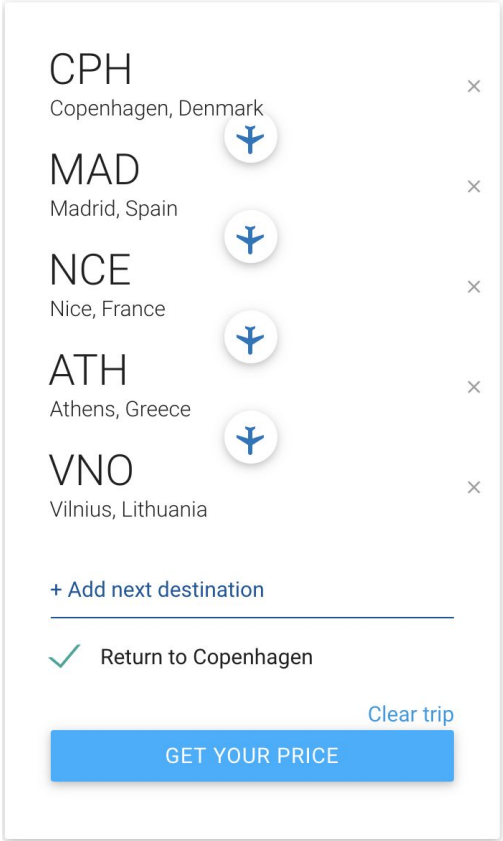
Connected flight route search

Using Skyscanner's Travel API

Services available today

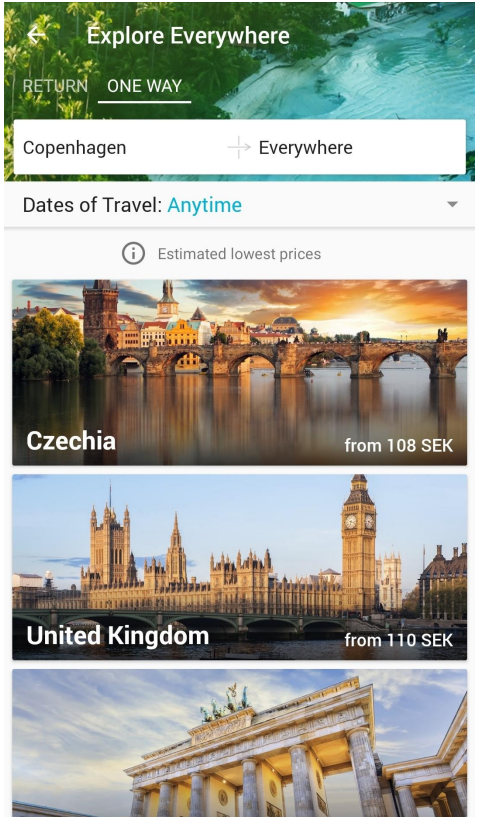
- AirTreks TripPlanner
- Kilroy
- Cheap Flights Finder
- Many others...

All these services require the user to choose every destination and date in their journey.



