~~Application designed to Find the Cheapest way to get from A to Z or A back around to A~~

Application designed to find the most cost effective way for a user to get from A to Z or A back around to A. The user inputs a list of cities/Airports they wish to go to and the program finds the best route for the user without repetition.

Scanned by CamScanner

array list of user entered locations

{A, B, C, D, E}

- recursive algorithm

start location

hash map of best route

{ }

Pre - conditions

A { →B = 10, →C = 15, →D = 5, →E = 20 }

B { →A = 15, →C = 5, →D = 10, →E = 10 }

C { D = 5, B = 15, A = 20, E = 25 }

D { A = 10, B = 20, C = 15, E = 5 }

E { A = 5, B = 25, C = 10, D = 20 }

Scanned by CamScanner

array list of user entered locations

{A, B, C, D, E}

^ start location

- recursive algorithm

dictionary/ hash map of best route

{ }

Pre-conditions stored in DB (for Sample)

A {→B = 10, →C = 15, →D = 5, →E = 20}

B {→A = 15, →C = 5, →D = 10, →E = 10}

C {D = 5, B = 15, A = 20, E = 25}

D {A = 10, B = 20, C = 15, E = 5}

E {A = 5, B = 25, C = 10, D = 20}

user input array → userIn { }

temp hashmap array → temp { }

final hash map array → final { }

best_route (current, user_in array) {

cost = get Cost (location);

{dest_cost += getCost (dest_location)

temp.append (dest_location, dest cost)}

for (i → end of array) {

if (current array loc == current) {

skip do nothing

} else {

user input

}

}

Scanned by CamScanner

```
user_in  array
temp  hashmap
final  hashmap
bestroute (index, user_in array) {
    for (i → end of array)
        destcost = getCost (array[i])
        temp.append (array[i], destCost)
    sorting algorithm to find cheapest destination
        final . append (cheapest)
        user_in . remove (cheapest)
        clear temp array
    if (user_in.is empty ())
            return final
        else
            best route (cheapest, user_in array)
}
```

Scanned by CamScanner