

Objectives

- Wrapping up implementing BFS and DFS
- Graph Application: Bipartite Graphs
- Directed Graphs

Jan 28, 2013

CSCI211 - Sprenkle

1

Review: Comparing BFS vs DFS

- What do they do?
- How are their outcomes different?
- What is the runtime of BFS? DFS?

Jan 28, 2013

CSCI211 - Sprenkle

2

Implementing DFS

- Keep nodes to be processed in a *stack*

```
DFS(s):
  Initialize S to be a stack with one element s
  Explored[v] = false, for all v } O(n)
  Parent[v] = 0, for all v
  DFS tree T = {}
  while S != {}:
    Take a node u from S
    if Explored[u] = false
      Explored[u] = true
      Add edge (u, Parent[u]) to T (if u ≠ s)
      for each edge (u, v) incident to u
        Add v to the stack S
        Parent[v] = u
```

How many times is a node added/removed from the stack?

Jan 28, 2013

CSCI211 - Sprenkle

3

Analyzing DFS

 $O(n+m)$

```
DFS(s):
  Initialize S to be a stack with one element s
  Explored[v] = false, for all v } O(n)
  Parent[v] = 0, for all v
  DFS tree T = {}
  while S != {}:
    Take a node u from S
    if Explored[u] = false
      Explored[u] = true
      Add edge (u, Parent[u]) to T (if u ≠ s)
      deg(u) for each edge (u, v) incident to u
        Add v to the stack S
        Parent[v] = u
```

A node is added/removed from the stack $2m = O(m)$ times

Jan 28, 2013

CSCI211 - Sprenkle

4

Analyzing Finding All Connected Components

- How can we find set of all connected components of graph?

```
R* = set of connected components (a set of sets)
while there is a node that does not belong to R*
  select s not in R*
  R = {s}
  while there is an edge (u,v) where u ∈ R and v ∉ R
    add v to R
  Add R to R*
```

But the inner loop is $O(m+n)$!
How can this RT be possible?

Running time: $O(m+n)$

Jan 28, 2013

CSCI211 - Sprenkle

5

Set of All Connected Components

- How can we find set of all connected components of graph?

```
R* = set of connected components (a set of sets)
while there is a node that does not belong to R*
  select s not in R*
  R = {s}
  while there is an edge (u,v) where u ∈ R and v ∉ R
    add v to R
  Add R to R*
```

Imprecision in the running time
of inner loop: $O(m+n)$

But that's m and n of the
connected component,
let's say m_i and n_i .
 $\sum_i O(m_i + n_i) = O(m+n)$

Where i is the subscript of the
connected component

Jan 28, 2013

CSCI211 - Sprenkle

6

BIPARTITE GRAPHS

Jan 28, 2013

CSCI211 - Sprenkle

7

Bipartite Graphs

- **Def.** An undirected graph $G = (V, E)$ is **bipartite** if the nodes can be colored **red** or **blue** such that every edge has one red and one blue end

➤ Generally: vertices divided into sets X and Y

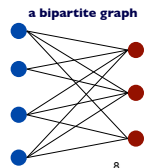
- Applications:

➤ Stable marriage:

- men = red, women = blue

➤ Scheduling:

- machines = red, jobs = blue



Jan 28, 2013

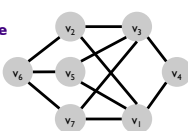
CSCI211 - Sprenkle

8

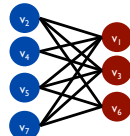
Testing Bipartiteness

- Given a graph G , is it bipartite?
- Many graph problems become:
 - Easier if underlying graph is bipartite (e.g., matching)
 - Tractable if underlying graph is bipartite (e.g., independent set)
- Before designing an algorithm, need to understand structure of bipartite graphs

a bipartite graph G :



another drawing of G :



Jan 28, 2013

CSCI211 - Sprenkle

9

How Can We Determine if a Graph is Bipartite?

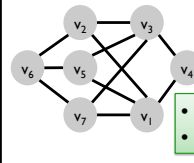
- Given a connected graph

Why connected?

1. Color one node red

- Doesn't matter which color (Why?)

