Analysis of Algorithms

V. Adamchik

CSCI 570

Lecture 1

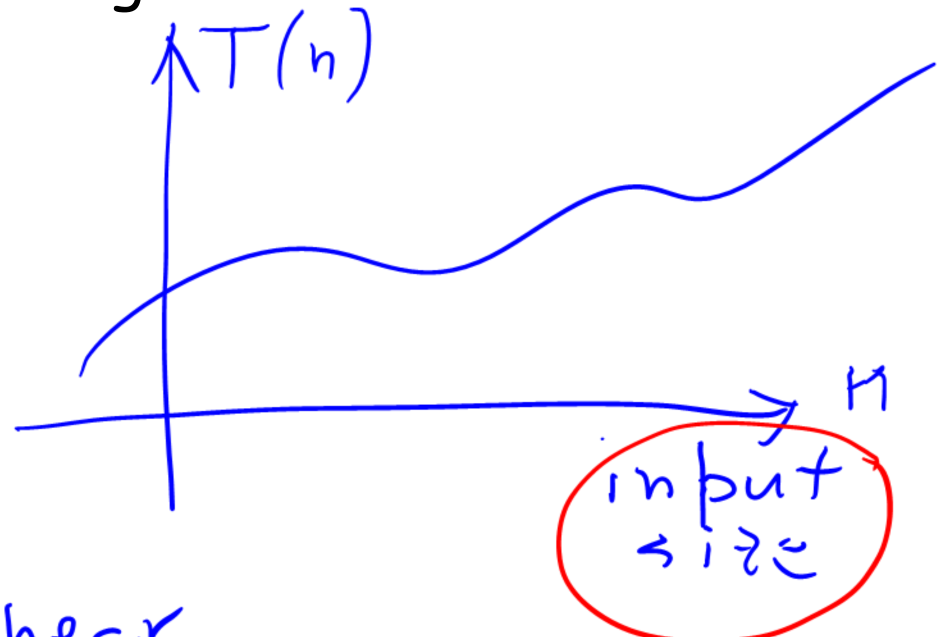
University of Southern California

Review

Reading: chapter 1

Chapter 1.1: Runtime Complexity

The term analysis of algorithms is used to describe approaches to study the performance of computer programs. We interested to find a runtime complexity of a particular algorithm as a function of T(*n*) that describes a relation between algorithm's execution time and the input size *n*.



$\boxed{\lim_{n \to \infty} T(n)}$

CPU: 64 bits

$2+2 \to O(1)$ const.

$O(n)$ linear

Runtime Complexity

In this course we will perform the following types of analysis:

- the worst case complexity upper bound O
- the best case complexity lower bound Ω
- the average case complexity $[L, U]$
- the amortized time complexity
  lecture 2

We measure the run time of an algorithm using following asymptotic notations: O, Ω, Θ.

Big-O (upper bound)

For any monotonic functions f, g from the positive integers to the positive integers, we say

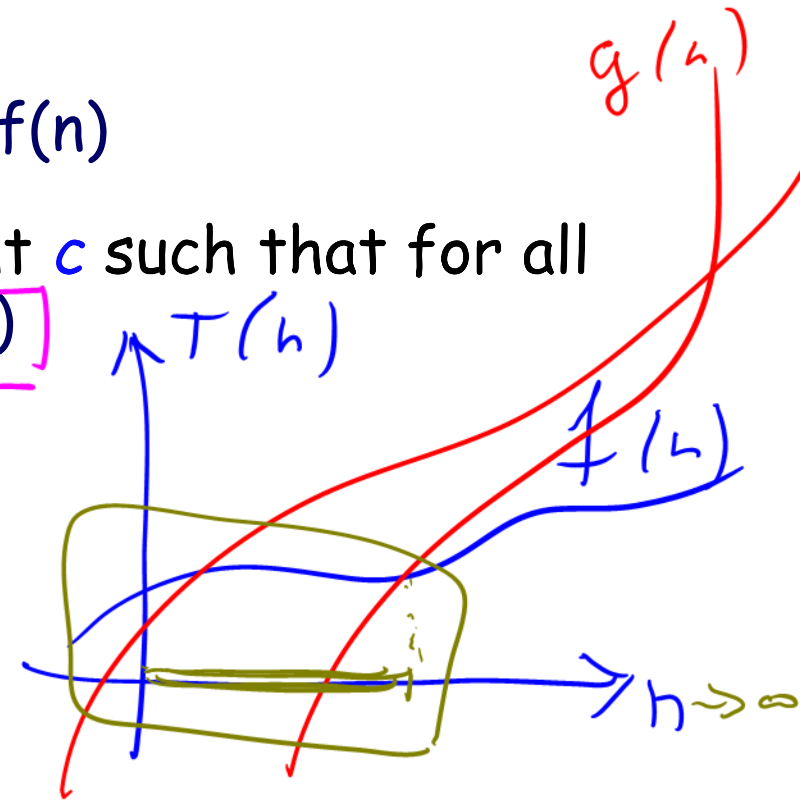
$$f(n) = O(\,g(n)\,)$$

if

g(n) eventually dominates f(n)

