Mo Felonda

HW5 CS577

(1)

Algorithm Design

Input: A list L of the lengths l and speed limits v (i.e. $L = [(l_1, v_1) \dots (l_n, v_n)]$), the number of times you can break the speed limit k, and the amount of speed you break it by v. (k and v are positive integers)

Output: A rational number minTime that is the minimum time to go from your house to work.

```
procedure SHORTEST-COMMUTE(L,k,v)
for i=1...n do
    A[i] ← L[i].l ÷ L[i].v
    B[i] ← L[i].l ÷ (L[i].v + v)
for i=1...n do
    C[i] ← (A[i] - B[i], A[i], B[i])
[C is a list of times i.e. C_i = (timeSaved_i, origTime_i, speedBreakTime_i)]
Sort(C by largest timeSaved (largest difference between A[i] and B[i]))
for i=1...n do
    if i ≤ k then
        minTime += C[i].speedBreakTime
    else
        minTime += C[i].origTime
return minTime
```

Proof of optimality (Format and some wording derived from HW5 discussion problem #1)

For the purposes of our proof, let us define an inversion as any pair i and j where $\frac{l_i}{v_i} - \frac{l_i}{v_i+v} > \frac{l_j}{v_j} - \frac{l_j}{v_j+v}$ and i comes after j in the list. By definition, any ordering that differs from the ordering in the greedy solution has at least one inversion. Further, any ordering with at least one inversion has a pair of consecutive elements i and j that constitute an inversion. Let us consider the case in which $\frac{l_i}{v_i} - \frac{l_i}{v_i+v} < \frac{l_j}{v_j} - \frac{l_j}{v_j+v}$ and i and j are consecutive elements, meaning $(\dots i, j \dots)$ no other elements in the list will be affected by swapping i and j. Let C_{i-1} represent all route parts before L_i, and let $i = k$. In other words, C_{i-1} represents all route parts in which the speed limit has been broken, other than the i_{th} element. The final necessary swap can be represented by $C_{i-1} + \frac{l_i}{v_i+v} + \frac{l_j}{v_j}$ before the swap, and $C_{i-1} + \frac{l_j}{v_j+v} + \frac{l_i}{v_i}$ after the swap. Mathematically we can conclude that the total commute time does not increase if and only if $\frac{l_i}{v_i} - \frac{l_i}{v_i+v} \leq \frac{l_j}{v_j} - \frac{l_j}{v_j+v}$, which is the criterion of this case. By this method we can swap all ensuing inversions, leading us to our greedy solution with no increase in cost. Thus, the greedy solution returns the minimum commute time.

HW5 Continued

Time Complexity: We know that sorting takes $O(n\log n)$ time. We also know that all calculations performed take $O(n)$ time. This includes calculating timeSaved and mintime. Taking all operations in this algorithm into consideration we can conclude that the total running time = $O(n\log n) + O(n)$ which is ultimately $O(n\log n)$

Counterex: (a) $L = [(100mi, 70mph), (90mi, 20mph)]$, $k=1$, $v=10$. While the greedy algorithm presented would suggest breaking the speed limit along the 100mi part, this would result in a travel time of 5.75 hours. Breaking the speed limit along the 90mi part results in a travel time of 4.43 hours, proving this greedy algorithm is incorrect.

(b) $L = [(100mi, 30mph), (20mi, 20mph)]$, $k=1$, $v=10$. The greedy algorithm present may suggest breaking the speed limit along the 20mi part, but this in incorrect. That would result in a travel time of about 4 hours, while choosing the 100mi part would only take 3.5 hours. Thus we can conclude that this greedy algorithm is incorrect.