GRAPH SEARCHES

Marcus Ritt

CMP 601 – Algorithms and Theory of Computation — April 2020

Outline

1. Review
2. Depth-first search
3. Implementation aspects
4. Bipartiteness

- We're exploring an *undirected* graph $G = (V, E)$.

vertices $n = |V|$

edges $m = |E|$

Breadth-first search

$BFS(s) :=$

$T := \emptyset$

$L_0 := \{s\}$

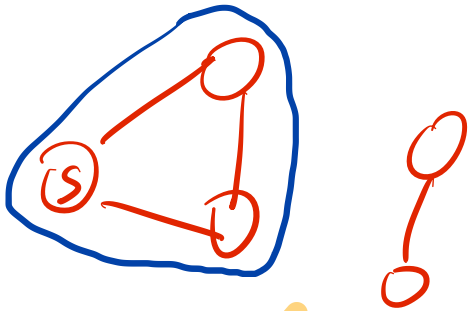
while $L_i \neq \emptyset$

$L_{i+1} := \{v \in V \mid \{u, v\} \in A, u \in L_i, v \text{ unseen}\}$

$T := T \cup \{\{u, v\} \mid \{u, v\} \in A, u \in L_i, v \text{ unseen}\}$

$i := i + 1$

end while

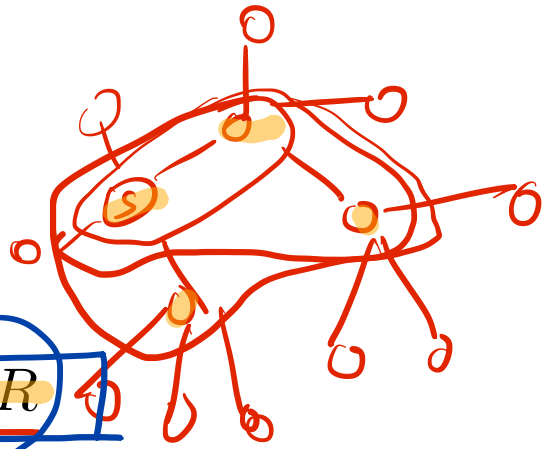


Observations:

- L_i contains nodes at distance i from s. An s-t path exists, if $t \in L_i$ for some i.
- BFS defines a BFS tree T.
- Proposition: If $x, y \in T$ and $x \in L_i, y \in L_j$: $|i - j| \leq 1$.

Random search

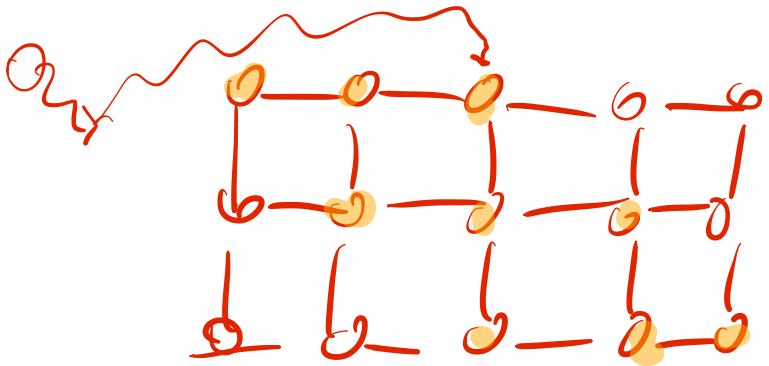
$RS(s) :=$
$R := \{s\}$
while $\exists \{u, v\} \in A : u \in R, v \notin R$
$R := R \cup \{v\}$
end while

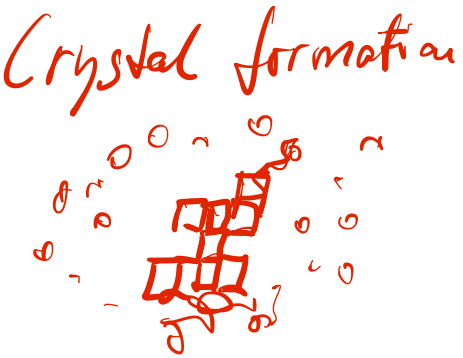


Implementation?

Observations:

- Proposition: R is the connect component of s.