Formally: there exists a constant c such that for all *sufficiently large* n: $\boxed{f(n) \le c * g(n)}$

$O(1) \le O(\log n) \le O(n) \le O(n \log n) \le O(n^2) \le \dots \le O(n!)$



$\log_2$

Discussion Problem 1

Arrange the following functions in increasing order of growth rate with g(n) following f(n) in your list if and only if $f(n) = O(g(n))$, $n \to \infty$

$n^{1.4}$

5 7 8 4 1 3 2 6

$\log n^n$, n^2, $n^{\log n}$, $n \log \log n$, $n^{1/\log n}$ $2^{\log n}$, $\log^2 n$, $n^{\sqrt{2}}$

$= n \cdot \log n$ $= 2$ $= n$

$O(1)$

$$\boxed{\log_b a = \frac{\log a}{\log b}}$$

$$\frac{1}{\log n} = \frac{\log_2 2}{\log_2 n} = \log_n 2$$

Discussion Problem 2

Suppose that $f(n)$ and $g(n)$ are two positive non-decreasing functions such that $f(n) = O(g(n))$.

Is it true that $2^{f(n)} = O(2^{g(n)})$? FALSE

if it's true, then prove it
if it's false, then provide an example

$f(n) = 2n$, $g(n) = n$, $2n = O(n)$

$2^{f(n)} = 4^n$, $2^{g(n)} = 2^n$

$\frac{4^n}{2^n} = 2^n$

$\boxed{4^n \neq O(2^n)}$ is it true? NO

Omega: Ω (lower bound)

For any monotonic functions f, g from the positive integers to the positive integers, we say

$$f(n) = \Omega(\ g(n)\ )$$

if:

f(n) eventually dominates g(n)

Formally: there exists a constant c such that for all *sufficiently large* n: $\boxed{f(n) \geq c*g(n)}$ Definition

$$4^n = \Omega(2^n) = \Omega(n)$$

Discussion Problem 3

Suppose that $f(n)$ and $g(n)$ are two positive non-decreasing functions such that $f(n) = \Omega(g(n))$.

Is it true that $2^{f(n)} = \Omega(2^{g(n)})$?

$f(n) = n, \quad g(n) = 2n, \quad n = \Omega(2n)$

$2^{f(n)} = 2^n$

$2^{g(n)} = 4^n$

$$\boxed{\begin{array}{c} 2^n \neq \Omega(4^n) \\ \text{is it true?} \end{array}}$$

FALSE

Theta: Θ

For any monotonic functions f, g from the positive integers to the positive integers, we say

$$f(n) = \Theta(\ g(n)\ )$$ $c_2 g(n) \le f(n) \le c_1 g(n)$

if:

$$f(n) = O(\ g(n)\ ) \quad \underline{\text{and}} \quad f(n) = \Omega(\ g(n)\ )$$

In this class we will be mostly concerned with a big-O notation.

$T(n) \in [L, U]$

$L = U$

Quickies

1. $n \overset{\geq}{=} \Omega(n^2)$? F
2. $n = \Theta(n + \cancel{\boxed{\log n}})$? T
3. $\log n \overset{\geq}{=} \Omega(n)$? F
4. $n^2 = \Omega(n \log n)$? T
5. $n^2 \log n = \Theta(n^2)$? F
6. $3n^2 + \cancel{4n} + \cancel{5} = \Theta(n^2)$? T
7. $2^n + 100n^2 + n^{100} = \Omega(n^{101})$? T
8. $\underset{\to 0}{(1/3)^n} + 100 = \Theta(1)$? T

