Mark Demore II, 2d Lt

CSCE686 - Dr. Lamont

Spr 2020 - Homework 2

Greedy Algorithm for the Minimum Cabs Problem

<https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/practice-problems/algorithm/minimum-cabs-0798cfa5/>

1. Introduction
2. Problem Domain Requirements
3. Size of Solution Space: n for queue
4. Objective Function: min ∑ni=1 ci where ci = 1 if someone is waiting for a cab and cj for all j < i are in use
5. Constraint: Define Q as a queue of cab riders with start and end time of trips

min ∑ni=1 ci

End of Search, Q is empty

1. Domains: Q is queue of riders, C is set of cabs needed
2. Search Algorithm Domain Requirements
3. Name: Greedy-Search (Di)
4. Domains: Data input – Di set of candidates (riders), Data output – Do set of solutions (cab/rider pairings)
5. Operations: Assign(x) – x is in Di, pop from queue, assign to cab ci and add (x,ci) to solution set C
6. Search Algorithm Design Specification
7. Name: Min Cab Greedy Search (Di)
8. Domains: Di – riders and travel time pairings, Do – riders and cab pairings
9. Operations: findCab(time) – search for available cab, if not available, add cab

assign(x) – x is in Di, findCab and add cab/rider pairing to C, pop(x)

pop(x) – remove rider from queue

1. Expanded Algorithmic Design Specification

Set C = empty

Queue Q = empty

Set c = empty

findCab(time):

for cab in c:

if cab is available at time:

return cab

c.add(cab)

return cab

pop(rider):

Q.dequeue(rider,time)

assign(rider,time):

cab = findCab(time)

C.append(rider,cab)

pop(rider,time)

MinCabGreedySearch(Di):

for pair in Di:

Q.append(rider,time)

while Q != empty:

assign(Q.top)

return C

1. Conclusion
2. References

Lec2 – pg 16+29