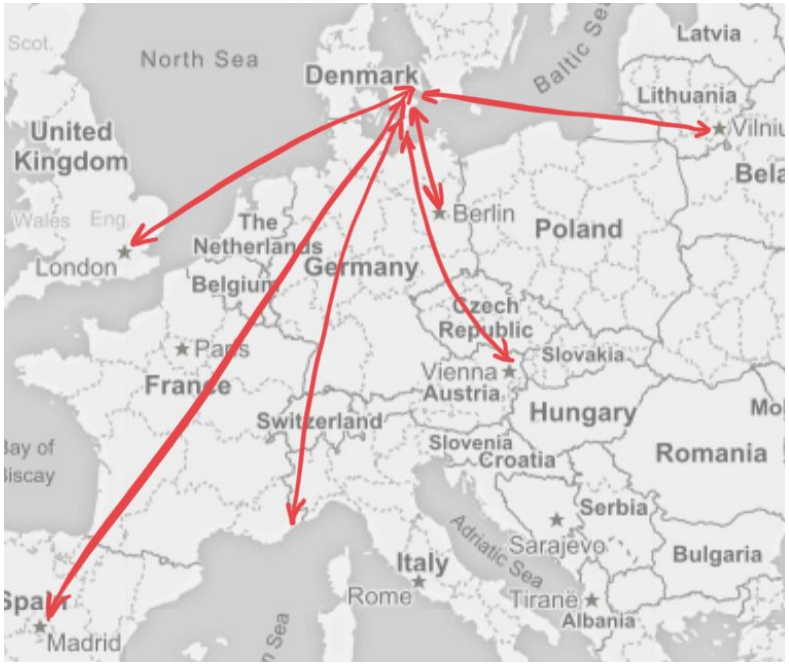
Skyscanner

- Flight search service, connected to 1200 travel partners
- “Explore Everywhere” - cheap flights from your nearest airport (one-way or return)
- **Open API** (via third party)



What if you want to explore the world?

- Round trips are inefficient
- Use Skyscanner API to construct routes with Python
- Say you:
  - Broke student and need to find the cheapest alternatives
  - Have vacation between two dates
  - Want to stay at each stop between a-b amount of days



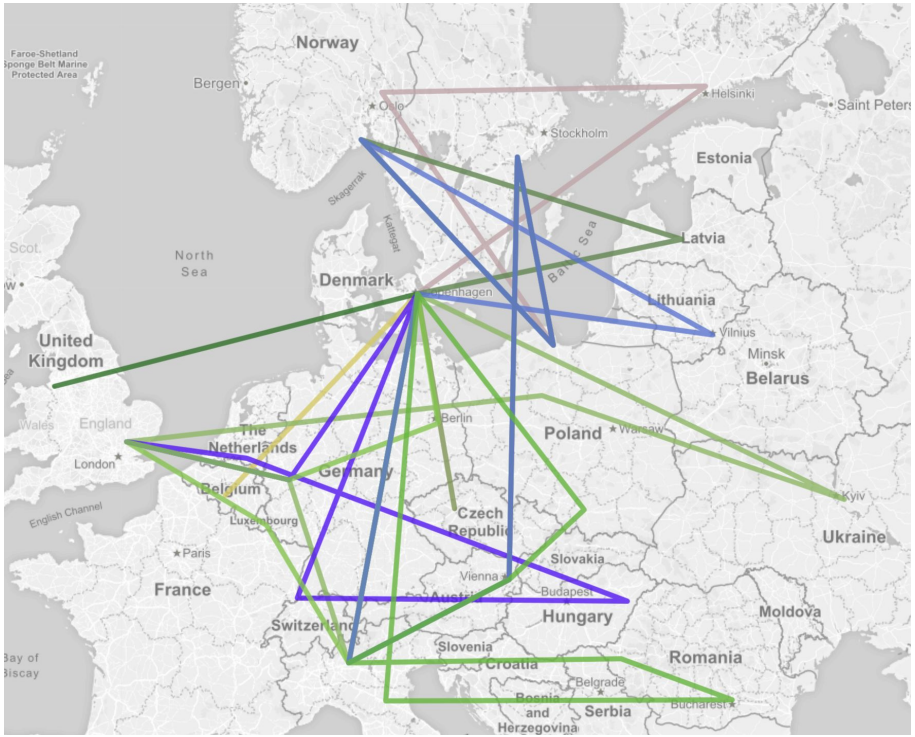
The problem(s)

Root-to-root

Longest distance

Root-to-root

- Uniform cost search variant
- Several parallel searches at once
- Parallelization of search window
- Finds routes of varying length
- Avoids visiting the same airport twice
- Flight cost as cost function



A typical result from a root-to-root search

Algorithm (In very broad terms)

1. Expand root node
2. Start searches for N top nodes by cost
    a. Expand(node ∓ window) → Frontier (Priority Queue) // Expansions done in parallel
    b. Pop from frontier and add to explored until either:
        i. Return **node** is found → Return **node**
        ii. End date is reached → Return **Expand(Previous node, return airport)**
        iii. Timeout is reached → Return **None**
3. Present result for searches that found a solution