$\lim_{n \to \infty}$

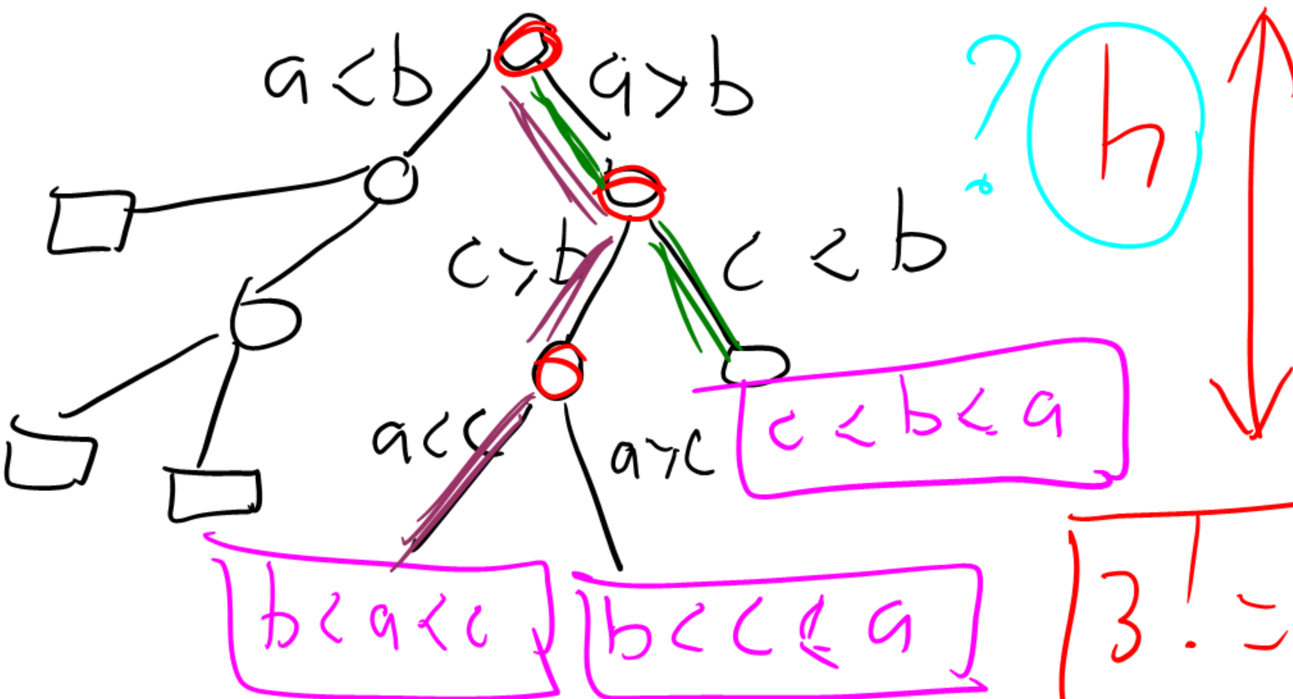
$\lim_{n \to \infty}$

$\Omega(1) \neq \Omega(n)$

Chapter 1.2: Sorting Lower Bound

We will show here that any deterministic comparison-based sorting algorithm must take $\Omega(n \log n)$ time to sort an array of n elements in the worst-case.

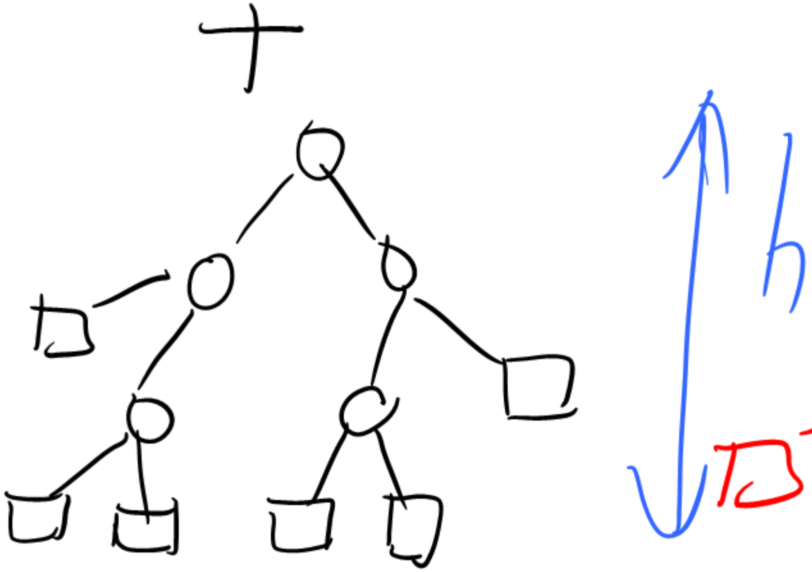
|a/b/c| $a \neq b \neq c$ Decision tree

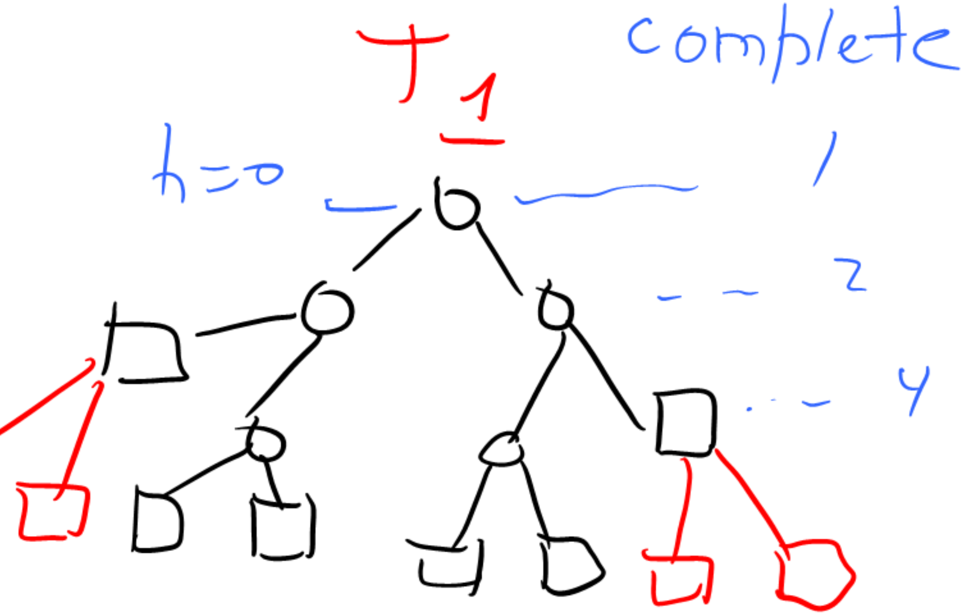


$h \overset{?}{=} f(n)$

h is runtime complexity

$3! = 6$





leaves: $n!$

leaves: 2^h

$n! \le 2^h$ ⟸ why?

