A Algorismes

A.1 DFS

Codi

```
parent={}
topo=[]
def DFS(Adj):
    node=[]
    for i in range(0, len(Adj)):
        node.append(i)

    for s in node:
        if s not in parent:
            print "From node %d:" %s
            print s
            parent[s]=None
            DFS_recursive(Adj, s)
    print "Recursion order (topological sort for directed acyclic graphs):"
    topo.reverse()
    print topo


def DFS_recursive(Adj, s):
    for v in Adj[s]:
        if v not in parent:
            print v
            parent[v]=s
            DFS_recursive(Adj, v)
    topo.append(s)
```

Exemple d'entrada

Exemple de sortida

A.2 BFS

Codi

```
def BFS(Adj, s):
    level={s:0}
    parent={s:None}
    i=1
    frontier=[s]
    print s
    while frontier:
        next=[]
        for u in frontier:
            for v in Adj[u]:
                if v not in  level:
                    level[v]=i
                    parent[v]=u
                    next.append(v)
                    print v
        frontier=next
        i+=1
    print level
```

Exemple d'entrada

Exemple de sortida

A.3 Dijkstra

Codi

```
def Dijkstra(Adj, s):
    Q={}
    dist={}
    tree={}
```

```
    for i in range(0, len(Adj)):
        Q[i]=float("inf")
        dist[i]=float("inf")
    Q[s]=0
    while Q:
        u = min(Q, key=Q.get)
        dist[u] = Q[u]
        for v in Adj[u]:
            if v in Q:
                if Q[v] > Q[u] + Adj[u][v]:
                    Q[v] = Q[u] + Adj[u][v]
                    tree[v] = u
        Q.pop(u)

    return dist, tree


def OrderedDijkstra(Adj, s):
    Q = dict.fromkeys(Adj.keys(), float("inf"))
    dist = dict.fromkeys(Adj.keys(), float("inf"))
    tree = {}
    Q[s] = 0
    while Q:
        u = min(Q, key=Q.get)
        dist[u] = Q[u]
        for v in Adj[u]:
            if v in Q:
                if Q[v] > Q[u] + Adj[u][v]:
                    Q[v] = Q[u] + Adj[u][v]
                    tree[v] = u
        Q.pop(u)

    return dist, tree
```

Exemple d'entrada

Exemple de sortida

A.4 Bellman-Ford

Codi

```
def BellmanFord(Adj, s):
    dist={}
    tree={}
    for i in range(0, len(Adj)):
        dist[i]=float("inf")
        tree[i]=None
    dist[s]=0

    for i in range(0, len(Adj)-1):
        for u in range(0, len(Adj)):
            for v in Adj[u]:
                if dist[v] > dist[u] + Adj[u][v]:
                    dist[v] = dist[u] + Adj[u][v]
                    tree[v]=u
    for u in range(0, len(Adj)):
        for v in Adj[u]:
            if dist[v] > dist[u] + Adj[u][v]:
                print "There are negative-weight cycles"
                break
    return dist, tree
```

Exemple d'entrada

Exemple de sortida

A.5 Prim

Codi

```
def Prim(Adj):
    Q={}
```

```
    tree={}
    for i in range(0,len(Adj)):
        Q[i]=float("inf")
    Q[0]=0
    while Q:
        u = min(Q, key=Q.get)
        for v in Adj[u]:
            if v in Q and Adj[u][v] < Q[v]:
                Q[v] = Adj[u][v]
                tree[v] = u
        Q.pop(u)
    return tree
```

Exemple d'entrada

Exemple de sortida

A.6 Kruskal

Codi

```
def Kruskal(Adj):
    subtree = UnionFind()
    tree = []
    for e, u, v in sorted((Adj[u][v],u,v) for u in Adj for v in Adj[u]):
        for u in Adj:
            for v in Adj[u]:
                if subtree[u] != subtree[v]:
                    tree.append((u,v))
                    subtree.union(u,v)
    return tree
```