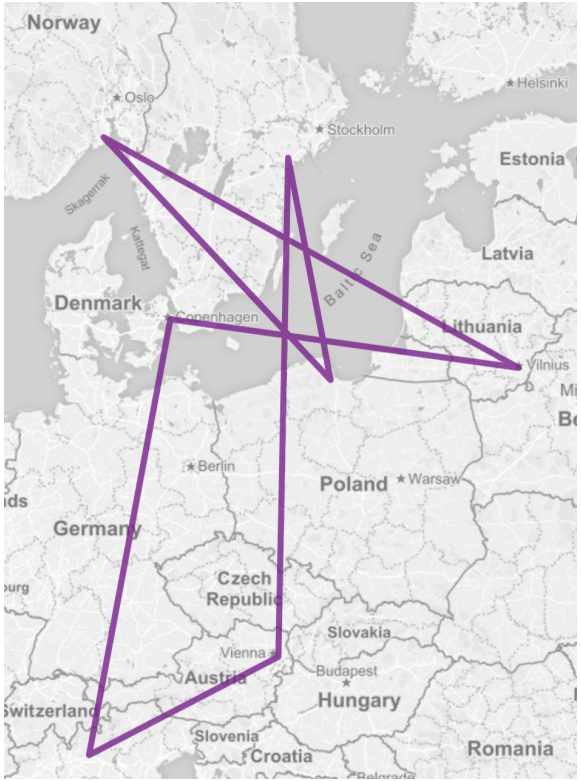
Example result

Input: **CPH**, **07-01** to **08-01**, stay time: **5**, window: **1**.

1.	Copenhagen, Denmark.	
2.	Vilnius, Lithuania.	**448 SEK**
3.	Oslo, Norway.	**110 SEK**
4.	Gdansk, Poland.	**140 SEK**
5.	Stockholm, Sweden.	**134 SEK**
6.	Vienna, Austria.	**161 SEK**
7.	Milan, Italy.	**161 SEK**
8.	Copenhagen, Denmark.	**387 SEK**

Total cost: **1541 SEK**

Dates: **07-01 to 08-01**



Example result

Input: **AMS**, **07-01** to **08-01**, stay time: **5**, window: **1**.

1.	Amsterdam, Netherlands.	
2.	Madrid, Spain.	**727 SEK**
3.	Ibiza, Spain.	**236 SEK**
4.	Valencia, Spain.	**187 SEK**
5.	Bordeaux, France.	**193 SEK**
6.	Naples, Italy.	**215 SEK**
7.	Milan, Italy.	**213 SEK**
8.	Amsterdam, Netherlands.	**644 SEK**

Total cost: **2415 SEK**

Dates: **07-01 to 08-01**

Longest distance

- Same algorithm as root-to-root, but:
  - Has another cost function
  - Does not return to root node
- Focus on maximizing distance over cost

A typical result from a longest distance search

Example result

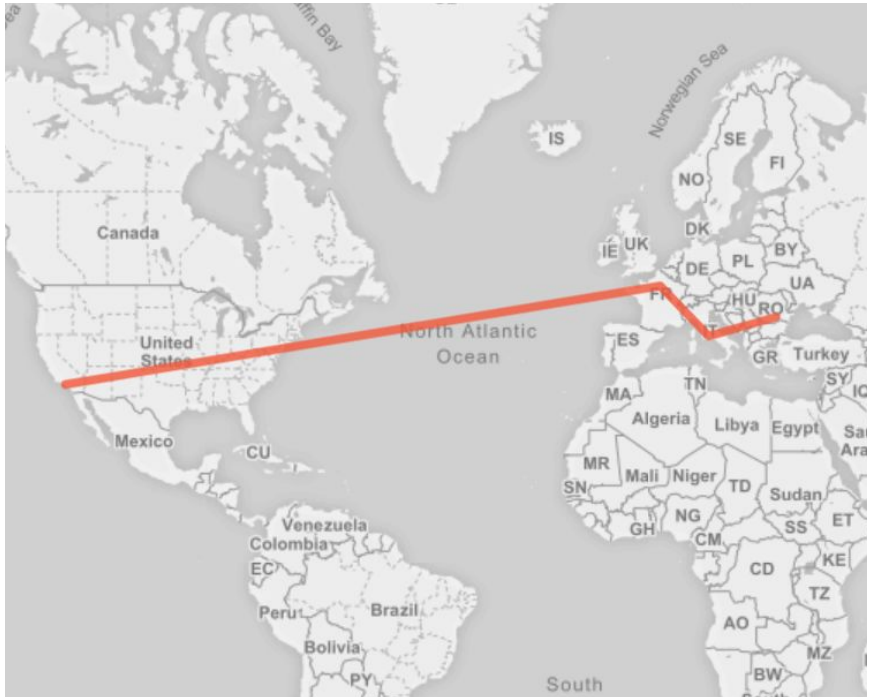
Input: **LAX**, **08-01** to **08-30**, stay time: **10**, window: **5**.