Apply $\log_2$

$h \ge \log(n!)$?

$$\boxed{\lim_{n \to \infty} T(n)}$$

$$h \geq \log(n!) = \log(n \cdot (n-1)(n-2) \ldots 2 \cdot 1) \geq$$

$$\geq \log(n\,(n-1)(n-2) \cdots \tfrac{n}{2}) \geq$$

$$\geq \log\left(\frac{n}{2} \cdot \frac{n}{2} \cdot \frac{n}{2} \cdot \ldots \cdot \frac{n}{2}\right) =$$

$$= \log\left(\frac{n}{2}\right)^{n/2} = \frac{n}{2} \cdot \log\left(\frac{n}{2}\right)$$

Definition of Ω

$$h = \Omega(n \cdot \log n)$$

Discussion Problem 4

What is the Big-O runtime complexity of the following function? Give the tightest bound.

$\text{bigOh1}(n) = O(n \log n)$

```
void bigOh1(int n):
  for i=1 to n
     j=1;
     while j < i
          j = j*2;
```

n

log n

Discussion Problem 5

$$\sum x^k = \frac{1}{1-x}$$

What is the Big-O runtime complexity of the following function? Give the tightest bound.

$x = \frac{1}{4}$

upper bound

```
void bigOh2(int[] L, int n)
    while (n > 0)
        find_max(L, n); //finds the max in L[0…n-1]
        n = n/4;
```

$O(n)$ $O(n \cdot \log n)$

$\log_4 n$

Theta Θ - ??

step by step analysis:

geometric series

$$n + \frac{n}{4} + \frac{n}{16} + \ldots + 1 = n\left(1 + \frac{1}{4} + \frac{1}{16} + \ldots + \frac{1}{n}\right) \le$$

$$\le n\left(1 + \frac{1}{4} + \frac{1}{16} + \ldots\right) = n \sum_{k=0}^{\infty} \frac{1}{4^k} = \Theta(n)$$

Discussion Problem 6

What is the Big-O runtime complexity of the following function? Give the tightest bound.

???

```
string bigOh3(int n)
  if(n == 0) return "a";
  string str = bigOh3(n-1);
  return str + str;
```

$\lim_{n \to \infty}$ bigOh3(n)

$O(n)$

No

~~$O(n^2)$~~

complexity?

linear in the input size.

???

sizeof str in n?

bigoh3(0) = a

bigoh3(1) = aa

bigoh3(2) = aaaa

bigoh(3) = aaaaaaaa

2^n

upper bound

$O(n \cdot 2^n)$

$1+2+4+\dots+2^n \approx 2^{n+1}$

$\approx \Theta(2^n)$

Chapter 1.3: Trees and Graphs

A graph G is a pair (V, E) where V is a set of vertices (or nodes) E is a set of edges connecting the vertices.

An undirected graph is connected when there is a path between every pair of vertices.

A tree is a connected graph with no cycles.

A path in a graph is a sequence of distinct vertices.

A cycle is a path that starts and ends at the same vertex.

We start with reviewing mathematical proofs (induction and contradiction).

Theorem. Let G be a graph with V vertices and E edges. The following statements are equivalent:

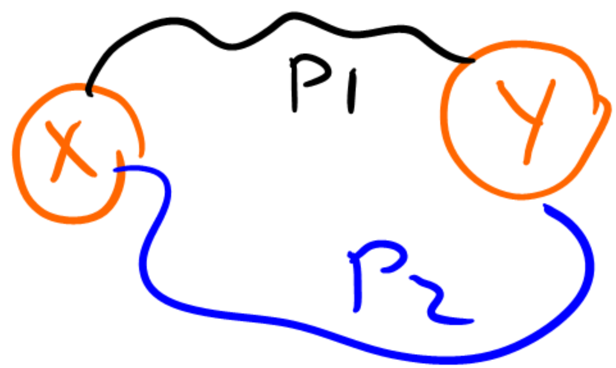
1. G is a tree (a connected graph with no cycles).
2. Every two vertices of G are connected by a unique path.
3. G is connected and $\boxed{V = E + 1}$. ??
4. G is acyclic and $V = E + 1$.
5. G is acyclic and if any two non-adjacent vertices are joined by an edge, the resulting graph has exactly one cycle.

1→2: Given 1, prove 2

Prove that a path is unique.

Proof by contradiction.

Assume that a path is NOT unique.



