基本图算法

Yanjie Ze, May 2021

DFS

每个顶点只explore一次。每条边check两次：

$$T=O(|V|+|E|)$$

BFS

用link list队列实现的话O(1)，每个点入队出队一次，每条边检查两次：

$$T=O(|V|+|E|)$$

单源最短路

Dijkstra（无负权）

把点插入堆V次，提出堆V次，2E次再插入点。

如果用二叉堆：

$$T=O((|V|+|E|)log|V|)$$

Bellman-Ford（有负权）

Algorithm 3: Bellman-Ford Algorithm

foreach *node* $u \in V$ **do**
  $M[u] \leftarrow \infty$;
  $predecessor[u] \leftarrow \emptyset$;
$M[s] \leftarrow 0$;
for $i = 1$ *to* $n - 1$ **do**
  foreach *node* $u \in V$ **do**
    if $M[u]$ *has been updated in previous iteration* **then**
      foreach *edge* $(u, v) \in E$ **do**
        if $M[v] > M[u] + w(u, v)$ **then**
          $M[v] \leftarrow M[u] + w(u, v)$;
          $predecessor[v] \leftarrow u$;
  If no $M[v]$ changed in this iteration, stop.

$$RunningTime : T = O(|V| \times |E|)$$

多源最短路

Floyd-Warshall

Algorithm 4: Floyd-Warshall Algorithm

for $k \leftarrow 1$ *to* n **do**
  for $i \leftarrow 1$ *to* n **do**
    for $j \leftarrow 1$ *to* n **do**
      if $c_{ij} > c_{ik} + c_{kj}$ **then**
        $c_{ij} \leftarrow c_{ik} + c_{kj}$;

$$Running\,Time: O(|V|^3)$$

Johnson

首先用bellman-ford进行reweight：

$$T_1 = O(|V| \times |E|)$$

再对每个点用Dijkstra：

$$T_2 = O(|V| \times (|V| + |E|)log|V|) = O((|V|^2 + |V| \times |E|)log|V|)$$

再reweight回去，并获得最短路：

$$T_3 = O(|V|^2)$$

总共为：

$$T = O((|V|^2 + |V| \times |E|)log|V|)$$

网络流

Ford-Fulkerson

Algorithm 1: AUGMENT(f, c, P)

$\delta \leftarrow$ bottleneck capacity of augmenting path P;
foreach $e \in P$ **do**
  if $e \in E$ **then**
    $f(e) \leftarrow f(e) + \delta$; `/* forward edge */`
  else
    $f(e^R) \leftarrow f(e^R) - \delta$; `/* reverse edge */`
return f;

Algorithm 3: Ford-Fulkerson Algorithm

Input: $G = (V, E), c, s, t$
foreach $e \in E$ **do**
  $f(e) \leftarrow 0$;
$G_f \leftarrow$ residual graph;
while *there exists augmenting path* P **do**
  $f \leftarrow$ AUGMENT(f, c, P);
  update G_f;
return f;

Improved Ford-Fulkerson

Algorithm 4: Scaling Max-Flow Algorithm

Input: $G = (V, E), c, s, t$

foreach $e \in E$ **do**
    $f(e) \leftarrow 0$;
$G_f \leftarrow$ residual graph;
$\Delta \leftarrow$ smallest power of 2 greater than or equal to C;
while $\Delta \geq 1$ **do**
    $G_f(\Delta) \leftarrow \Delta$-residual graph;
    while *there exists augmenting path P in $G_f(\Delta)$* **do**
        $f \leftarrow \textsc{Argument}(f, c, P)$;
        update $G_f(\Delta)$;
    $\Delta \leftarrow \Delta/2$;
return f;

Find a max flow in $O(mlogC)$ augmentations.

Thus,

$$Running\,Time : O(m^2 logC)$$