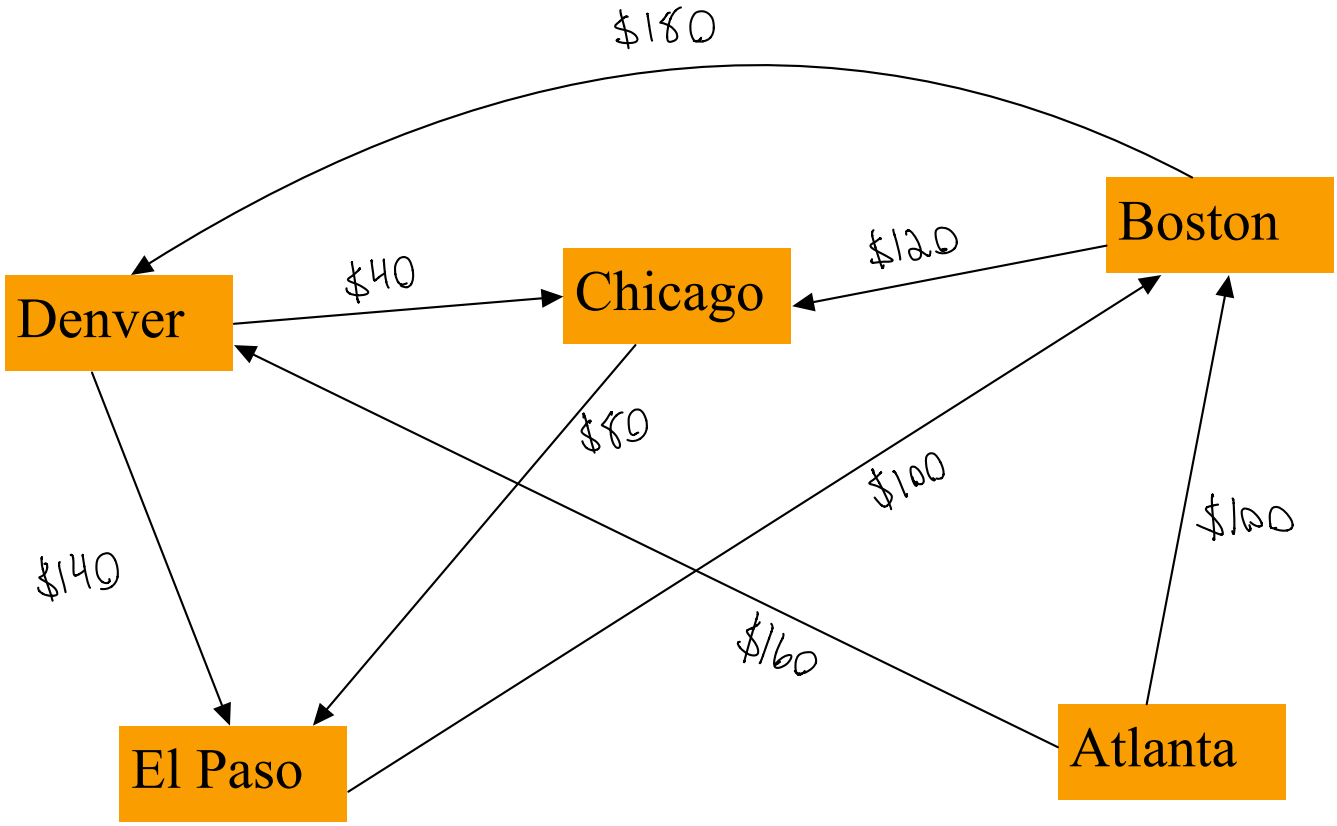
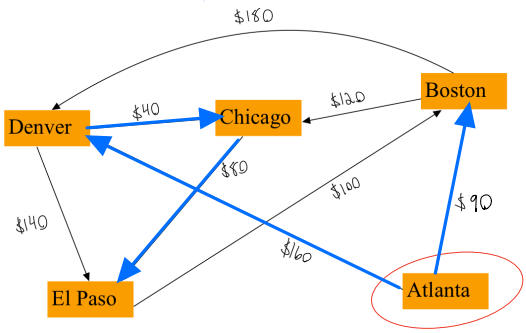


Vertex (city)	Cheapest flight from Atlanta	Previous vertex (city)
Atlanta	$0	
Boston	$100	Atlanta
Chicago	$200	Denver
Denver	$160	Atlanta
El Paso	$260	Chicago

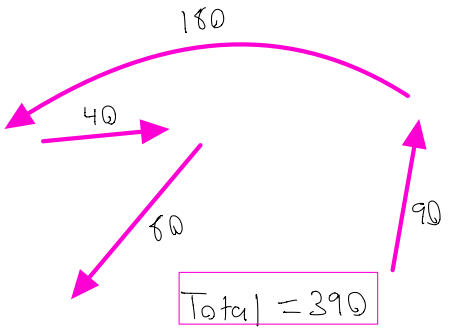
```
Shortest(startCity, destination, routes):
    cheapestTable = { }
    CheapestPrevious = { }
    unvisited = [ ]
    visited = { }
    PriceThru = 0
    CheapestTable[startCity.city] = 0
    Current = start city
    While current
        for each city in current.routes
            if visited[city.name] = False
                add city to unvisited
            PriceThru = cheapestTable[city.name] + city.cost
            if !CheapestTable[city.name] || PriceThru < CheapestTable[city.name]
                CheapestTable[city.name] = PriceThru
                CheapestPrevious[city.name] = current.name
        Current = cheapest adjacent city to current
    ShortestPath = [ ]
    Current = destination
    While current != start
        add current to shortest path
        Current = cheapestPrevious[current]
    add starting city to shortest path
    Print(ShortestPath.reverse)
```