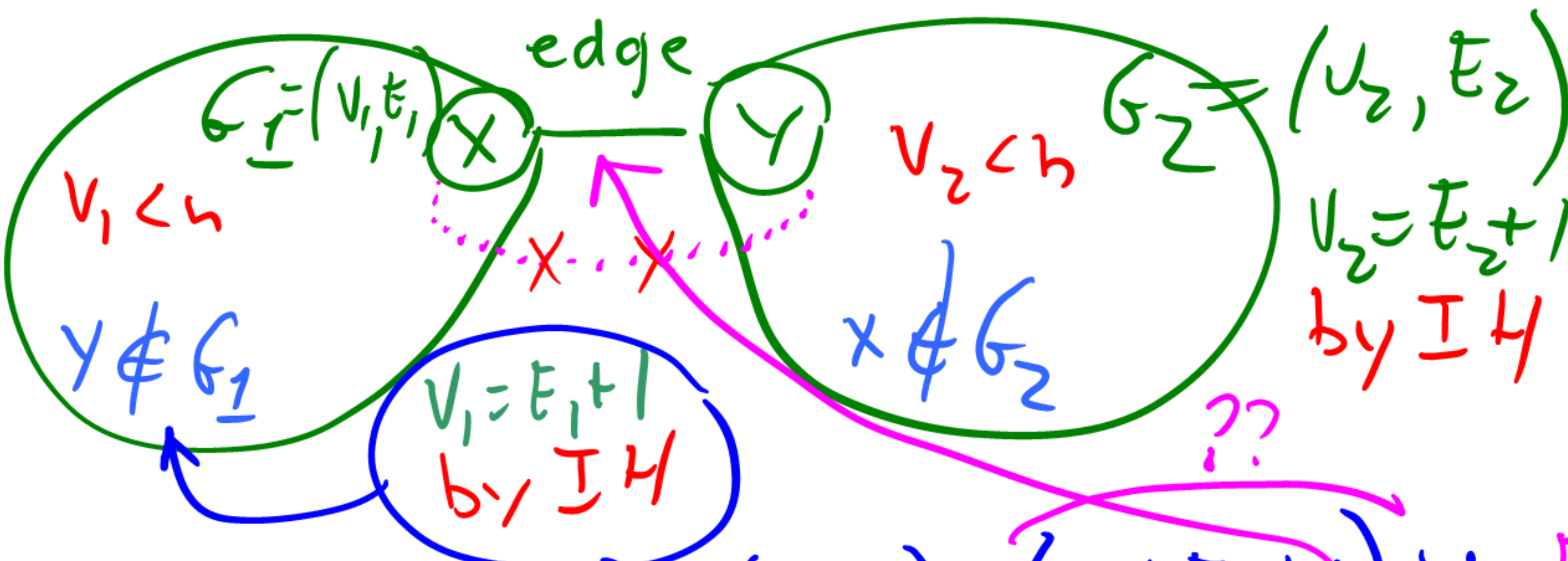
$P_1 + P_2$ is a cycle.

Contradiction!

2→3: Given 2, Prove $V=E+1$

Proof by induction on vertices

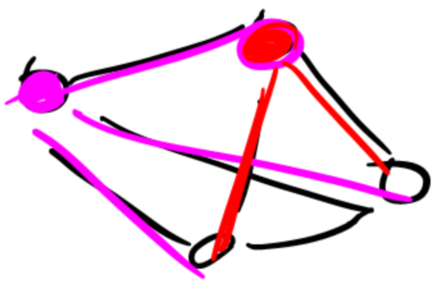
1. Base case. $V=2$  $V=E+1$
2. IH: Assume $V=E+1$ for graphs $V<n$.
3. IS: Prove $V=E+1$ for graphs $V=n$.

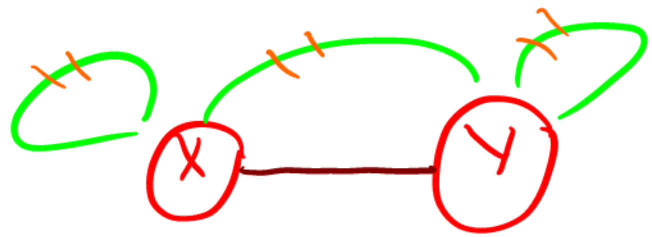


$$V = V_1 + V_2 = (E_1+1)+(E_2+1) = (E_1+E_2+1)+1 = E+1$$

Theorem. Prove that in an undirected simple graph $G = (V, E)$, there are at most $V(V-1)/2$ edges. In short, using the asymptotic notation, $E = O(V^2)$.

Proof.





