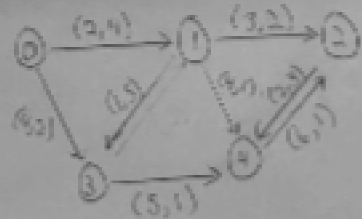
Tomasz Hulka and Frank Errichiello
CS 401: Algorithms
Program 2: Cost Constrained Shortest Paths

PART I. (Pencil and Paper)

The Priority Queue and Drawing with Heap Implementation

(1)



Priority (<v>, <cost>, <time>)
Order by cost, if cost is the same, use time as a tie breaker.

Priority Queue

(0,0,0)
(1,2,4)
(3,3,9)
(3,4,2)
(2,5,6)
(4,6,5)

Some values are not added due to lack of efficiency and not making the path better in any way

(4,7,9) Not added to list
(4,8,10) Not added to list
(4,9,3)
(2,12,6) Not added to list
(2,13,10) Not added to list
(2,14,11) Not added to list
(2,15,4)
(4,17,7) Not added to list

(inc cost; dec time)

v	(cost, time) list Order list
0	(0,0)
1	(2,4)
2	(5,6) (15,4)
3	(3,9) (4,2)
4	(6,5) (9,3)

(2)

For any $v \in V$, $|S[v]| = O(\frac{M}{V})$.

In the worst case, v has outgoing edges to all other vertices, and every path examined is non-dominated.

For our heap, we have an equal amount of insertions and deletions (Implemented with a min-heap). In the worst case, we have $|E|$ heap operations if we examine only optimal paths (dominated).
So, the amount of heap operations is

$$\sum_{v \in V} |S[v]| = O(V \cdot \frac{M}{V}) = O(M)$$

With $O(M)$ heap operations and a heap-based implementation having $O(1)$ deletes, $O(\log V)$ insertions

$T(N) = O(M \log V)$

PART II. (Implementation)

Our implementation was done in c++ using a heap-based approach and an adjacency list representation of a graph.

For our Heap we use a standard library priority queue, the queue was restrained to increasing cost and decreasing time, if an entry to the same vertex as one already in the queue was being added, the check would see if the cost got more expensive and the time got lessened, if it didn't it was thrown away and the path is never considered. (I.E) in the image above all the ones in the heap not added to the list, would also not be added to the heap.

For S the insertion works the same, the check would see if the cost got more expensive and the time got lessened, if it didn't it was thrown away and the pair is never considered for the best paths. (I.E) in the image above all the ones in the heap not added to the list. To get the paths, we knew that if a path was non-dominated, and was about to be added to the queue, we pushed the path signature and the path. It is always the case that the path to the vertex being observed (u) + edge (u,v) was the correct non-dominated path to v.