Exemple d'entrada

Exemple de sortida

A.7 Floyd-Warshall

```
def FloydWarshall(Adj):
    dist=[[float("inf") for x in range(len(Adj))] for y in range(len(Adj))]
    for i in range(0,len(Adj)):
       dist[i][i] = 0
    for v in range(len(Adj)):
        for u in Adj[v]:
            dist[v][u] = Adj[v][u]
    for x in range(len(Adj)):
        for u in range(len(Adj)):
            for v in range(len(Adj)):
                if dist[u][v] > dist[u][x] + dist[x][v]:
                    dist[u][v] = dist[u][x] + dist[x][v]
    return dist
```

Exemple d'entrada

Exemple de sortida

A.8 Hamilton

Codi

```
def Hamilton_recursive(Adj, s, e, path):
    path = path + [s]
    if s == e:
        return path
    for n in Adj[s]:
        if n not in path:
            nou_path = Hamilton_recursive(Adj, n, e, path)
            if nou_path:
                return nou_path
    return None
```

```
def Hamilton(Adj, s, e):
    path=[]
    return Hamilton_recursive(Adj, s, e, path)
```

Exemple d'entrada

Exemple de sortida

A.9 Euler

Codi

```
def Euler(Adj):
    graf = Adj
    senar = [v for v in graf.keys() if len(graf[v])%2 != 0]
    senar.append(graf.keys()[0])
    print senar

    if len(senar)>3:
        return None

    Q = [senar[0]]
    path = []
    while Q:
        v = Q[-1]
        if graf[v]:
            u = graf[v][0]
            Q.append(u)
            del graf[u][graf[u].index(v)]
            del graf[v][0]
        else:
            path.append(Q.pop())

    return path
```

Exemple d'entrada

Exemple de sortida

A.10 Coloració

Codi

```
def coloring(Adj):
    graph = sorted(Adj, key=lambda k:len(Adj[k]), reverse=True)
    colors = {}
    usat = False
    actual = 0

    for i in range(0, len(Adj)):
        colors[i]=None
    colors[graph[0]]=0

    while None in colors.values():
        for v in graph:
            if colors[v] == None:
                for k in Adj[v]:
                    if colors[k] == actual:
                        usat = True
                        break

                if usat == False:
                    colors[v] = actual
                usat = False
        actual = actual + 1
    return colors
```

Exemple d'entrada

Exemple de sortida

A.11 Metro

Codi

```
def metro(Adj, inici, final):
    recorregut=[]

    print "Punt inicial:", inici.decode("ISO-8859-15")

    print "Punt final:", final.decode("ISO-8859-15")

    dist, tree = OrderedDijkstra(Adj, inici)
    print type(inici)
    print type(final)

    i = final
    while tree[i] != inici:
        recorregut.append(tree[i])
        i = tree[i]

    recorregut.append(inici)
    recorregut.reverse()

    total= dist[final]+(25*(len(recorregut)-2))

    minuts = total/60
    segons = (total%60)*0.60
    print "Temps net del recorregut:", dist[final]
    print "Temps total del recorregut:", int(minuts),"minuts i", int(segons), "seg
    print "Recorregut:",
    print "[",
    for i in range(0,len(recorregut)):
        print recorregut[i].decode("ISO-8859-15")+",",

