HW #8
Laboratory

Hsin-Hung Lin (林信宏)

hlin@iis.sinica.edu.tw

2018/12/24

Breadth-First Search

```
Algorithm Breadth_First_Search(G, v);
begin
    mark v;
    put v in a queue;
    while the queue is not empty do
        remove vertex w from the queue;
        perform preWORK on w;
        for all edges (w, x) with x unmarked do
            mark x;
            add (w, x) to the BFS tree T;
            put x in the queue
end
```

1. (Lemma 1)

Lemma 1 of BFS

If an edge **(u,w)** belongs to a BFS tree T such that u is a parent of w, then u has the minimal BFS number among vertices with edges leading to w.

Symbols: G = (V,E)
T: BFS tree
Q: queue

Breadth-First Search

```
Algorithm Breadth_First_Search(G, v);
begin
    mark v;
    put v in a queue;
    while the queue is not empty do
        remove vertex w from the queue;
        perform preWORK on w;
        for all edges (w, x) with x unmarked do
            mark x;
            add (w, x) to the BFS tree T;
            put x in the queue
end
```

```
// BFS numbering
num++;
w.bfs_num = num;
```

1. (Lemma 1)

Proof - preparation

a. Add BFS numbering.
b. BFS number is the ordering of being pushed to Q (the queue).
c. When a vertex is processed in BFS, all its unmarked children will be pushed to Q.
d. To have lower BFS number than u, we can see if there exists a vertex, say v, which is another parent of w, and is pushed to Q before u.

1. (Lemma 1)

Proof by contradiction

a. Assume there exists such v, since v is also parent of w, and there exists an edge (v,w).
b. When v is processed, (v,w) is added to T, and w is marked and pushed to Q.
c. When u is later processed, (u,w) will not be added to T since w is already marked.
d. proved by contradiction.

2. (Lemma 2)

Lemma 2 of BFS

For each vertex w, the path from the root to w in T is a shortest path from the root to w in G.

Symbols: G = (V,E)
T: BFS tree
Q: queue

Breadth-First Search

```
Algorithm Breadth_First_Search(G, v);
begin
    mark v;
    put v in a queue;
    while the queue is not empty do
        remove vertex w from the queue;
        perform preWORK on w;
        for all edges (w, x) with x unmarked do
            mark x;
            add (w, x) to the BFS tree T;
            put x in the queue
end
```

```
// after mark x do
x.d = w.d+1;
```

2. (Lemma 2)

Proof - preparation

a. Add level (depth) attribute to each vertex.
b. Proof by induction (w/ contradiction)

2. (Lemma 2)

Proof

a. Each vertex in T has only one path from root, and its length is its distance (i.e. v.d)
b. let dist(v) be the shortest path distance, we want to prove that path from root to v in T (denote as path(v)) is a shortest path and v.d = dist(v)

2. (Lemma 2)

Proof by induction

c. Base case: level=0 (root) → s.d = dist(s) = 0

d. Induction hypothesis: for vertice of level n > 0, select a vertex k: path(k) in T is a shortest path from the root to k in G. k.d = dist(k)

2. (Lemma 2)

Proof by induction – w/ contraduction

d. let u be the child of k in T
e. u.d = k.d + 1 = dist(k) + 1
f. if path(u) in T is not the shortest path in G, that is, dist(k) + 1 > dist(u), there should be another vertex w such that (w,u) in the shortest path in G.

2. (Lemma 2)

Proof by induction – w/ contraduction

h. Note that (w,u) is not in T
i. dist(w) + 1 = dist(u) < dist(k) + 1
j. dist(w) < dist(k) = k.d
k. w.d < k.d (w should be in T and a lower level than k)
m. (w,u) should be in T instead of (k,u)
n. Proved by contradiction

3. (Lemma 3)

Lemma 3 of BFS

If (v,w) is an edge in E that does not belong to T, then it connects two vertices whose level number differ by at most one.

Symbols: G = (V,E)
T: BFS tree
Q: queue

Breadth-First Search

```
Algorithm Breadth_First_Search(G, v);
begin
    mark v;
    put v in a queue;
    while the queue is not empty do
        remove vertex w from the queue;
        perform preWORK on w;
        for all edges (w, x) with x unmarked do
            mark x;
            add (w, x) to the BFS tree T;
            put x in the queue
end
```

```
// BFS numbering
num++;
w.bfs_num = num;
```

```
// after mark x do
x.d = w.d+1;
```

3. (Lemma 3)

Proof - preparation

a. Add BFS numbering
b. Add level (depth) attribute to each vertex.
c. Apply Lemma 1
d. Apply Lemma 22.3 [C]
e. Proof by property related to queue

3. (Lemma 3)

Lemma 22.3 [C]

Suppose that during the execution of BFS on a graph G = (V,E), the queue Q contains the vertices <v(1),v(2),...,v(r)> where v(1) is the head of Q and v(r) is the tail. Then, v(r).d <= v(1).d + 1 and v(i).d <= v(i+1).d for i = 1,2,...,r-1.

3. (Lemma 3)

Lemma 22.3 [C] – proof sketch

- Base case (root): trivial
- Induction case:
  a. Induction hyppthesis: Q = <v(1),v(2),...,v(r)> holds
  b. Case DEQUEUE: check v(r).d <= v(2).d + 1
  c. Case ENQUEUE: check v(r+1).d <= v(1).d + 1 & v(r).d <= v(r+1).d

3. (Lemma 3)

Proof

a. Since (v,w) is not an edge in T, let v.bfs_num < w.bfs_num
b. From lemma 1, there exists another vertex, say u, such that (u,w) is in T and u is a parent of w.
c. u.bfs_num is the minimum among vertices with edges to w.
d. u.bfs_num < v.bfs_num < w.bfs_num
e. Ordering of en(de)-queue: u->v->w.

3. (Lemma 3)

Proof

g. When u is dequeued, (u,w) will be scanned and w will be enqueued to Q
h. The ordering of enqueue suggests that v is also enqueued before w
i. From Lemma 22.3[C], we get w.d<=v.d+1
j. v's and w's level numbers differ by at most one