Big O Notation and Graphs

CMSC 132 - Week 9
Graphs API

- A graph is a pair \((V, E)\), where
  - \(V\) is a set of nodes, called vertices
  - \(E\) is a collection of pairs of vertices, called edges
  - Vertices and edges can be objects that store some information.

- Example:
  - A vertex represents an airport and stores the 3-letter airport code
  - An edge represents a flight route between 2 airports, and stores the route mileage

Figure from Dr. Noha Adly’s lectures, Alexandria University
Graphs API

- Edge Types:
  - Directed edge: ordered pair of vertices \((u, v)\), first vertex is the source
  - Undirected edge: unordered pair of vertices \((u, v)\)
Terminology

- End vertices or endpoints of an edge \((u, v)\)
- Degree of a vertex in an undirected graph (in-degree and out-degree for directed graphs)
- Parallel edges: \(h\) and \(i\) are parallel edges
- Self-loops: \(j\) is a self-loop
Graph Representation

List of Edges

(A,B) → (A,C) → (A,D) → (B,E) → (B,F) → (C,F)

Adjacency List

Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
# Graph Representation

<table>
<thead>
<tr>
<th>representation</th>
<th>space</th>
<th>add edge</th>
<th>edge between v and w?</th>
<th>iterate over vertices adjacent to v?</th>
</tr>
</thead>
<tbody>
<tr>
<td>list of edges</td>
<td>( E )</td>
<td>1</td>
<td>( E )</td>
<td>( E )</td>
</tr>
<tr>
<td>adjacency matrix</td>
<td>( V^2 )</td>
<td>1 *</td>
<td>1</td>
<td>( V )</td>
</tr>
<tr>
<td>adjacency lists</td>
<td>( E + V )</td>
<td>1</td>
<td>degree(v)</td>
<td>degree(v)</td>
</tr>
</tbody>
</table>

* disallows parallel edges
Depth-first Search

```java
public class DepthFirstPaths {
    private boolean[] marked;
    private int[] edgeTo;
    private int s;

    public DepthFirstSearch(Graph G, int s) {
        ...
        dfs(G, s);
    }

    private void dfs(Graph G, int v) {
        marked[v] = true;
        for (int w : G.adj(v))
            if (!marked[w])
                dfs(G, w);
        edgeTo[w] = v;
    }
}
```
Depth-first Search Recursive

for all vertices X
  X.tag = false
create a new queue
enqueue the start vertex in the queue
set the start vertex's tag to true
while (the queue is not empty)
  take a vertex X out of the queue
  process X
  for each neighbor Y of X
    if (Y.tag == false)
      enqueue Y in the queue
      Y.tag = true
Depth-first Search

Example

- A (unexplored vertex)
- • A (visited vertex)
- ---- (unexplored edge)
- →→ (discovery edge)
- ──── (back edge)

Figure from Dr. Noha Adly's lectures, Alexandria University
Depth-first Search

Example (cont.)

A-B-C-D-E

Figure from Dr. Noha Adly’s lectures, Alexandria University
Depth-first Nonrecursive Pseudocode

for all vertices X
    X.tag = false
create a new stack
push the start vertex onto the stack
set the start vertex’s tag to true
while (the stack is not empty)
    pop a vertex X from the stack
    process X
    for each neighbor Y of X
        if (Y.tag == false)
            push Y onto the stack
            Y.tag = true
What is the output of DFS traversal if we start at A and use the nonrecursive version?
Example

Output: A D C F B E
1. What is the traversal order of this graph using DFS-recursive version? The start point is 6.

A. 6, 4, 3, 2, 1, 5
B. 6, 4, 5, 2, 3, 1
C. 6, 4, 5, 1, 2, 3
D. 6, 4, 3, 2, 5, 1
Clicker Quiz

1. What is the traversal order of this graph using DFS-recursive version? The start point is 6.

A. 6, 4, 3, 2, 1, 5
B. 6, 4, 5, 2, 3, 1
C. 6, 4, 5, 1, 2, 3
D. 6, 4, 3, 2, 5, 1
2. What is the traversal order of this graph using DFS-nonrecursive version? The start point is 6.

A. 6, 4, 3, 2, 5, 1
B. 6, 4, 5, 1, 2, 3
C. 6, 4, 5, 2, 1, 3
D. 6, 4, 3, 2, 1, 5
2. What is the traversal order of this graph using DFS-nonrecursive version? The start point is 6.

A. 6, 4, 3, 2, 5, 1
B. 6, 4, 5, 1, 2, 3
C. 6, 4, 5, 2, 1, 3
D. 6, 4, 3, 2, 1, 5
Clicker Quiz

3. What is the traversal order of this graph using DFS-recursive version? The start point is 1.

A. 1, 3, 5, 4, 2
B. 1, 2, 4, 5, 3
C. 1, 2, 5, 4, 3
D. 1, 2, 3, 4, 5
3. What is the traversal order of this graph using DFS-recursive version? The start point is 1.

