Duyệt đồ thị

anhtt-fit@mail.hut.edu.vn

Duyệt đồ thị

- Ta cần thuật toán để duyệt đồ thị giống như duyệt cây
- Duyệt đồ thị có thể bắt đầu với 1 đỉnh ngẫu nhiên (duyệt cây bắt đầu với nút gốc)
- 2 khó khăn với duyệt đồ thị (không có với duyệt cây):
  - Đồ thị có thể có chu trình
  - Đồ thị có thể không kết nối
- 2 phương pháp duyệt quan trọng:
  - Duyệt theo chiều rộng, dựa trên tìm kiếm theo chiều rộng (breadth-first search - BFS).
  - Duyệt theo chiều sâu, dựa trên tìm kiếm theo chiều sâu (depth-first search - DFS).

Duyệt theo chiều rộng

Duyệt theo chiều rộng :

- Về cơ bản, gần giống duyệt theo từng nấc của cây.
- Khởi đầu với 1 đỉnh ngẫu nhiên
- Thăm tất cả các đỉnh lân cận
- Sau đó, thăm tất cả các đỉnh lân cận chưa thăm của các đỉnh vừa thăm
- Tiếp tục tiến trình đến khi tất cả các đỉnh đã thăm

Thuật toán BFS (Breadth-First Search)

Cài đặt với hàng đợi (queue):

- Thăm các đỉnh lân cận chưa được thăm của đỉnh hiện tại, đánh dấu nó, đưa đỉnh đó vào queue, thăm đỉnh tiếp theo
- Nếu không còn cạnh lân cận nào để thăm, lấy 1 đỉnh ra khỏi queue, gán nó là đỉnh hiện tại
- Nếu queue rỗng và không có đỉnh nào để đưa vào queue, quá trình duyệt kết thúc

BFS demo

- 51demo-bfs.ppt

Pseudocode

```
BFS(G,s)
   for each vertex u in V do
      visited[u] = false

   initialize an empty Q
   Enqueue(Q,s)

   While Q is not empty do
      u = Dequeue(Q)
      if not visited[u] then
            Report(u)
            visited[u] = true
            for each v in Adj[u] do
                  if not visited[v] then
                        Enqueue(Q,v)
```

Bài 1

- Cài đặt đồ thị sử dụng cây đỏ đen trong bài trước

  typedef JRB Graph;
  Graph createGraph();
  void addEdge(Graph graph, int v1, int v2);
  int adjacent(Graph graph, int v1, int v2);
  …

- Viết hàm duyệt đồ thị sử dụng thuật toán BFS

  void BFS(Graph graph, int start, int stop, void (*func)(int));
  - start = đỉnh đầu tiên thăm
  - stop = đỉnh cuối cùng thăm. Nếu stop = -1, tất cả các đỉnh đã được thăm
  - func = con trỏ trỏ đến hàm xử lý trên các đỉnh được thăm

Ví dụ

```
void printVertex(int v) { printf("%4d", v); }

   Graph g = createGraph();
   addEdge(g, 0, 1);
   addEdge(g, 1, 2);
   addEdge(g, 1, 3);
   addEdge(g, 2, 3);
   addEdge(g, 2, 4);
   addEdge(g, 4, 5);
   printf("\nBFS: start from node 1 to   5 : ");
   BFS(g, 1, 4, printVertex);
   printf("\nBFS: start from node 1 to all : ");
   BFS(g, 1, -1, printVertex);
```

Hướng dẫn

- Sử dụng cấu trúc dữ liệu danh sách nối kép trong libfdr để biểu diễn queue như sau:
- Tạo 1 queue
  - Dllist queue = new_dllist();
- Thêm 1 nút đã thăm
  - dll_append(queue, new_jval_i(v))
- Kiểm tra queue có rỗng không
  - dll_empty(queue)
- Lấy 1 đỉnh ra khỏi queue
  - node = dll_first(queue)
  - v = jval_i(node->val)
  - dll_delete_node(node)

Tìm kiếm sâu (Depth-First Search)

- Từ 1 nút đã cho, thăm 1 trong các đỉnh lân cận và không duyệt các đỉnh còn lại
- Thăm 1 trong các đỉnh lân cận của đỉnh vừa xét
- Tiếp tục quá trình này, duyệt đồ thị theo chiều sâu đến khi:
  - Gặp 1 nút đã thăm
  - Gặp đỉnh kết thúc

Duyệt theo chiều sâu

- Khởi tạo quá trình duyệt từ 1 đỉnh ngẫu nhiên
- Áp dụng tìm kiếm sâu
- Khi quá trình tìm kiếm kết thúc, quay lui đến đỉnh trước của điểm kết thúc
- Lặp lại tìm kiếm sâu trên các đỉnh lân cận khác, sau đó quay lui lại 1 mức trên
- Tiếp tục quá trình đến khi tất cả các đỉnh có thể đến được từ 1 đỉnh khởi đầu đã được thăm.
- Lặp lại các quá trình trên đến khi tất cả các đỉnh đã được thăm.

Thuật toán tìm kiếm sâu

- DFS được cài đặt với 1 ngăn xếp (stack) vì việc hồi qui và lập trình với stack là tương đương
- Thăm 1 đỉnh v
- Đẩy tất cả các đỉnh chưa thăm của v vào stack
- Đẩy 1 đỉnh ra khỏi stack để duyệt
- Lặp lại quá trình
- Nếu stack rỗng hoặc không có đỉnh nào được đẩy vào stack, quá trình duyệt kết thúc

DFS demo

- demo-dfs-undirected.ppt

Pseudocde

```
DFS(G,s)
   for each vertex u in V do
      visited[u] = false

   initialize an empty stack S
   Put(S, s)

   While S is not empty do
      u = Pop(S)
      if not visited[u] then
          Report(u)
          visited[u] = true
          for each v in Adj[u] do
             if not visited[v] then Put(S,v)
```

Bài 2

- Viết 1 hàm duyệt đồ thị theo chiều sâu

  void DFS(Graph graph, int start, int stop, void (*func)(int));
  - start = đỉnh đầu tiên thăm
  - stop = đỉnh cuối cùng thăm. Nếu stop = -1, tất cả các đỉnh đã được thăm
  - func = con trỏ trỏ đến hàm xử lý trên các đỉnh được thăm

Solution

- graph_traversal.c

Các ứng dụng

- Các đường được duyệt theo BFS hoặc DFS tạo thành cây (gọi là cây BFS hoặc cây DFS)
- Cây BFS cũng là đường đi ngắn nhất từ đỉnh xuất phát.
- Cây DFS được dùng để kiểm tra có tồn tại đường đi giữa 2 đỉnh không. Nó có thể dùng để xác định xem đồ thị có liên thông không

Bài 3

- Viết 1 hàm mới trong chương trình metro để tìm đường đi ngắn nhất giữa 2 sân ga.