Mihoc Adelin
group 915/1

Documentation

Bellman-Ford Algorithm

The algorithm keeps two mappings:

- *dist[x]* = the cost of the minimum cost walk from *s* to *x* known so far
- *prev[x]* = the vertex just before *x* on the walk above.

Initially, *dist[s]*=0 and *dist[x]*=∞ for *x* ≠ *s*; this reflects the fact that we only know a zero-length walk from *s* to itself.

Then, we repeatedly performs a *relaxation* operation defined as follows: if *(x,y)* is an edge such that *dist[y]* > *dist[x]* + *c(x,y)*, then we set:

- dist[y] = dist[x] + c(x,y)
- prev[y] = x

Algorithm:

Initialization

```
# step 1
distance = [inf for _ in range(self._no_of_vertices)]
distance[source] = 0
predecessor = [False for _ in range(self._no_of_vertices)]
```

Relaxation

```
# step 2
for _ in range(self._no_of_vertices - 1):
    flag = False
    for item in self._cost_dict.items():
        if distance[item[0][1]] > distance[item[0][0]] + item[1]:
            distance[item[0][1]] = distance[item[0][0]] + item[1]
            predecessor[item[0][1]] = item[0][0]
            flag = True
    # if there are no modifications in the distance then the minimum distance was found
    if not flag:
        break
```

Search for negative cycles

```
# step 3
for item in self._cost_dict.items():
    if distance[item[0][1]] > distance[item[0][0]] + item[1]:
        raise ValueError("Graph contains a negative cost cycle!\n")
```

Path construction

```
# construct the path
print("A lowest cost walk from " + str(source) + " to " + str(target) + " is " +
str(distance[target]))
path = ""
current_vertex = target
while current_vertex != source:
    path += str(current_vertex) + " <- "
    current_vertex = predecessor[current_vertex]
path += str(source)
print(path)
```