大部分问题都是图中的搜索问题(search problem)。

几个关系。

DFS ~ BFS ~ Dijkstra

The only difference is the data structure used in the algorithm.

- BFS/DFS

Initialization - O(n)

en\_DS(s)

while(not empty)

  u = de\_DS

  visit[u] = true

  for each (u, v)

    if not visit

      relax

      en\_DS

Same element in the DS above is allowed, since they will be filtered by the visit[] and the order of operation is totally decided by the DS.

- Dijkstra

Initialization - O(n)

en\_priority\_queue(s)

while(not empty)

  u = de\_queue

  visit[u] = true

  for each (u, v)

    if not visit

      en\_priority\_queue

In Dijkstra, multiple same element is not allowed cause it makes no sense. So each time the en\_queue operation would either be to update the key, which is decreasing the key exactly, or add a totally new element into the queue.

- Dijkstra

A\* ~ BFS ~ DFS ~ Dijkstra

- A\*: heuristic function: f(n) = g(n) + h(n), g(n) distance from source, h(n) estimated distance to target

- BFS: set h(n) = 0, so every time we will just take the ones generated before, since g(n) is increasing.

- DFS: set g(n) = 0, so every time we will just take the newly generated ones, since h(n) is decreasing.

- Dijkstra: set g(n) = distance from source, h(n) = 0, so we will always take the closest one.

BFS ~ Dijkstra ~ Prim

Consider BFS is a special case of Dijkstra. It means we assign every edge weight of 1. So while we take the closest vertex each time and update its adjacent vertices in Dijkstra, we take the adjacent vertices of the previous one instead. Or, think it as this opposite way. While we take the adjacent vertices of the previous one, we are now treating those with least weight as the most adjacent vertices.

Only modify the distance between the current vertex and the source vertex to the current vertex and its previous adjacent vertex, then we get the Prim Algorithm code from the Dijkstra Algorithm.

BFS ~ Bellman-Ford Algorithm ~ SPFA

In an acyclic graph, the longest path would consist of at most |V| - 1 edges. So in the Bellman-Ford we will do |V| - 1 calculation of relaxation of all the edges, quite like what the BFS does.

In the SPFA, we use a queue to record all the vertices waiting to be visited. If a vertex is updated in the relaxation, then we put it into queue again. Otherwise we can just leave out with it in order to cut repetitive operation. When there only exists one way to each vertex, it is BFS.

Hungarian Algorithm

- augmenting path, similar to one in the Ford-Fulkerson method

- using DFS strategy to find each possible augmenting path

Ford-Fulkerson method + Edmonds-Karp Algorithm

Method is a method. Its runtime is decided by the specific way of operation used in it. Algorithm is just replacing one step of finding augmenting path with BFS.

——

为什么

if (dis[u] + w(u, v) < dis[v])

  dis[u] = dis[v] + w(u, v)

这一段叫做松弛技术呢？(relax)

——

堆

二叉堆

二项树

二项堆

斐波那契堆