Dijkstra's Algorithm

Single source shortest path
No negative weight edges

idea
- Start with s at distance 0
  - S is set of nodes whose distances are known repeatedly
  - find vertex u ∈ V - S whose shortest path estimate is minimum
  - relax all edges leaving u

Dijkstra (G, w, s)

Initialize Single Source (G, s)
for each v ∈ V
  v.dist = ∞, v.prev = nil
s.dist = 0
Q = V priority queue
while Q not empty do
  u = Q. extract min
  S = S ∪ {u}
  for each vertex v ∈ adj[u]
    relax (u, v)
relax (u, v)
if v.dist > u.dist + w(u, v)
  Then v.dist = u.dist + w(u, v)
  v.prev = u

Time:
- if Q is binary heap (Adjacency List) V extracting E heap燃烧 key
  O((E + V) log V)
- if Q is Fibonacci heap
  O(E log V + E)

O(E log V)
Proof of Correctness

Let $d(s,v)$ be the (actual) length of the shortest path from $s$ to $v$.

At the start of each iteration of the while loop:

$v \cdot \text{dist} = d(s,v)$ for all $v \in S$

Alternate:
- For all $v \in S$, $v \cdot \text{dist} = d(s,v)$
- For all $v \notin S$, $v \cdot \text{dist}$ is the length of the shortest path whose intermediate vertices are in $S$

Also note:

Fact: If $u$ is an intermediate vertex on the shortest path from $s$ to $v$, then that part of the path from $s$ to $u$ is the shortest path to $u$. 
Let \( u \) be chosen as the node with minimum \( u \cdot \text{dist} \).

Assume that it is not the length of the shortest path, \( d(s,u) < u \cdot \text{dist} \).

By part II, it is the shortest path going through \( S \). Thus the actual shortest path must include some nodes not in \( S \).

Let \( x \) be last node in \( S \), \( y \) first node outside. (Note: path could go back into \( S \)).

By fact, \( y \cdot \text{dist} = d(s,y) \).

So \( y \cdot \text{dist} = d(s,y) < d(s,u) < u \cdot \text{dist} \), \( \uparrow \leq \uparrow \) (no weight edges).

But then \( u \) would not have been chosen. Contradiction.

Proof of part II: correctness of relax.