1. Los Angeles, United States.
2. Paris, France. **2032 SEK**
3. Rome, Italy. **415 SEK**
4. Budapest, Hungary. **212 SEK**

Total cost: **2659 SEK**

Distance over cost: **3.98 km/SEK**

Dates: **08-01 to 08-30**



Example result

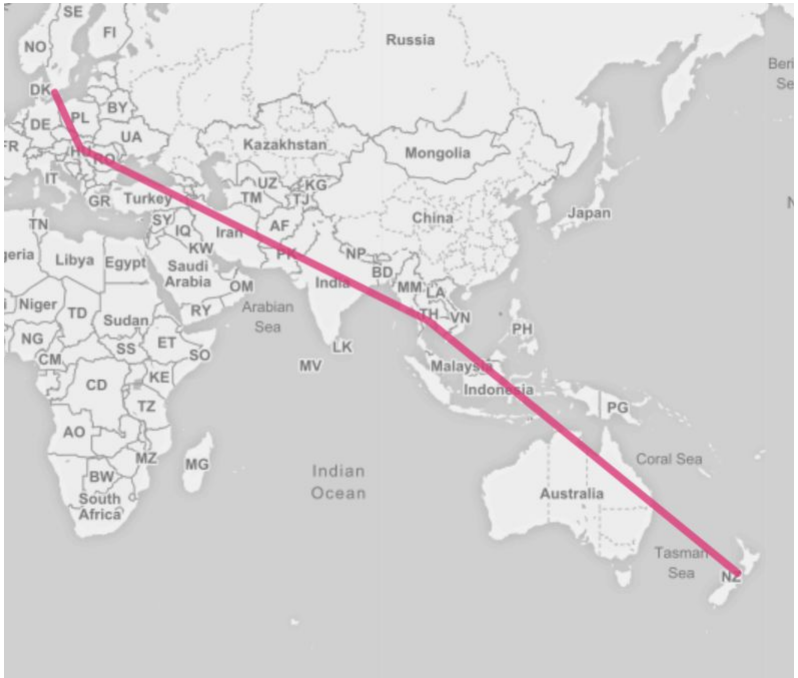
Input: **MMX**, **08-01** to **08-30**, stay time: **7**, window: **5**.

1.	Malmö, Sweden.	
2.	Budapest, Hungary.	**109 SEK**
3.	Bangkok, Thailand.	**2226 SEK**
4.	Nelson, New Zealand.	**3655 SEK**

Total cost: **5990 SEK**

Distance over cost: **2.98 km/SEK**

Dates: **08-01 to 08-30**



Optimization

- API calls are really slow

- Threaded API calls
  - Concurrent.futures library
  - Doing multiple searches concurrently
  - Searches multiple dates at the same time

- Limited number of API calls per minute

- Caching
  - Pickle library
  - Save API response to dictionary on runtime
  - Write dictionary to binary file on finished execution

Lessons learned

- Flights are really cheap (if you know where to look)
- The best search algorithm depends on the problem
- Finding admissible heuristics for real world problems is very hard

Future work

- User determined constraints
  - Price, bagage, number of travellers
  - Choose countries NOT to go to
  - Choose countries to prioritize (fuzzy constraints)
- Use “live” Skyscanner API data
- Improve presentation of routes (Website or similar)
- Add hotel price search for the duration of the stay via hotels API

Questions?