➤ What should we do next?



- How will we know when we're finished?
- What does this process sound like?

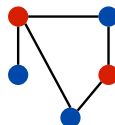
Jan 28, 2013

CSCI211 - Sprenkle

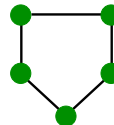
10

An Obstruction to Bipartiteness

- **Lemma.** If a graph G is bipartite, it cannot contain an odd-length cycle.



bipartite
(2-colorable)



not bipartite
(not 2-colorable)

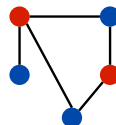
Jan 28, 2013

CSCI211 - Sprenkle

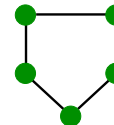
11

An Obstruction to Bipartiteness

- **Lemma.** If a graph G is bipartite, it cannot contain an odd-length cycle.
- **Pf.** Not possible to 2-color the odd cycle, let alone G .



bipartite
(2-colorable)



not bipartite
(not 2-colorable)

If find an odd cycle,
graph is NOT bipartite

Jan 28, 2013

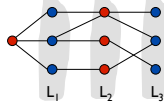
CSCI211 - Sprenkle

12

How Can We Determine if a Graph is Bipartite?

- Given a connected graph
 - Color one node red
 - Doesn't matter which color (Why?)
 - What should we do next?
- How will we know that we're finished?
- What does this process sound like?
 - BFS: alternating colors, layers

How can we implement the algorithm?



Jan 28, 2013

CSCI211 - Sprenkle

13

Review: BFS Implementation

```

BFS(s):
  Discovered[v] = false, for all v
  Discovered[s] = true
  L[0] = {s}
  layer counter i = 0
  BFS tree T = {}
  while L[i] != {}
    L[i+1] = {}
    For each node u ∈ L[i]
      Consider each edge (u,v) incident to u
      if Discovered[v] == false then
        Discovered[v] = true
        Add edge (u, v) to tree T
        Add v to the list L[i + 1]
    i+=1
  
```

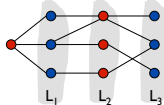
Jan 28, 2013

CSCI211 - Sprenkle

14

Implementing Algorithm

- Modify BFS to have a Color array
- When add v to list L[i+1]
 - Color[v] = red if i+1 is even
 - Color[v] = blue if i+1 is odd



What is the running time of this algorithm? $O(n+m)$

Marks a change in how we think about algorithms
to apply known algorithms to solve new problems

Jan 28, 2013

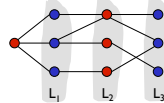
CSCI211 - Sprenkle

15

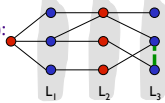
Analyzing Algorithm's Correctness

- Lemma.** Let G be a connected graph, and let L_0, \dots, L_k be the layers produced by BFS starting at node s. Exactly one of the following holds:
 - (i) No edge of G joins two nodes of the same layer
→ G is bipartite
 - (ii) An edge of G joins two nodes of the same layer
→ G contains an odd-length cycle and hence is not bipartite

Case (i):



Case (ii):



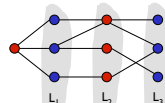
Jan 28, 2013

CSCI211 - Sprenkle

16

Analyzing Algorithm's Correctness

- Lemma.** Let G be a connected graph, and let L_0, \dots, L_k be the layers produced by BFS starting at node s. Exactly one of the following holds:
 - (i) No edge of G joins two nodes of the same layer
→ G is bipartite
- Pf. (i)**
 - No edge joins two nodes in the same layer
 - Implies all edges join nodes on adjacent level
 - Bipartition
 - red = nodes on odd levels
 - blue = nodes on even levels



Jan 28, 2013

CSCI211 - Sprenkle

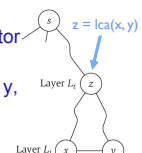
17

Analyzing Algorithm's Correctness

- Lemma.** Let G be a connected graph, and let L_0, \dots, L_k be the layers produced by BFS starting at node s. Exactly one of the following holds:
 - (ii) An edge of G joins two nodes of the same layer → G contains an odd-length cycle and hence is not bipartite