A. 1, 3, 5, 4, 2
B. 1, 2, 4, 5, 3
C. 1, 2, 5, 4, 3
D. 1, 2, 3, 4, 5
Clicker Quiz

4. What is the traversal order of this graph using DFS-nonrecursive version? The start point is 1.

A. 1, 2, 3, 4, 5
B. 1, 2, 5, 4, 3
C. 1, 2, 4, 5, 3
D. 1, 4, 3, 5, 2
Clicker Quiz

4. What is the traversal order of this graph using DFS-nonrecursive version? The start point is 1.

A. 1, 2, 3, 4, 5
B. 1, 2, 5, 4, 3
C. 1, 2, 4, 5, 3
D. 1, 4, 3, 5, 2
Big-O Notation Examples

```c
void foo(int n) {
    int i,j,k;
    for(i = 1; i <= n; i++)
        for(j = 1; j <= i; j++)
            for(k=1; k <= 100; k++) {
                print("good");
            }
}}
```

```
i = 1           i = 2               i = 3           i = n
j = 1           j = 2               j = 3 =>     j = n
k = 100         k = 200             k = 300        k = n * 100
```

```
total = 100 + 200 + 300 + 400 + 500 = 100 (1+2+3+..+n) = 100( n(n-1)/2) = O(n^2)
```
void foo(int n) {
    int i,j,k;
    for(i = 1; i <= n; i++)
        for(j = 1; j <= i*i; j++)
            for(k=1; k <= n/2; k++)
                print("good");
}

i = 1           i = 2              i = 3              i = n
j = 1           j = 4              j = 9    =>      j = n^2
k = n/2       k = n/2 * 4    k = n/2 * 9     k = n/2 * n ^2

total = n/2 + n/2 * 4 + n/2 * 9 + … + n/2 * n^2 = n/2 (1 + 4 + 9 + n^2)
= n/2 * (n(n+1)(2n + 1)/6) = O (n^4)
void foo()
{
    int i,j,k;
    for(i= n/2, i <= n; i++) //n/2 times
        for(j = 1; j <= n/2, j++) //n/2 times
            for(k=1; k <=n; k=k*2) // log_2^n times
                print("yes");

    O(n/2 * n/2 * log_2 n) = O(n^2 log_2 n)
Clicker Quiz

5. What is the Big-O of the following code?

A. O (nlog₂ n)

B. O (n)

C. O (log₂ n)

D. O (log₂ n²)

```c
void foo(int n)
{
    for(int i = n; i > 0; i = i / 2)
        print("good");
}
```
Clicker Quiz

5. What is the Big-O of the following code?

A. O (nlog_2 n)
B. O (n)
C. O (log_2 n)
D. O (log_2 n^2)

```c
void foo(int n)
{
    for(int i = n; i > 0; i = i / 2)
        print("good");
}
```
6. What is the Big-O of the following code?

A. O(n)
B. O(log₂ n)
C. O(n² log₂ n)
D. O(log₂ n²)

```c
void foo()
{
    int i, j, k;
    for(i = n/2; i <= n; i++)
        for(j = 1; j <= n/2, j++)
            for(k = 1; k <= n; k = k * 2)
                print("yes");
}
```
6. What is the Big-O of the following code?

A. O (n)

B. O (log_2 n)

C. O (n^2 log_2 n)

D. O (log_2 n^2)

```c
void foo()
{
    int i,j,k;
    for(i= n/2, i <= n; i++)
        for(j = 1; j <= n/2, j++)
            for(k=1; k <=n; k=k*2)
                print("yes");
}
```
Clicker Quiz

7. What is the Big-O of the following code?

A. $O \left( n^2 \log_2 n \right)$
B. $O \left( n \left( \log_2 n \right)^2 \right)$
C. $O \left( n \right)$
D. $O \left( \log_2 n^2 \right)$

```c
void foo()
{
    int i,j,k;
    for(i = n/2; i <=n; i++)
        for(j = i; j <=n;j=j*2)
            for(k=1; k <=n; k=k*2)
                print("good");
}
```
Clicker Quiz

7. What is the Big-O of the following code?

A. $O(n^2 \log_2 n)$
B. $O(n (\log_2 n)^2)$
C. $O(n)$
D. $O(\log_2 n^2)$

```c
void foo()
{
    int i,j,k;
    for(i = n/2; i <=n; i++)
        for(j = i; j <=n;j=j*2)
            for(k=1; k <=n; k=k*2)
                print("good");
}
```
8. What is the Big-O of the following code?

A. $O(n)$
B. $O(n^2)$
C. $O(1)$
D. $O(\log_2 n)$

```cpp
void (int n)
{
    for(i = 1; i<=n; i++)
        for(j = 1; j <= i; j = j+i)
            print("good");
}
```
8. What is the Big-O of the following code?

A. O (n)
B. O (n^2)
C. O (1)
D. O (log_2 n)

```c
void (int n)
{
    for(i = 1; i<=n; i++)
    {
        for(j = 1; j <= i; j = j+i)
            print("good");
    }
}
```
Clicker Quiz

9. What is the Big-O of the following code?

A. O (log_2 n)
B. O (1)
C. O (n^2)
D. O (n)

```java
void foo(int n)
{
    for(int i = 1; i < n; i = i * 2)
        print("good");
}
```
9. What is the Big-O of the following code?

A. \(O(\log_2 n)\)

B. \(O(1)\)

C. \(O(n^2)\)

D. \(O(n)\)

```cpp
void foo(int n)
{
    for(int i = 1; i < n; i = i * 2)
        print("good");
}
```
10. What is the Big-O of the following code?

A. O(1)
B. O(log_2 n)
C. O(n)
D. O(n^2)

```
void (int n)
{
    for(i = 1; i<=n; i++)
        for(j = 1; j <= n; j++)
            print("good");
}
```
Clicker Quiz

10. What is the Big-O of the following code?

A. $O(1)$
B. $O(\log_2 n)$
C. $O(n)$
D. $O(n^2)$

```c
void (int n)
{
    for(i = 1; i<=n; i++)
        for(j = 1; j <= n; j++)
            print("good");
}
```