Implementing Dijkstra with Heaps

```
import heapq

def dijkstra(HG, v):
    dist_so_far = {v: 0}
    final_dist = {}
    heap = [(0, v)]
    while dist_so_far:
        (w, k) = heapq.heappop(heap)
        if k in final_dist or (k in dist_so_far and w > dist_so_far[k]):
            continue
        else:
            del dist_so_far[k]
            final_dist[k] = w
        for neighbor in [nb for nb in HG[k] if nb not in final_dist]:
            nw = final_dist[k]+ HG[k][neighbor]
            if neighbor not in dist_so_far or nw < dist_so_far[neighbor]:
                dist_so_far[neighbor] = nw
                heapq.heappush(heap, (nw, neighbor))
    return final_dist

```

Weighted Marvel Graph

Least Obscure Path

```
121     return final_dist
122
123 def solve(graph, actor_0, actor_1):
124
125     return amended_dijkstra(graph, actor_0)[actor_1]
126
127 if __name__ == '__main__':
128     total_graph = read_graph("imdb-1.tsv")
129     movie_obscurity = read_obscurity("imdb-weights.tsv")
130
131     HG = make_hop_graph(total_graph, movie_obscurity)
132
133     print "answer = {"
134     for ch1, ch2 in answer:
135         #ch1, ch2 = t[0], t[1]
136         #routes = amended_dijkstra(HG, ch1)
137         answer[ch1, ch2] = solve(HG, ch1, ch2)
138
139         print '\t(\"' + ch1 + '\", \"'+ ch2 +'\"):', answer[ch1, ch2], ","
140     print "}"
```