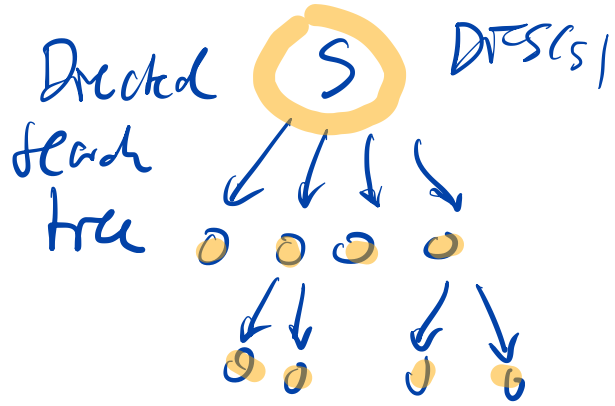


Depth-first search

$v \in V$

$DFS(v) :=$
  $visited(v) := true$
  $for\ u \in N(v) | visited(u) = false\ do$
    $DFS(u)$
    $\| T := T \cup \{\{v, u\}\}$

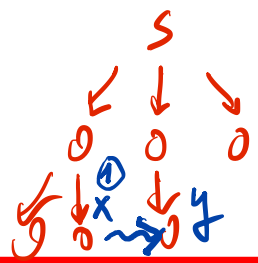
$T \leftarrow$ undirected edge

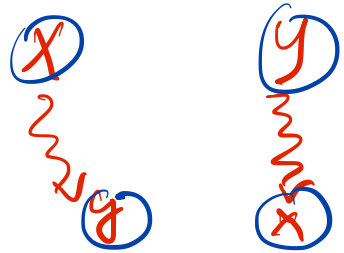


$T := T \cup \{(u, v)\}$ directed arc/tree

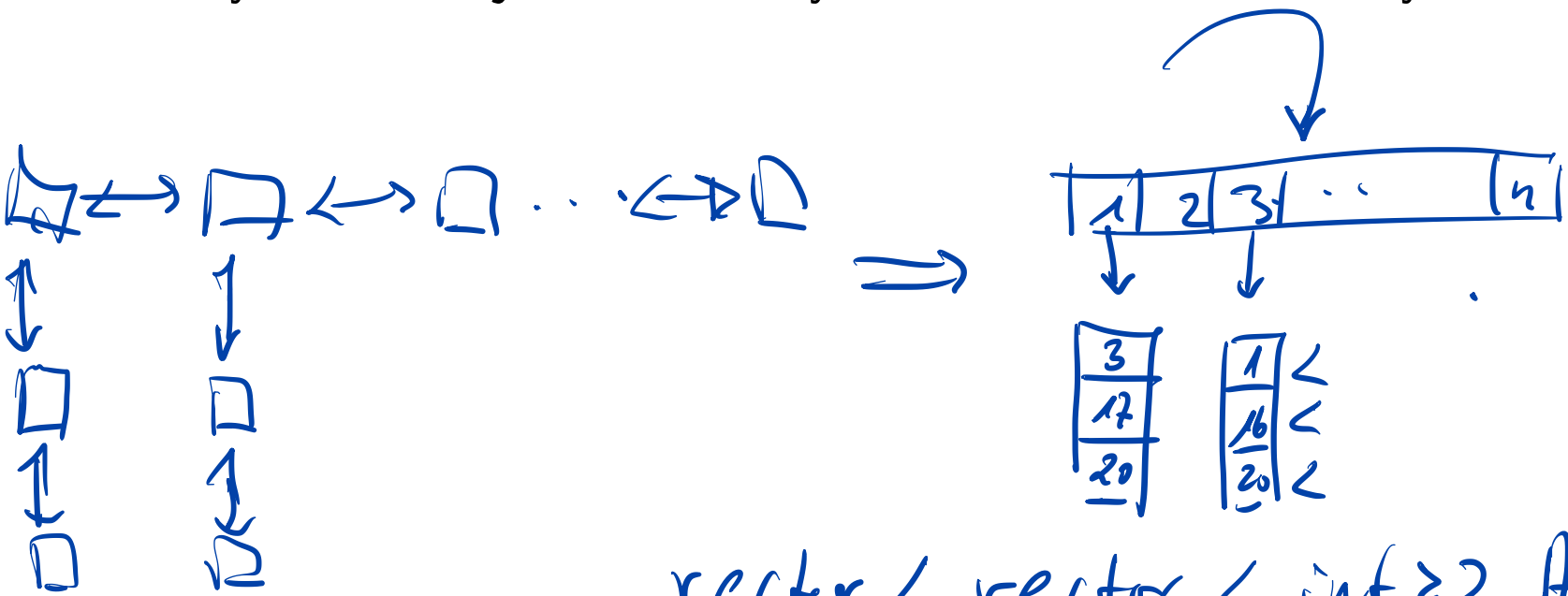
Observations:

- Again DFS(s) defines a tree T: every vertex except s has exactly one incoming arc, and T is connected, by construction.
- All explored vertices during DFS(s) are descendants of s.
- Proposition: If $\{x, y\} \in E$ and $\{x, y\} \notin T$ then x is ancestor of y or vice versa.

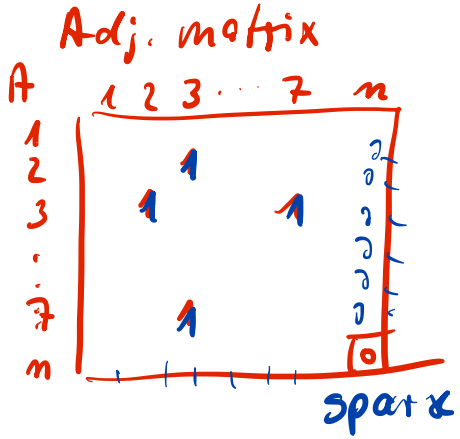




- Why don't we just visit every vertex, in some arbitrary order?



vector < vector < int >> A;



$\{3,2\} \in E$

$A[2,3] = A[3,2] = 1$

$\{3,7\} \in E$

$A[3,7] = A[7,3] = 1$

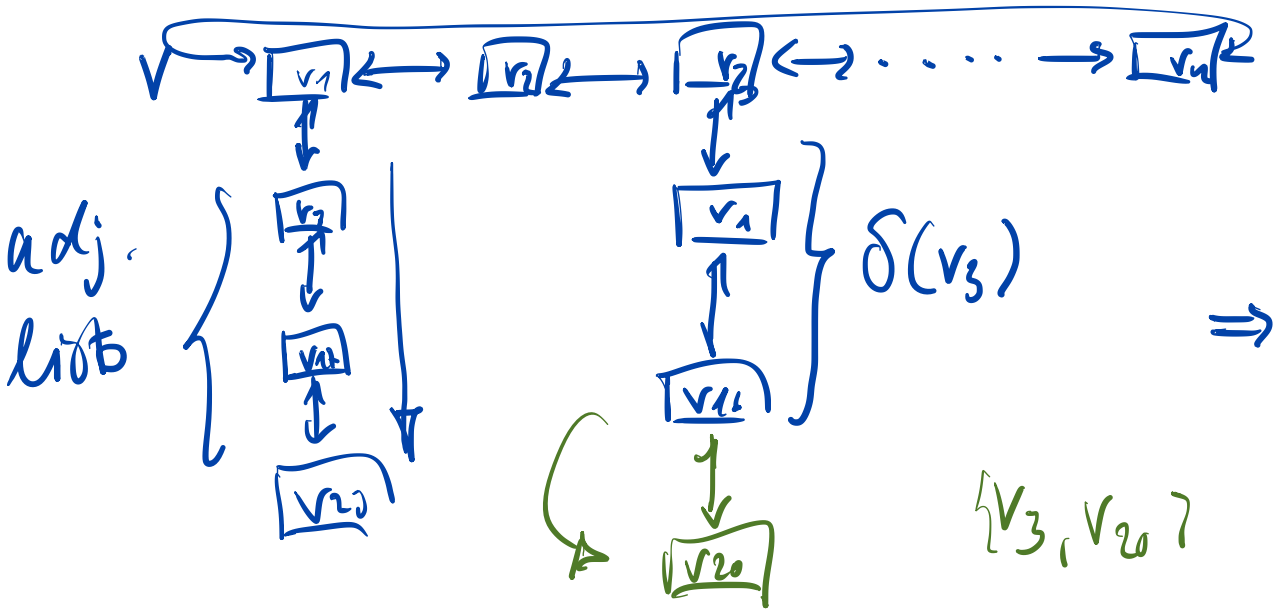
$\{u,v\} \in E$? $A[u,v] = 1$?

