Algorithmics
Correction Final Exam #4 (P4)

Undergraduate 2^{nd} year (S4) - API – Epita

16 May 2017 - 10*h*

Solution 1 **(MST et SP ... –** ***3 points*****)**

1. The Bellman algorithm is usable in cases where the costs of the arcs are of any kind, but where the graph does not have a circuit.
2. The algorithm determining the mst of an undirected graph whose principle is close to that of Dijkstra is PRIM.
3. The MST of the graph is that of figure 1.

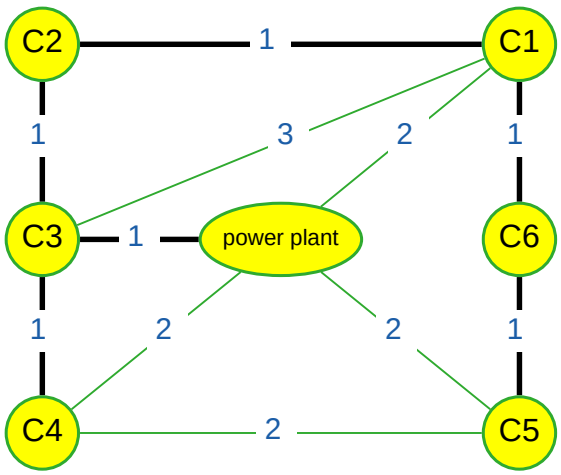


Figure 1: MST of the graph.

4. The shortest path tree from "power plant" vertex of the graph is that of figure 2.

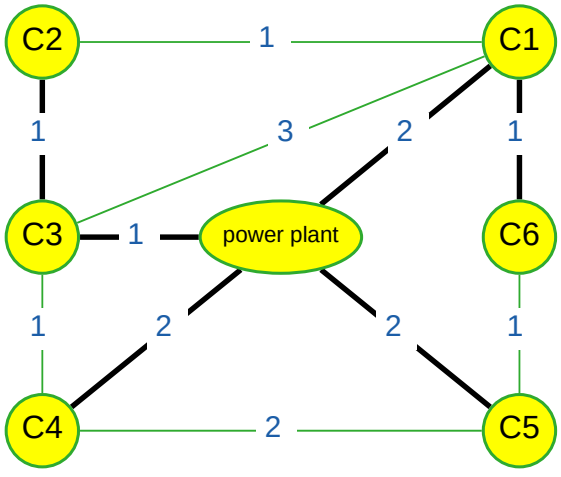


Figure 2: Shortest path tree from "power plant" vertex of the graph.

***Solution 2* (Condensation)**

Specifications:

The function `condensation(`G`,` *scc*`)` builds the condensation G_R of a digraph G, with *scc* its component list. The function returns G_r and the vector of components: a vector that give for each vertex, the number of its component (the vertex in G_R).

```
def condense(G, scc):

    comp = [-1] * G.order
    k = len(scc)
    for i in range(k):
        L = scc[i]                          for s in scc[i]:
        for j in range(len(L)):                 comp[s] = i
            comp[L[j]] = i

    Gr = graph.Graph(k, directed = True)
    for s in range(G.order):
        for adj in G.adjLists[s]:
            (x, y) = (comp[s], comp[adj])
            if x != y: # (and y not in Gr.adjLists[x])
                Gr.addEdge(x, y)      # Gr.adjLists[x].append(y)

    return (Gr, comp)
```

***Solution 3* (Graphes and Mystery – *3 points*)**

1.

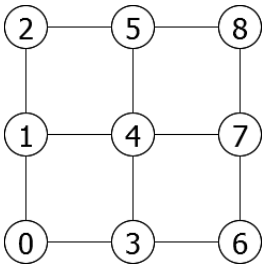
	Call number	*Returned result*
(a) `test(`G_2`)`	5	False
(b) `test(`G_3`)`	7	True

2. *What is the information returned by* `test(G)`*?*

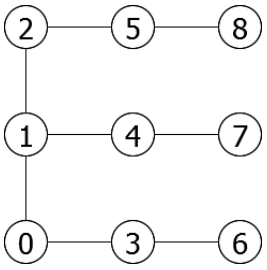
`test(`G`)` tests if G is strongly connected.

***Solution 4* (T-spanner – *10 points*)**

1.
(a) t-spanners for a stretch factor of 2



(b) t-spanners for a stretch factor of 5



2. (a) **Specifications:**
The function `Dijkstra(G, src, dst)` returns the length of the shortest path between *src* and *dst* in G, $+\infty$ if there is no path.

```
def Dijkstra(G, src, dst): # 4.5 pts

    dist = [inf] * G.order
    dist[src] = 0
    H = Heap(G.order)
    update(H, src, 0)

    while not H.isEmpty():
        (_, cur) = pop(H)
        if cur == dst:
            return dist[dst]
        for s in G.adjLists[cur]:
            if dist[s] > dist[cur] + G.costs[(cur, s)]:
                dist[s] = dist[cur] + G.costs[(cur, s)]
                update(H, s, dist[s])

    return dist[dst]    #inf
```

(b) **Specifications:**
The function pathGreedy(n, L, t) returns a t-spanner (with stretch factor $= t$) for the set of n points (number form 0 to $n-1$) with L the list of triplets (p, q, $|pq|$).

```
def pathGreedy(order, edges, stretch):  # 4.5 pts

    edgeHeap = []
    for (x, y, w) in edges:
        heappush(edgeHeap, (w, x, y))

    G = graph.Graph(order, False, costs = True)
    while edgeHeap != []:
        (w, x, y) = heapq.heappop(edgeHeap)
        if Dijkstra(G, x, y) > stretch * w:
            G.addEdge(x, y, w)

    return G
```

bonus When the stretch factor is $n-1$ with n the number of points, what is the t-spanner?

The t-spanner is an MST.