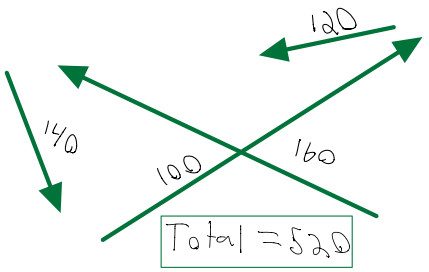
Minimum Spanning Tree

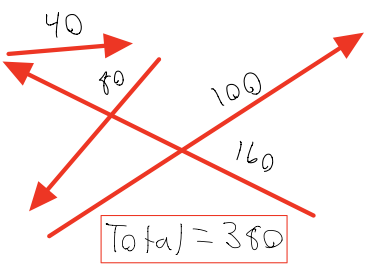


Has to start here

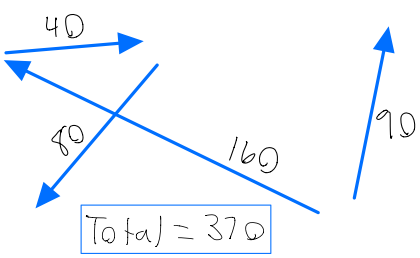
```
visited = {string: Bool}
PriceCompare = 0
cityCompare = " "

unvisited = [vertex1, vertex2, ...]
Min_cost = 0

unvisited[start].erase
while (!unvisited.empty)
    visited[current] = true
    unvisited[current].erase
```








```
for each city
    for(i in range(cities-1))
        find min price
        add previous city to a list
        add price to a list
add minPrice list together
print previous cities list
```



Edge	Weight
Atlanta → Boston	$90
Atlanta → Denver	$160
Denver → Chicago	$40
Chicago → El Paso	$80

```
Struct HashNode:
    String name
    int cost

Struct city:
    String name
    HashNode routes
```

```
HashNode addRoute(city, price):
    routes[city] = price
```

???

city Atlanta

```
Struct vertex:
    String city
    Vector<edge> edgeName[allsize]
```

"Price"

```
Struct edge
    String city
    int cost

vertex.price.Pushback("Boston", 100)
```

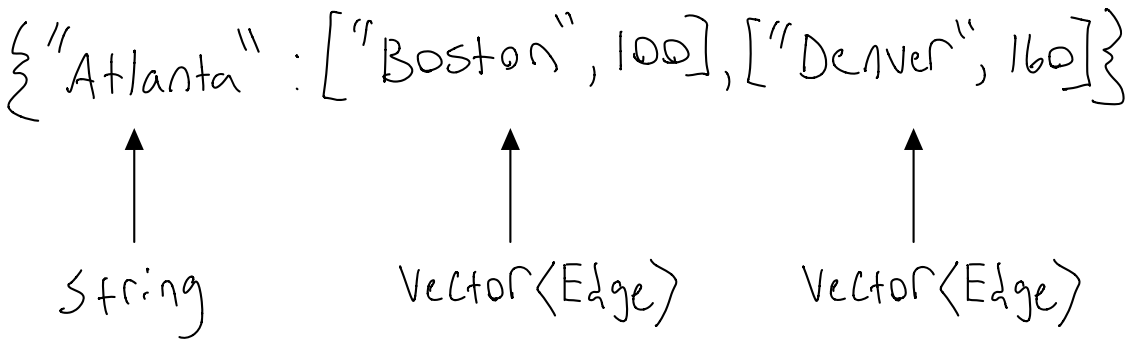
```
Struct Edge
    String city
    int cost

Vector<vector<Edge>> graph

for i in graph[u].size()
    Edge edge = graph[u][i]
```

access vertexes from vertex u

```
Edge new
new.city = "Boston"
new.cost = 100
graph["Atlanta"].Pushback(new)
```

adds a $100 flight from Boston to Atlanta

is this both ways? Maybe Put city & destination in edge struct?

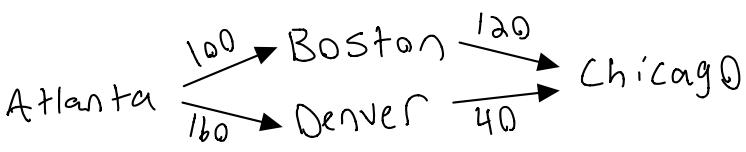


add_vertex(name):

```
Map<string, vector<Edge>> routes;
return routes[name]
```

routes["Atlanta"].push_back({"Boston", 100})

add_edge(start, end



start	current	end	routes
Atlanta	Boston Denver	Chicago	routes

Minimum(start, current, end, routes)

```
for routes[current]
  if dest == start
    continue
  if not visited
    Mark visited
  if cheapest[dest] == 0 || cost < cheapest[dest]
    cheapest[dest] = cost
```

once per city

Minimum(start, dest, end, routes)

```
Minimum(start, current, end, routes)

    for routes[current]
        if dest == start
            continue
        if not visited
            Mark visited
        if cheapest[dest] == 0 || cost < cheapest[dest]
            cheapest[dest] = cost
```

```
current = start
if current not visited
    Mark visited
```

```
unvisited[city]
                                  Min_price
                                  compare
Boston.cost
if denver.cost < compare
    compare = denver.cost
    Min_price = denver.name
```