
Annexes TP graphes (2/2)

1	Algorithme de Dijkstra dans un réseau carré	1
2	Algorithme A* dans un réseau carré	2



1 Algorithme de Dijkstra dans un réseau carré

```
1 def DIJKSTRA_carre(n,dep,fin):
2     G=dico_voisins(n)
3     file=filePrio()
4     file=put(dep,0,file)
5     distances_origines={dep:0}
6     for v in G:
7         if v!=dep:
8             distances_origines[v]=float("inf")
9     parents={dep:None}
10    chemin=[]
11    plt.pause(0.1)
12    carre(n)
13    plt.plot(dep[0],dep[1],'bo')
14    plt.plot(fin[0],fin[1],'bo')
15    while not empty(file):
16        s,d=get(file)
17        if s!=fin:
18            for v,delta in G[s]:
19                distance=distances_origines[s]+delta
20                if distance<distances_origines[v]:
21                    distances_origines[v]=distance
22                    file=put(v,distance,file)
23                    parents[v]=s
24                    plt.plot(v[0],v[1],"ro")
25                    plt.pause(0.1)
26        else:
27            plt.plot(s[0],s[1],'bo')
28            chemin=[fin]
29            while s!=dep:
30                s=parents[s]
31                chemin=[s]+chemin
32                plt.plot(s[0],s[1],'bo')
33                plt.pause(0.1)
34    return distances_origines[fin],chemin
```

2 Algorithme A* dans un réseau carré

```

1 def algoA(n,dep,fin):
2     G=dico_voisins(n)
3     distances_origines={}
4     distances_origine={dep:0}
5     for v in G:
6         if v!=dep:
7             distances_origine[v]=float("inf")
8     h={} # heuristique
9     for v in G:
10         h[v]=euclide(v,fin) # distance de v à l'arrivée
11     score={}
12     score[dep]=h[dep]+distances_origine[dep] # distance origine+fin
13     file=filePrio()
14     file=put(dep,score[dep],file) # priorité au score le plus faible
15     parents={dep:None}
16     plt.pause(0.1)
17     carre(n)
18     plt.plot(dep[0],dep[1],'yo')
19     plt.plot(fin[0],fin[1],'yo')
20     while not empty(file):
21         s,d=get(file) # sommet et distance au parent
22         if s!=fin:
23             for v,delta in G[s]:
24                 #on cherche le sommet le plus proche de l'origine...
25                 #...et le moins loin de l'arrivée
26                 distance=distances_origine[s]+delta
27                 if distance<distances_origine[v]:
28                     distances_origine[v]=distance
29                     score[v]=distance+h[v]
30                     file=put(v,score[v],file)
31                     parents[v]=s
32                     plt.plot(v[0],v[1],"ro")
33                     plt.pause(0.1)
34             else:
35                 plt.plot(s[0],s[1],'yo')
36                 chemin=[fin]
37                 while s!=dep:
38                     s=parents[s]
39                     chemin=[s]+chemin
40                     plt.plot(s[0],s[1],'yo')
41                     plt.pause(0.1)
42                 return distances_origine[fin],chemin

```