at most one edge

$$E = (V-1) + (V-2) + (V-3) + \dots + 1 = \frac{V(V-1)}{2} = O(V^2)$$

Sparse: $E \sim V$

Dense: $E \sim V^2$

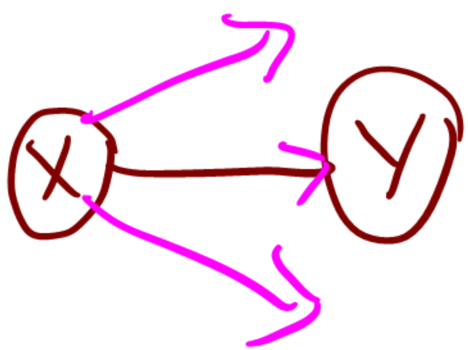
Representing Graphs

Adjacency List sparse

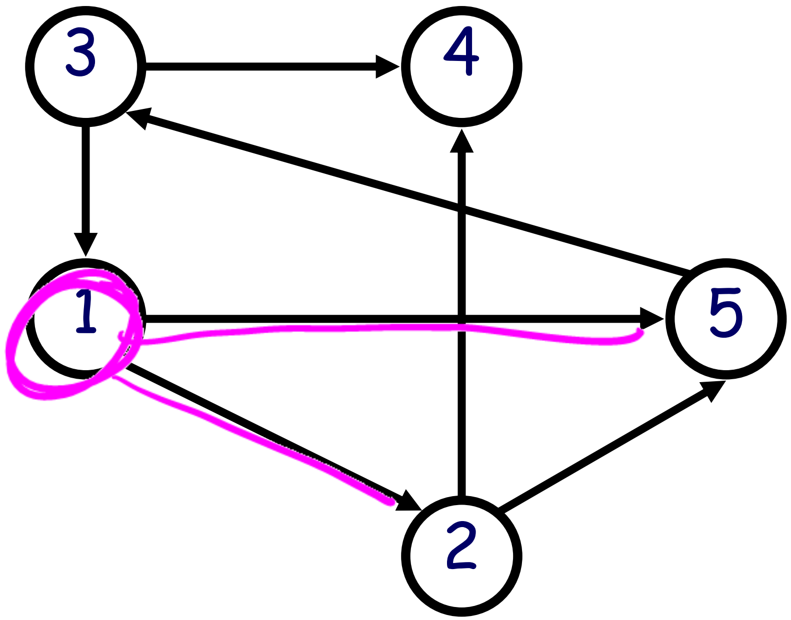
or

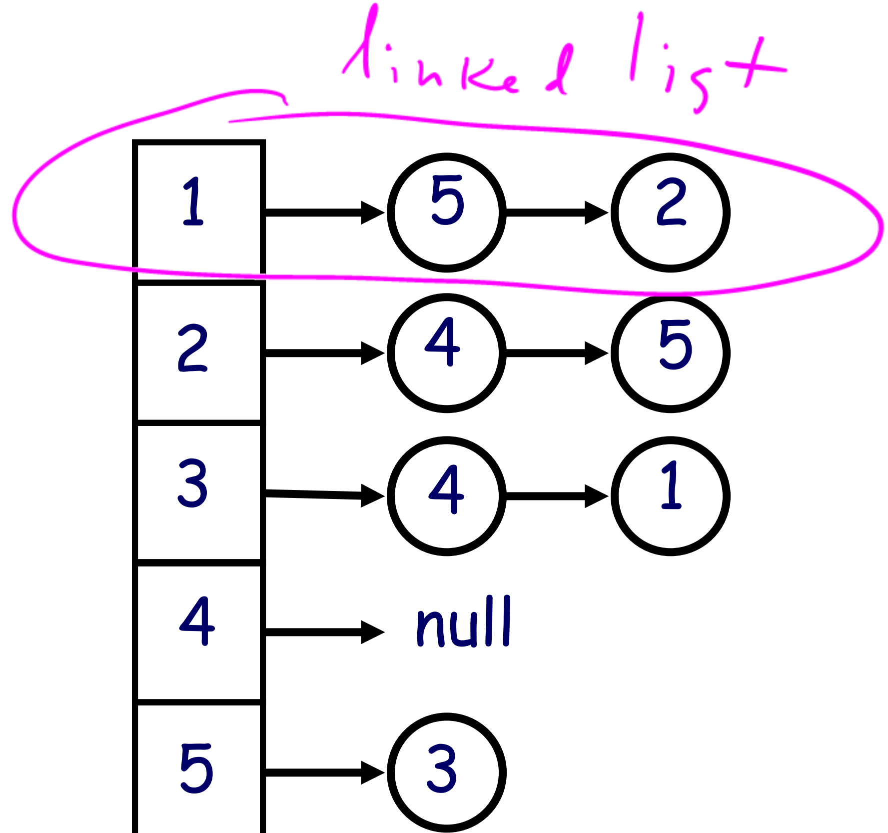
Adjacency Matrix dense

Vertex X is *adjacent* to vertex Y if and only if there is an edge (X, Y) between them.



