# Duyệt đồ thị

# Duyệt đồ thị

- Chúng ta cần giải thuật duyệt đồ thị, tương tự như duyệt cây
- Duyệt đồ thị có thể bắt đầu tại một đỉnh bất kì (duyệt cây thường bắt đầu tại đỉnh gốc)
- Hai khó khăn trong duyệt đồ thị (không phải trong duyệt cây)
  - Đồ thị có thể bao gồm chu trình
  - Đồ thị có thể không liên thông
- Có hai cách duyệt quan trọng
  - BFS: duyệt theo chiều rộng (Breadth First Search)
  - DFS: duyệt theo chiều sâu (Deepth First Search)

# Giải thuật BFS

- Về cơ bản giống duyệt theo từng mức (level) của cây có thứ tự
- Bắt đầu duyệt tại đỉnh bất kì
- Thăm tất cả các đỉnh liền kề với nó
- Sau đó, thăm tất cả các đỉnh liền kề chưa được thăm của những đỉnh đã thăm mức cuối cùng
- Tiếp tục cho đến khi tất cả các đỉnh đã được thăm

# Giải thuật BFS

- Được thực thi với hàng đợi queue
- Thăm một đỉnh liền kề chưa được thăm từ đỉnh hiện tại, đánh dấu nó, chèn vào queue, thăm đỉnh tiếp theo
- Nếu không còn đỉnh liền kề nào chưa được thăm, lấy một đỉnh khỏi queue và coi là đỉnh hiện tại
- Nếu queue rỗng, như vậy không còn đỉnh nào để đưa vào queue, quá trình duyệt hoàn thành

# BFS demo

- 51demo-bfs.ppt

# Mã giả

```
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

- Thực thi một graph sử dụng red black tree với các hàm như bài thực hành tuần 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 graph sử dụng giải thuật 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 thì duyệt qua tất cả các đỉnh có thể
  - func: con trỏ hàm, trỏ tới hàm xử lý đối với một đỉnh đã thăm (như hiển thị…)

# 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);
```

# Hướng dẫn

- Sử dụng cấu trúc dữ liệu danh sách liên kết đôi trong libfdr để biểu diễn queue
- Để tạo queue
  - Dllist queue = new_dllist();
- Để thêm một nút đã thăm
  - dll_append(queue, new_jval_i(v))
- Để kiểm tra xem queue có rỗng không?
  - dll_empty(queue)
- Để lấy ra một đỉnh khỏi queue
  - node = dll_first(queue)
  - v = jval_i(node->val)
  - dll_delete_node(node)

# Giải thuật DFS

- Từ đỉnh hiện tại, thăm một trong các đỉnh liền kề của nó
- Tiếp tục, thăm một trong các đỉnh liền kề của đỉnh trước đó
- Lặp lại quá trình, thăm đồ thị ở mức sâu nhất có thể, cho đến khi
  - Gặp lại đỉnh đã thăm
  - Gặp đỉnh kết thúc
- .

# Giải thuật DFS

- Bắt đầu duyệt từ một đỉnh bất kì
- Thực hiện tìm kiếm theo chiều sâu
- Khi quá trình tìm kiếm dừng, quay lui lại đỉnh trước điểm kết thúc
- Lặp lại tìm kiếm theo chiều sâu trên cácđỉnh lân cận, sau đó quay lui lên một mức trên
- Lặp lại quá trình tìm kiếm cho tới khi tất cả các đỉnh đều được thăm

# Giải thuật DFS

- Cài đặt DFS sử dụng stack
- Thăm đỉnh hiện tại v
- Push tất cả các đỉnh liền kề chưa được thăm của v vào stack
- Pop một đỉnh khỏi stack cho tới khi đỉnh này chưa được thăm
- Coi đỉnh này là đỉnh hiện tại, lặp lại các bước
- Nếu stack rỗng => kết thúc quá trình duyệt

# DFS demo

- demo-dfs-undirected.ppt

# Mã giả

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

    initialize an empty stack S
    Push(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 Push(S,v)
```

# Bài 2

- Duyệt đồ thị sử dụng giải thuật DFS

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

  - start: là đỉnh đầu tiên thăm
  - stop: là đỉnh cuối cùng thăm, nếu stop = -1 thăm tất cả các đỉnh có thể
  - func là con trỏ hàm, trỏ tới hàm xử lý đỉnh đã thăm (như hiển thị…)
  - func is a pointer to the function that process on the visited vertices

# Gợi ý

- Cài đặt stack dưới dạng queue (như Bài tập 1)
- Khác biệt
  - Queue: lấy phần tử đầu tiên khỏi danh sách
    - Hàm dll_first(queue)
  - Stack: lấy phần tử cuối cùng khỏi danh sách
    - Hàm ?

# Ứng dụng

- Đường đi được duyệt bởi BFS hay DFS tạo thành một cây, gọi là cây BFS hay DFS
- Cây BFS là đường đi ngắn nhất từ đỉnh gốc tới các đỉnh khác trong đồ thị
- Cây DFS được sử dụng để kiểm tra xem có tồn tại đường đi giữa hai đỉnh hay không, kiểm tra tính liên thông của đồ thị

# Bài 3

- Thêm hàm trong chương trình tàu điện (tuần 6) để tìm đường đi ngắn nhất giữa hai nhà ga