EECS 336 Problem 3.2

Algorithm 1 Shortest travel time

Require: Directed Graph $G = (V, E)$, travel time functions $f_e(t), e \in E$, starting vertex s.

function ShortestTravelTime($G, s, f_e(t)$)
  Let S be the set of explored nodes.
  for each $u \in S$, we store the earliest arrival time T(u)
  Initialize $S = \{\}$ and Insert$(s, 0)$
  Insert(v, ∞) for $v \in V - s$, denote T'(v) be the key value
  while queue is not empty **do**
    $(v, d) = deletemin()$
    for each neighbor u of v **do**
      if $f_{u-v}(T(v)) < T'(u)$ (the key value of u) **then**
        decreasekey$(u, f_{u-v}(T(v))$
      end if
    end for
  end while
end function

RunTime

Step 1. The initialization of the priority queue takes $O(nlogn)$.

Step 2. The while loop is executed for n times. In each iteration, only the edges of the last vertex are searched. So each directed edge could be searched only once (totally m). Each of the searching could lead to an decreasekey operation, which takes log(n). So step2 takes O(mlog(n))

Therefore, T(n) = O(mlogn)

Correctness/Optimal

Claim: Consider set S_k is the set of explored vertices after the kth iteration of the algorithm's while loop. We prove that, for each $u \in S_k$, $T(u)$ is the shortest travel time from s.

Proof by induction:

Base case: When $k = 0$, $S_0 = \{s\}, T(s) = 0$ is correct.

Induction Hypothesis: For each $u \in S_k$, $T(u)$ is the shortest travel time from s.

Induction Step: When one more vertex v is explored (e.g. $S_k \to S_{k+1}$), the claim still holds.

Let P_v be the $s - v$ path found by the algorithm, and u is the last vertex before v on P_v. By the induction hypothesis, P_u has the shortest travel time starting from s to u. Now consider any other $s - v$ path P', we want to show that the travel time though P' is at least the same as that through P_v. Let x be the last vertex in S_k on P', and y be the first vertex in $V - S_k$ on P'.

Since $f_e(t) \geq t$, $T_{P'}(v) \geq T_{P'}(y)$ (Eq 1), which indicates the travel time through path P' from s to v should be at least that from s to y. Similarly,

$T_{P'}(y) = f_{x-y}(T_{P'}(x)) \geq T_{P'}(x)$. As $T(x)$ is the shortest travel time from s to x, $T_{P'}(x) \geq T(x)$.

Since $T_{P'}(x) \geq T(x)$, according to property 2, $T_{P'}(y) = f_{x-y}(T_{P'}(x)) \geq f_{x-y}(T(x))$. (Eq. 2) By combining (Eq 1 & 2), we can obtain $T_{P'}(v) \geq f_{x-y}(T(x))$. (Eq. 3)

Let $T'(m)$ denote the updated key of m in the priority queue. Since the key should be the shortest travel time, $T'(m) \leq f_{n-m}(T(n))$ for any n in the explored set S. Therefore, Eq 3 could be modified as $T_{P'}(v) \geq f_{x-y}(T(x)) \geq T'(y)$.

Since the algorithm select vertex v rather than y, $T'(v) \leq T'(y)$.

Therefore $T_{P'}(v) \geq T'(v)$, which indicates that the algorithm provides the path with shortest travel time.

Proved.