$O(\delta(v))$ $O(1)$

for $u \in N(v)$: • for $u = 1:n$ if $A[u,v]$:

$\Theta(n^2)$ $O(n)$ • $O(\delta(v))$

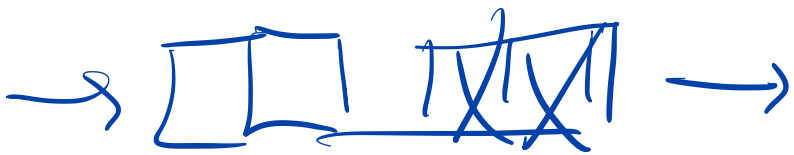
- Adjacency matrices
- Adjacency lists or vectors, forward or backward stars

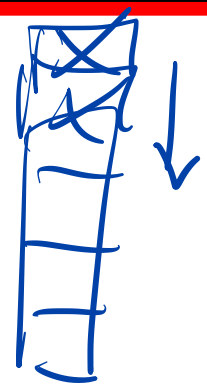


$\sum_{v \in V} \delta(v) = 2m$

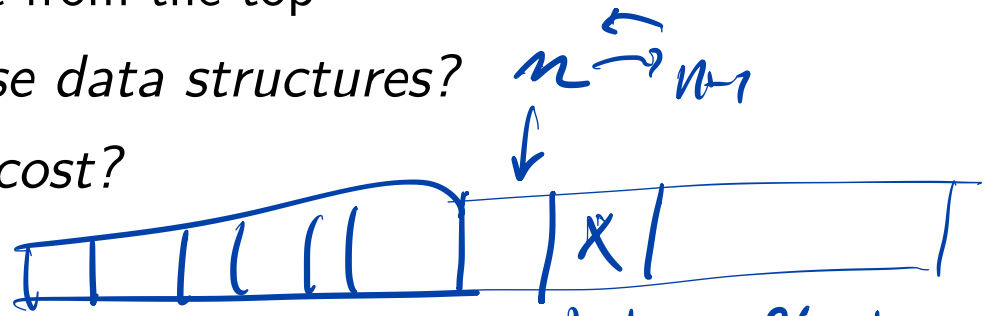
$\Rightarrow$ space (linear)

$O(n+m)$





- A *queue*: processes items in order (FIFO)
  - Enqueue (push): add an element at one end
  - Dequeue (pop): remove an element from the other end
- A *stack*: processes items in reverse order of addition (LIFO)
  - Push: add an element to the top
  - Pop: remove an element from the top
- *How do we implement these data structures?*
- *What do these operations cost?*



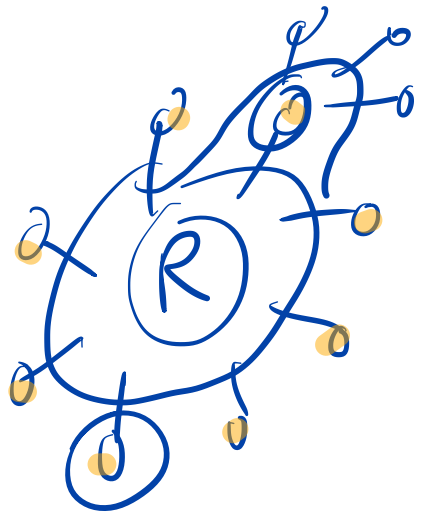
Stack: vector

Queue:

push/pop O(1)

linear time
→ O(n+m)

- BFS: maintain vertices in a queue
- DFS: maintain vertices in a stack (non-recursive)
- How do we implement a *random search*?
- What does it cost?



fringe vertex: unexplored vertex connected to a already explored vertex

fringe vertices

Queue or stack or vector of fringe vertices

A undirected graph $G = (V, E)$ such that

- Vertices have two parts: $V = L \cup R$, $L \cap R = \emptyset$
- Edges only between parts: $E \subseteq \{\{u, v\} \mid u \in L, v \in R\}$

The following is equivalent:

1. G is bipartite
2. G has no odd cycles
3. G has no edges connecting vertices in the same layer of the BFS tree.