Adjacency List Representation

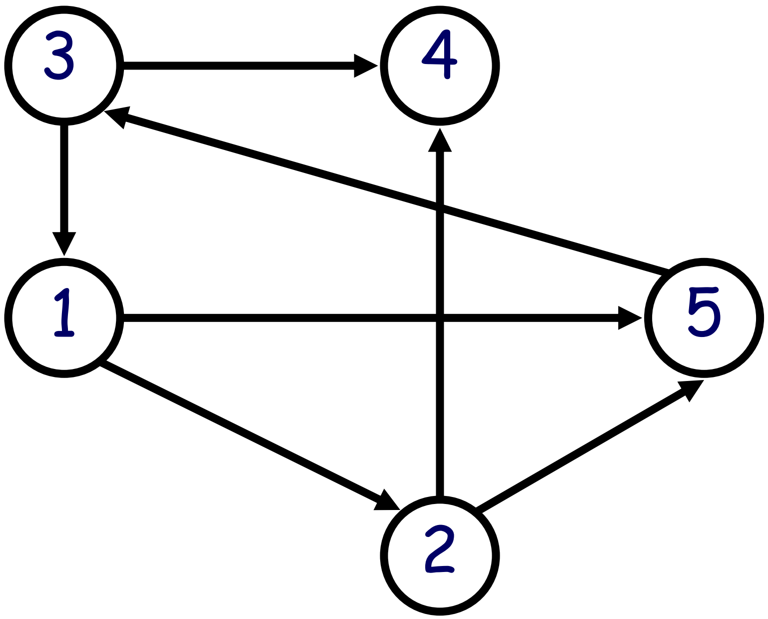




Is vertex 1 adjacent to 3?

It takes linear time to figure it out.

Adjacency Matrix Representation



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Is vertex 1 adjacent to 3?

It takes constant time to figure it out.

Representing Graphs

Adjacency List Representation is used for representation of the sparse ($E = O(V)$) graphs.

Adjacency Matrix Representation is used for representation of the dense ($E = \Omega(V^2)$) graphs.

Is the Facebook social graph sparse or dense?

sparse

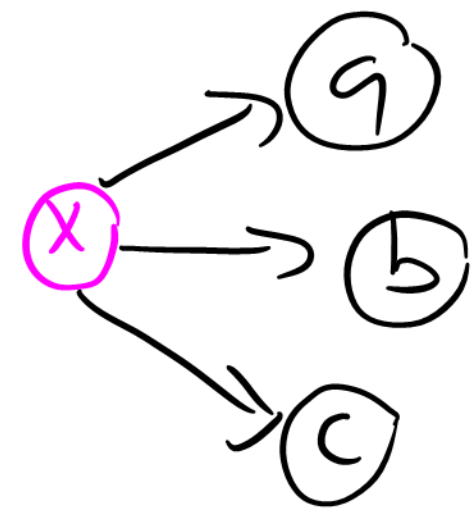
We can say a connected graph is maximally sparse if it is a tree.

We can say a graph is maximally dense if it is complete.

Graph Traversals

visit All vertices

Depth-First-Search (DFS)
Breadth-First-Search (BFS)

