Algorithme A*

Quentin Fortier

October 1, 2022

Rappels sur Dijkstra

Rappel : Dijkstra permet de trouver toutes les distances depuis un sommet de départ vers tous les autres sommets.

```
def dijkstra(G, s):
    n = len(G)
    dist = n*[float("inf")]
    dist[s] = 0
    q = PriorityQueue()
    for v in range(n):
        q.add(v, dist[v])

    while not q.is_empty():
        u = q.take_min()
        for v in range(n):
            d = dist[u] + G[u][v]
            if v in q and d < dist[v]:
                q.update(v, d)
                dist[v] = d
    return dist
```

Algorithme A*

La plupart du temps, on ne cherche pas les plus courts chemins à tous les autres sommets, mais seulement à un autre sommet fixé.

Exemples :

- GPS : Trouver un chemin le plus court d'une ville à une autre.
- Jeu vidéo : Déplacer un personnage d'un point à un autre.

Algorithme A*

La plupart du temps, on ne cherche pas les plus courts chemins à tous les autres sommets, mais seulement à un autre sommet fixé.

Exemples :

- GPS : Trouver un chemin le plus court d'une ville à une autre.
- Jeu vidéo : Déplacer un personnage d'un point à un autre.

Dans ce cas, l'**algorithme A*** peut être plus efficace.

Algorithme A*

But : Trouver un plus court chemin du sommet s au sommet t dans un graphe G.

Principe de l'algorithme A* :

1. Définir une fonction h telle que $h(v)$ soit une estimation de la distance de v à t.
2. Modifier l'algorithme de Dijkstra en utilisant `dist[u] + G[u][v] + h(v)` comme priorité, au lieu de `dist[u] + G[u][v]`.

Algorithme A*

Modifications de A* par rapport à Dijkstra :

```
def astar(G, s, §\hlg{t, h}§):
    n = len(G)
    dist = n*[float("inf")]
    dist[s] = 0
    q = PriorityQueue()
    for v in range(n):
        q.add(v, dist[v])
    while not q.is_empty():
        u = q.take_min()
        §\hlg{if u == t:}§
            §\hlg{return dist[t]}§
        for v in range(n):
            d = dist[u] + G[u][v]
            if v in q and d < dist[v]:
                q.update(v, §\hlg{d + h(v)}§)
                dist[v] = d
```

Choix de l'heuristique h

Choix possibles de l'heuristique h :

- Fonction nulle ($h = 0$) :

Choix de l'heuristique h

Choix possibles de l'heuristique h :

- Fonction nulle ($h = 0$) : On obtient Dijkstra.

Choix de l'heuristique h

Choix possibles de l'heuristique h :

- Fonction nulle ($h = 0$) : On obtient Dijkstra.
- Distance au sommet d'arrivée ($h(v) = d(v,t) - d(s,v)$) :

Choix de l'heuristique h

Choix possibles de l'heuristique h :

- Fonction nulle $(h = 0)$: On obtient Dijkstra.
- Distance au sommet d'arrivée $(h(v) = d(v,t) - d(s,v))$:
  Choisit à chaque étape le sommet v minimisant $d(v,t)$.
  $\Longrightarrow$ Explore seulement les sommets le long d'un plus court chemin.

Choix de l'heuristique h

Choix possibles de l'heuristique h :

- Fonction nulle ($h = 0$) : On obtient Dijkstra.
- Distance au sommet d'arrivée ($h(v) = d(v, t) - d(s, v)$) : Choisit à chaque étape le sommet v minimisant $d(v, t)$.
  $\Longrightarrow$ Explore seulement les sommets le long d'un plus court chemin.

Malheureusement, on ne connaît pas $d(v, t)$. On utilise donc souvent une heuristique qui approxime la distance au sommet d'arrivée, mais plus facile à calculer.

Choix de l'heuristique h

Choix possibles de l'heuristique h :

- Fonction nulle ($h = 0$) : On obtient Dijkstra.
- Distance au sommet d'arrivée ($h(v) = d(v,t) - d(s,v)$) : Choisit à chaque étape le sommet v minimisant $d(v,t)$.
  $\implies$ Explore seulement les sommets le long d'un plus court chemin.

Malheureusement, on ne connaît pas $d(v,t)$. On utilise donc souvent une heuristique qui approxime la distance au sommet d'arrivée, mais plus facile à calculer.
Par exemple, si les sommets sont des points dans $\mathbb{R}^2$ et $t = (t_1, t_2)$:

- Distance euclidienne ($h(x,y) = \sqrt{(x_1 - t_1)^2 + (x_2 - t_2)^2}$).
- Distance de Manhattan ($h(x,y) = |x_1 - t_1| + |x_2 - t_2|$).

Choix de l'heuristique h

Définition

On dit qu'une heuristique h est **admissible** si elle ne surestime jamais la distance au sommet d'arrivée, c'est-à-dire :

$$\forall v \in V,\ h(v) \leq d(v, t)$$

Choix de l'heuristique h

Définition

On dit qu'une heuristique h est **admissible** si elle ne surestime jamais la distance au sommet d'arrivée, c'est-à-dire :

$$\forall v \in V,\ h(v) \leq d(v, t)$$

Théorème (admis)

L'algorithme A* avec une heuristique admissible donne un chemin de poids minimum.