Pf. (ii)

- Suppose (x, y) is an edge with x, y in same level L_j .
- Let z = lca(x, y) = lowest common ancestor
- Let L_i be level containing z
- Consider cycle that takes edge from x to y, then path $y \rightarrow z$, then path from $z \rightarrow x$



Jan 28, 2013

CSCI211 - Sprenkle

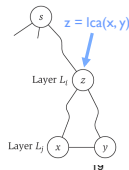
18

Analyzing Algorithm's Correctness

- **Lemma.** Let G be a connected graph, and let L_0, \dots, L_k be the layers produced by BFS starting at node s . Exactly one of the following holds:
 - (i) An edge of G joins two nodes of the same layer \rightarrow G contains an odd-length cycle and hence is not bipartite

- Pf. (ii)

- Suppose (x, y) is an edge with x, y in same level L_j .
- Let $z = \text{lca}(x, y)$ = lowest common ancestor
- Let L_i be level containing z
- Consider cycle that takes edge from x to y , then path $y \rightarrow z$, then path $z \rightarrow x$
- Its length is $1 + (j-i) + (j-i)$, which is odd



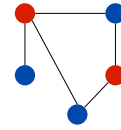
Jan 28, 2013

CSCI211 - Sprenkle

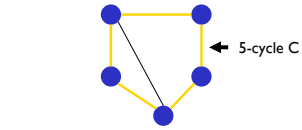
20

An Obstruction to Bipartiteness

- **Corollary.** A graph G is bipartite *iff* it contains no odd length cycle.

bipartite
(2-colorable)

Jan 28, 2013

not bipartite
(not 2-colorable)

CSCI211 - Sprenkle

20

DIRECTED GRAPHS

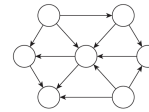
Jan 28, 2013

CSCI211 - Sprenkle

21

Directed Graphs $G = (V, E)$

- Edge (u, v) goes from node u to node v



- **Example:** Web graph - hyperlink points from one web page to another
 - Directedness of graph is crucial
 - Modern web search engines exploit hyperlink structure to rank web pages by importance

Jan 28, 2013

CSCI211 - Sprenkle

22

Representing Directed Graphs

- For each node, keep track of
 - Out edges (where links go)
 - In edges (from where links come in)
- Could only store *out* edges
 - Figure out *in* edges with increased computation/time
 - Useful to have both *in* and *out* edges

Jan 28, 2013

CSCI211 - Sprenkle

23

Rock Paper Scissors Lizard Spock



Jan 28, 2013

CSCI211 - Sprenkle

24

CONNECTIVITY IN DIRECTED GRAPHS

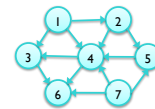
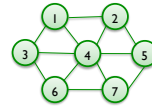
Jan 28, 2013

CSCI211 - Sprenkle

25

Graph Search

- How does **reachability** change with directed graphs?



- Example: Web crawler
 - Start from web page s .
 - Find all web pages linked from s , either directly or indirectly.

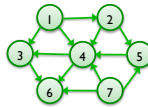
Jan 28, 2013

CSCI211 - Sprenkle

26

Graph Search

- Directed reachability.** Given a node s , find all nodes reachable from s .
- Directed s - t shortest path problem.** Given two nodes s and t , what is the length of the shortest path between s and t ?
 - Not necessarily the same as $t \rightarrow s$ shortest path
- Graph search.** BFS and DFS extend naturally to directed graphs
 - Trace through out edges
 - Run in $O(m+n)$ time



Jan 28, 2013

CSCI211 - Sprenkle

27

Problem

- Find all nodes with paths **to** s
 - Rather than paths from s to other nodes

Jan 28, 2013

CSCI211 - Sprenkle

28

Problem/Solution

- Problem.** Find all nodes with paths **to** s
- Solution.** Run BFS on *in edges* instead of out edges

Jan 28, 2013

CSCI211 - Sprenkle

29

Strong Connectivity

- Def.** Node u and v are **mutually reachable** if there is a *path* from $u \rightarrow v$ and also a *path* from $v \rightarrow u$ (not necessarily a direct edge)
- Def.** A graph is **strongly connected** if every pair of nodes is mutually reachable
- Lemma.** Let s be any node. G is strongly connected **iff** every node is reachable from s and s is reachable from every node