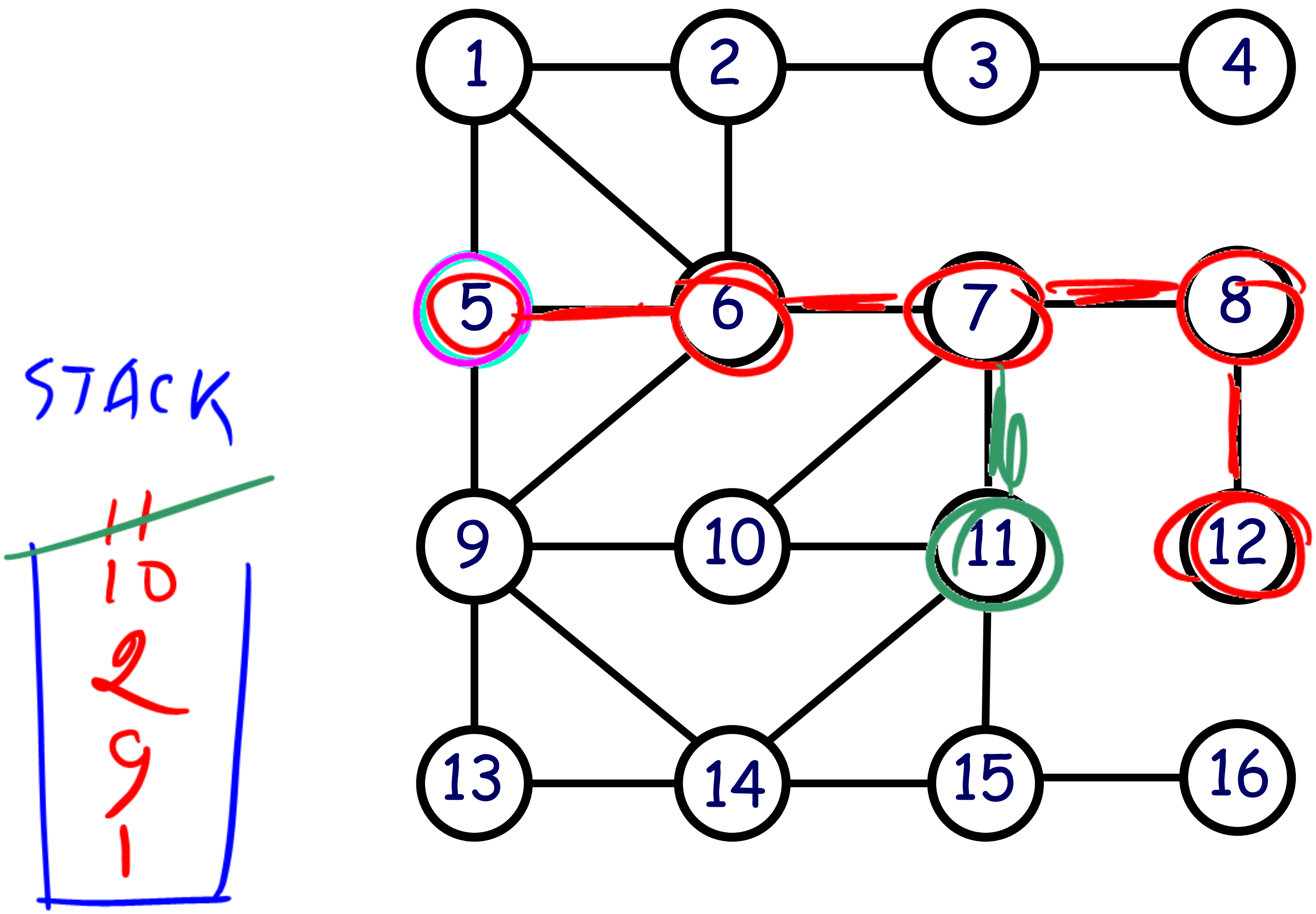


DFS uses a stack for backtracking.
BFS uses a queue for bookkeeping.

Runtime complexity: O(V + E)

Result: spanning tree.

Perform a DFS on the following graph



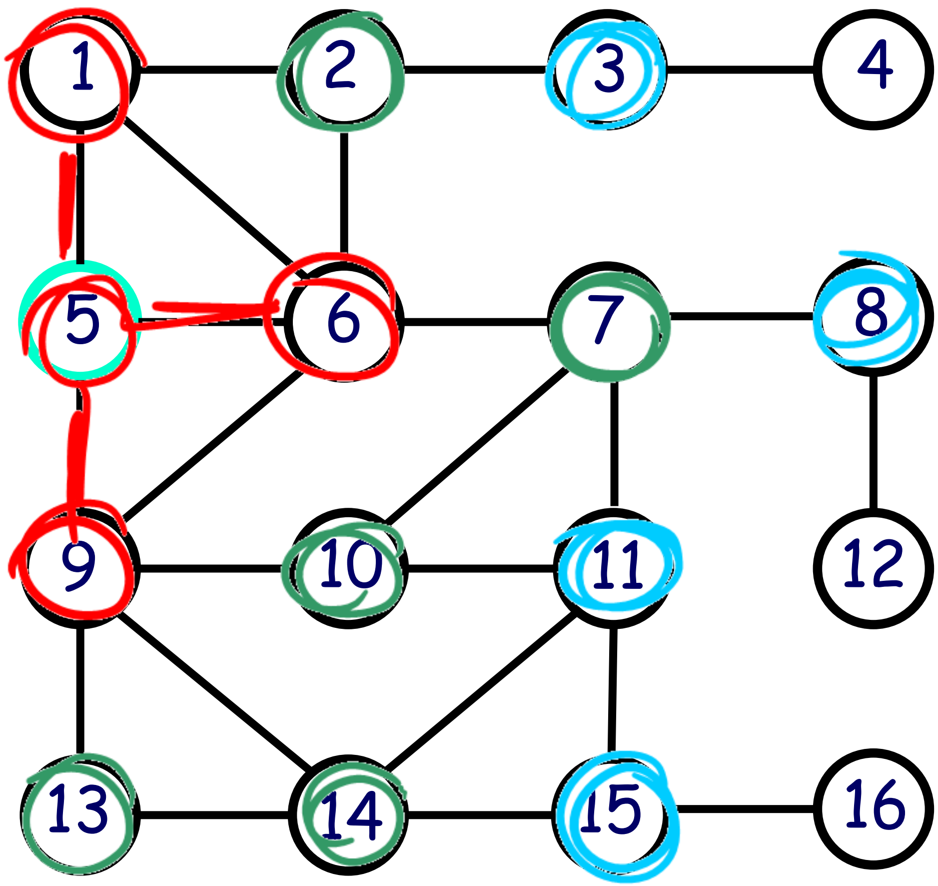
Perform a BFS on the following graph

level order

5

1, 6, 9

2, 7, 10, 13, 14

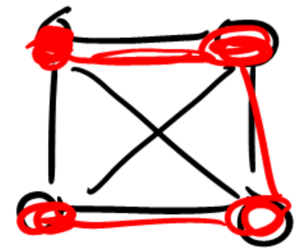


Discussion Problem 7

The complete graph on n vertices, denoted K_n, is a simple graph in which there is an edge between every pair of distinct vertices.

$h = n-1$

1. What is the height of the DFS tree for the complete graph K_n?

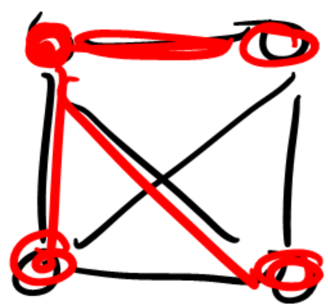


K_4

$h = 3$

$h = 1$

2. What is the height of the BFS tree for the complete graph K_n?



K_4

$h = 1$