Duyệt Đồ thị

Duyệt Đồ thị

- Chúng ta cần giải thuật duyệt đồ thị tương tự như đối với cây
- Duyệt đồ thị có thể bắt đầu tại một đỉnh bất kỳ (duyệt cây thường bắt đầu từ gốc):
  - Đồ thị có thể chứa chu trình;
  - Đồ thị có thể không liên thông
- Có hai phương pháp duyệt quan trọng
  - Duyệt theo chiều rộng (BFS).
  - Duyệt theo chiều sâu (DFS).

Duyệt theo chiều rộng

Duyệt đồ thị theo chiều rộng:

- Gần tương tự duyệt cây đã sắp xếp theo từng mức
- Bắt đầu từ một đỉnh bất kỳ
- Thăm tất cả các đỉnh liền kề
- Sau đó thăm tất cả các đỉnh liền kề chưa được thăm của các đỉnh đã được thăm ở mức trước
- Lặp lại thao tác cho đến khi tất cả các đỉnh đều được thăm

Giải thuật BFS

- Cài đặt bằng hàng đợi
- Thăm một đỉnh liền kề chưa được thăm của đỉnh hiện tại, đánh dấu, chèn đỉnh vào hàng đợi, thăm đỉnh tiếp theo
- Nếu không còn đỉnh liền kề để thăm, xóa một đỉnh khỏi hàng đợi (nếu có thể) và coi đó là đỉnh hiện tại
- Nếu hàng đợi rỗng và không còn đỉnh nào để chèn vào hàng đợi, 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 tập 1

- Cài đặt đồ thị bằng cây đỏ đen như buổi 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 giải thuật BFS

```
void BFS(Graph graph, int start, int stop, void (*func)(int));
```

  - start là đỉnh đầu tiên được thăm
  - stop là đỉnh cuối cùng được thăm, nếu stop = -1, thăm tất cả các đỉnh
  - func là con trỏ hàm xử lý 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, 5, printVertex);
    printf("\nBFS: start from node 1 to all : ");
    BFS(g, 1, -1, printVertex);
```

Gợi ý

- Sử dụng danh sách liên kết đôi trong thư viện libfdr để biểu diễn hàng đợi như sau:
- Tạo hàng đợi
  - Dllist queue = new_dllist();
- Thêm đỉnh vào hàng đợi
  - dll_append(queue, new_jval_i(v))
- Kiểm tra hàng đợi rỗng
  - dll_empty(queue)
- Lấy một đỉnh từ hàng đợi
  - node = dll_first(queue)
  - v = jval_i(node->val)
  - dll_delete_node(node)

Tìm kiếm theo chiều sâu

- Từ một đỉnh cho trước, thăm một đỉnh liền kề
- Sau đó thăm một đỉnh liền kề của đỉnh đó
- Thực hiện tiếp quá trình, duyệt đồ thị sâu nhất có thể tới khi
  - Gặp một đỉnh đã được thăm
  - Gặp đỉnh kết thúc

Tìm kiếm theo chiều sâu

- Bắt đầu duyệt từ một đỉnh bất kỳ
- Áp dụng tìm kiếm theo chiều sâu
- Khi việc tìm kiếm kết thúc, quay lui về đỉnh trước của điểm kết thúc
- Lặp lại tìm kiếm trên các đỉnh liền kề khác, sau đó quay lui lên một mức.
- Tiếp tục quá trình tới khi tất cả các đỉnh có thể thăm được từ đỉnh bắt đầu đã được thăm
- Lặp lại quá trình cho tới khi tất cả các đỉnh đã được thăm

Giải thuật DFS

- DFS có thể được cài đặt bằng ngăn xếp; giải thuật đệ quy và ngăn xếp là tương đương
- Thăm một đỉnh v
- Đẩy tất cả các đỉnh liền kề chưa được thăm của v vào ngăn xếp
- Lấy một đỉnh ra khỏi ngăn xếp cho tới khi gặp đỉnh chưa được thăm
- Lặp lại các bước trên
- Nếu ngăn xếp rỗng và không có đỉnh nào để đẩy vào ngăn xếp, 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 tập 2

- Tiếp tục viết hàm để duyệt đồ thị theo giải thuật DFS

  void DFS(Graph graph, int start, int stop, void (*func)(int));

  - start là đỉnh bắt đầu
  - stop là đỉnh kết thúc, nếu stop = -1, có thể thăm tất cả các đỉnh
  - func là con trỏ hàm xử lý đỉnh được thăm

Lời giải

- graph_traversal.c

Ứng dụng

- Đường đi tạo bởi BFS/DFS tạo thành một cây (gọi là cây BFS/DFS).
- Cây BFS T chứa đường đi ngắn nhất từ gốc: Mọi đỉnh v có đường đi từ s trong T và là đường đi ngắn nhất giữa v và s trong G.
- DFS được dùng để kiểm tra đường đi giữa hai đỉnh. Có thể được sử dụng để kiểm tra tính liên thông của đồ thị

Bài tập 3

- Thêm một chức năng vào chương trình metro để tìm đường đi ngắn nhất giữa hai trạm