Jan 28, 2013

CSCI211 - Sprenkle

30

Strong Connectivity

- If u and v are mutually reachable and v and w are mutually reachable, then u and w are mutually reachable

Jan 28, 2013

CSCI211 - Sprenkle

31

Strong Connectivity

- If u and v are mutually reachable and v and w are mutually reachable, then u and w are mutually reachable.
- Proof.** We need to show that there is a path from $u \rightarrow w$ and from $w \rightarrow u$.
 - By defn of mutually reachable
 - There is a path $u \rightarrow v$ & a path $v \rightarrow u$
 - There is a path $v \rightarrow w$, and a path $w \rightarrow v$
 - Take path $u \rightarrow v$ and then from $v \rightarrow w$
 - Path from $u \rightarrow w$
 - Similarly for $w \rightarrow u$

Jan 28, 2013

CSCI211 - Sprenkle

32

Strong Connectivity

- Def.** A graph is **strongly connected** if every pair of nodes is mutually reachable
- Lemma.** Let s be any node. G is **strongly connected** iff every node is reachable from s and s is reachable from every node.
 - 1st prove \Rightarrow
 - 2nd prove \Leftarrow
 - for any nodes u and v , is there a path $u \rightarrow v$ and $v \rightarrow u$?

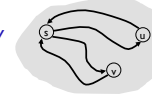
Jan 28, 2013

CSCI211 - Sprenkle

33

Strong Connectivity

- Def.** A graph is strongly connected if every pair of nodes is mutually reachable
- Lemma.** Let s be any node. G is **strongly connected** iff every node is reachable from s , and s is reachable from every node.
- Pf.** \Rightarrow Follows from definition of strongly connected
- Pf.** \Leftarrow For any nodes u and v , make path $u \rightarrow v$ and $v \rightarrow u$
 - $u \rightarrow v$: concatenate $u \rightarrow s$ with $s \rightarrow v$
 - $v \rightarrow u$: concatenate $v \rightarrow s$ with $s \rightarrow u$



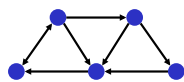
Jan 28, 2013

CSCI211 - Sprenkle

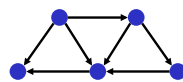
34

Strong Connectivity Problem

- Determine if G is strongly connected in $O(m + n)$ time



strongly connected



not strongly connected

Hint: Can we leverage any algorithms we know have $O(m+n)$ time?

Jan 28, 2013

CSCI211 - Sprenkle

35

Strong Connectivity: Algorithm

- Theorem.** Can determine if G is strongly connected in $O(m + n)$ time.
- Pf.**
 - Pick any node s
 - Run BFS from s in G
 - Run BFS from s in G_{rev}
 - reverse orientation of every edge in G
 - Or, the BFS using the in edges
 - Return true iff all nodes reached in both BFS executions
 - Correctness follows immediately from previous lemma
 - All reachable from one node, s is reached by all

Jan 28, 2013

CSCI211 - Sprenkle

36

Strong Components

- For any two nodes s and t in a directed graph, their strong components are either identical or disjoint

Hint: Consider a node in common...

Jan 28, 2013

CSCI211 - Sprenkle

37

Strong Components

- For any two nodes s and t in a directed graph, their strong components are either identical or disjoint
- Proof.**
 - Consider v in both strong components
 - $s \rightarrow v; v \rightarrow s; v \rightarrow t; t \rightarrow v \Rightarrow t \rightarrow s, s \rightarrow t$ (mutually reachable)
 - As soon as there is one common node, then have identical strong components
 - On the other hand, consider s and t are not mutually reachable
 - No node v that is in the strong component of each
 - What would it mean if there were?

Jan 28, 2013

CSCI211 - Sprenkle

38

Looking Ahead

- Summaries of readings associated with last week's lectures are due Tuesday night
- Problem Set 3 due Friday by class

Jan 28, 2013

CSCI211 - Sprenkle

39