```

```
32      print final.decode("ISO-8859-15"),"]"
```

Exemple d'entrada

```
graf_metro={"1_Hospital de Bellvitge":{"1_Bellvitge":90},
↪ "1_Bellvitge":{"1_Hospital de Bellvitge":90, "1_Av.
↪ Carrilet":100},"1_Av. Carrilet":{"1_Bellvitge":100, "1_Rbla.
↪ Just Oliveras":65, "8_L'Hospitalet - Av. Carrilet":180},
↪ "1_Rbla. Just Oliveras":{"1_Av. Carrilet":65, "1_Can
↪ Serra":60}, "1_Can Serra":{"1_Rbla. Just Oliveras":60,
↪ "1_Florida":60}, "1_Florida":{"1_Can Serra":60,
↪ "1_Torrassa":60}, "1_Torrassa":{"1_Florida":60, "1_Santa
↪ Eulàlia":85, "9S_Torrassa":240}, "1_Santa
↪ Eulàlia":{"1_Torrassa":85,"1_Mercat Nou":75}, "1_Mercat
↪ Nou":{"1_Santa Eulàlia":75, "1_Plaça de Sants":60}, "1_Plaça de
↪ Sants":{"1_Mercat Nou":60, "1_Hostafrancs":50, "5_Plaça de
↪ Sants":282}, "1_Hostafrancs":{"1_Plaça de Sants:":50,
↪ "1_Espanya":55}, "1_Espanya":{"1_Hostafrancs":55,
↪ "1_Rocafort":60, "3_Espanya":209, "8_Espanya":120},
↪ "1_Rocafort":{"1_Espanya":60, "1_Urgell":55},
↪ "1_Urgell":{"1_Rocafort":55, "1_Universitat":58},
↪ "1_Universitat":{"1_Urgell":58, "1_Catalunya":50,
↪ "2_Universitat":144}, "1_Catalunya":{"1_Universitat":50,
↪ "1_Urquinaona":58, "3_Catalunya":180, "6_Catalunya":360,
↪ "7_Catalunya":360}, "1_Urquinaona":{"1_Catalunya":58, "1_Arc de
↪ Triomf":85, "4_Urquinaona":256}, "1_Arc de
↪ Triomf":{"1_Urquinaona":85, "1_Marina":54}, "1_Marina":{"1_Arc
↪ de Triomf":54, "1_Glòries":80}, "1_Glòries":{"1_Marina":80,
↪ "1_Clot":78}, "1_Clot":{"1_Glòries":78,"1_Navas":65,
↪ "2_Clot":120}, "1_Navas":{"1_Clot":65, "1_La Sagrera":80},
↪ "1_La Sagrera":{"1_Navas":80, "1_Fabra i Puig":89, "5_La
↪ Sagrera":100, "9N_La Sagrera":178, "10_La Sagrera":178},
↪ "1_Fabra i Puig":{"1_La Sagrera":89, "1_Sant Andreu":101},
↪ "1_Sant Andreu":{"1_Fabra i Puig":101, "1_Torras i Bages":88},
↪ "1_Torras i Bages":{"1_Sant Andreu":88, "1_Trinitat Vella":77},
↪ "1_Trinitat Vella":{"1_Torras i Bages":77, "1_Baró de
↪ Viver":60}, "1_Baró de Viver":{"1_Trinitat Vella":60, "1_Santa
↪ Coloma":83}, "1_Santa Coloma":{"1_Baró de Viver":83,
↪ "1_Fondo":84}, "1_Fondo":{"1_Santa Coloma":84, "9N_Fondo":140},
↪ "2_Paral·lel":{"2_Sant Antoni":80, "3_Paral·lel":83}, "2_Sant
↪ Antoni":{"2_Paral·lel":80, "2_Universitat":66},
↪ "2_Universitat":{"2_Sant Antoni":66, "2_Passeig de Gràcia":80,
↪ "1_Universitat":144}, "2_Passeig de
↪ Gràcia":{"2_Universitat":80, "2_Tetuan":95, "3_Passeig de
↪ Gràcia":360, "4_Passeig de Gràcia":120}, "2_Tetuan":{"2_Passeig
↪ de Gràcia":95, "2_Monumental":93},
↪ "2_Monumental":{"2_Tetuan":93, "2_Sagrada Família":63},
↪ "2_Sagrada Família":{"2_Monumental":63, "2_Encants":114,
↪ "5_Sagrada Família":178}, "2_Encants":{"2_Sagrada Família":114,
↪ "2_Clot":53}, "2_Clot":{"2_Encants":53, "2_Bac de Roda":89,
↪ "1_Clot":120}, "2_Bac de Roda":{"2_Clot":89, "2_Sant
↪ Martí":61}, "2_Sant Martí":{"2_Bac de Roda":61, "2_La Pau":74},
↪ "2_La Pau":{"2_Sant Martí":74, "2_Verneda":76, "4_La Pau":60},
```

```
3 metro(graf_metro, "2_Paral·lel", "11_Casa de l'Aigua")
```

Exemple de sortida

Punt inicial: $2_{P}aral\Delta lel$
$Puntfinal: 11_{C}asadel'Aigua$
$Tempsnetdelrecorregut: 1245$
$Tempstotaldelrecorregut: 26minutsi6segons$
$Recorregut: [2_{P}aral\Delta lel, 2_{S}antAntoni, 2_{U}niversitat, 2_{P}asseigdeGrcia, 4_{P}asseigdeGrcia, 4_{G}irona,$