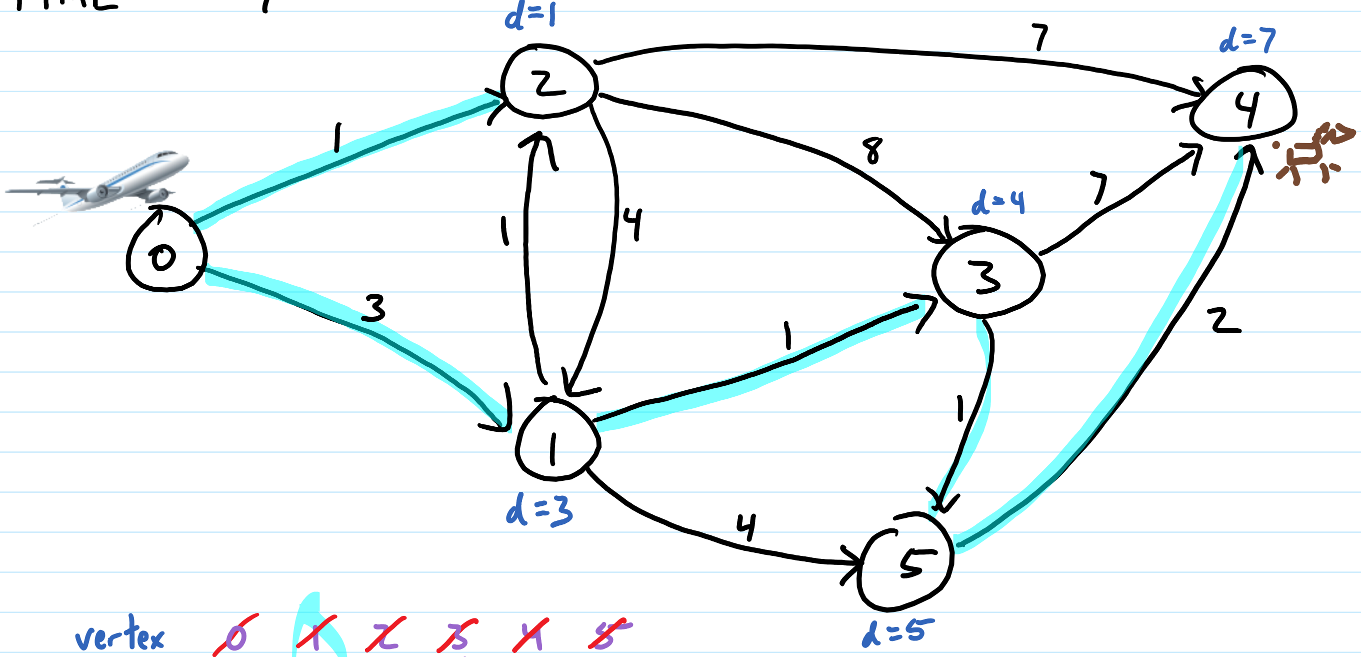
TIME 5



vertex	~~0~~	~~1~~	~~2~~	~~3~~	4	~~5~~
d	0	3	1	4	7	5
pred	–	0	0	1	5	3

Shortest Paths

TIME 7

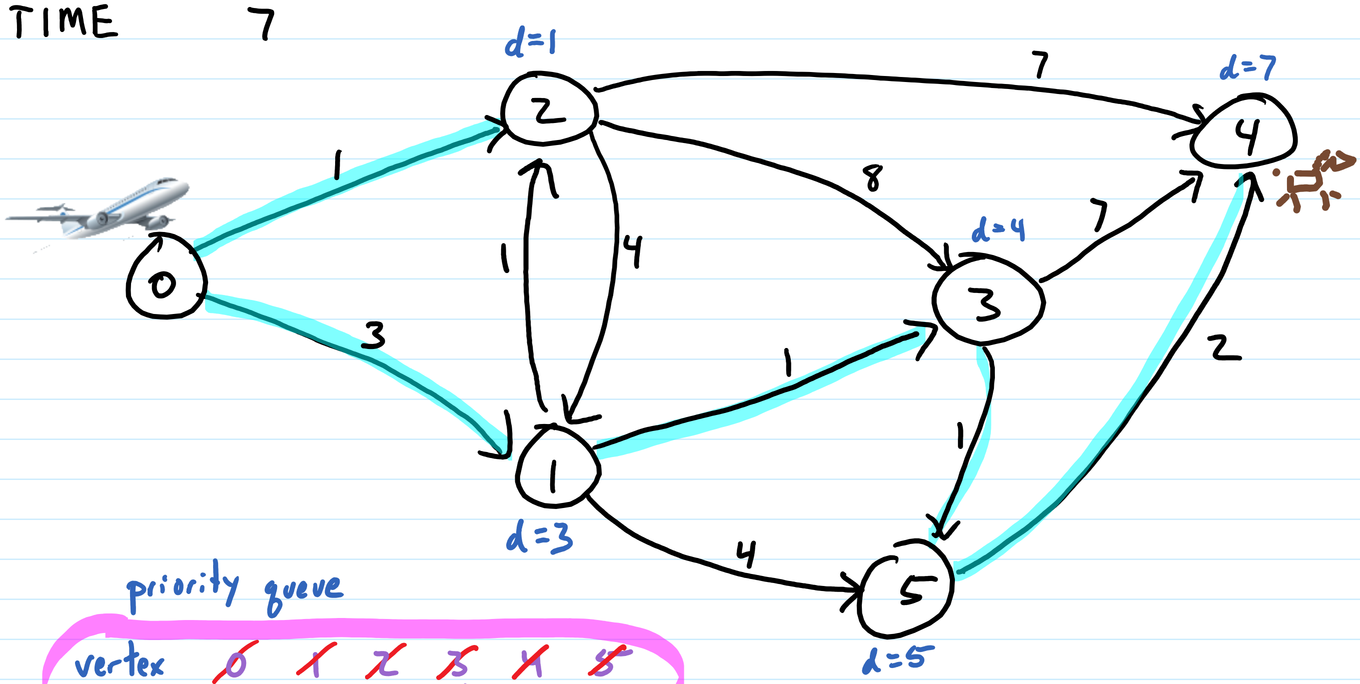


vertex	~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~
d	0	3	1	4	7	5
pred	–	0	0	1	5	3

shortest path $0 \rightsquigarrow 4$: 0 1 3 5 4

Shortest Paths

TIME 7



priority queue

vertex	~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~
d	0	3	1	4	7	5
pred	–	0	0	1	5	3

shortest path 0 ⇝ 4 : 0 1 3 5 4

Dijkstra's Algorithm

PRE: no negative weight edges

POST: d gives total weight of shortest paths, pred gives edges in shortest paths
(∞ to mean unreachable)

```
for each v
  color[v], pred[v], d[v] ← IN_QUEUE, NIL, ∞
d[s] ← 0

Q ← new PriorityQueue(d)

while Q not empty                          one iteration per vertex
  u ← dequeue(Q)                           n calls
  for each outneighbor v of u              one iteration per edge
    if color[v] = IN_QUEUE and d[v] > d[u] + w(u, v)
≤ m calls change_priority(Q, v, d[u] + w(u, v))
      d[v] ← d[u] + w(u, v)
      pred[v] ← u
  color[u] ← DONE
```

adjacency list: $\Theta(n+m)$ + time for priority queue operations

Dijkstra's Algorithm

PRE: no negative weight edges

POST: d gives total weight of shortest paths, pred gives edges in shortest paths
(∞ to mean unreachable)

```
for each v
  color[v], pred[v], d[v] ← IN_QUEUE, NIL, ∞
d[s] ← 0

Q ← new PriorityQueue(d)

while Q not empty  one iteration per vertex
  u ← dequeue(Q)  n calls
  for each outneighbor v of u  one iteration per edge
    if color[v] = IN_QUEUE and d[v] > d[u] + w(u, v)
≤ m calls change_priority(Q, v, d[u] + w(u, v))
      d[v] ← d[u] + w(u, v)
      pred[v] ← u
  color[u] ← DONE
```

adjacency list: O(n+m) + time for priority queue operations