CSE 101- Winter ‘18
Discussion Section
Week 2
Topics

- Topological ordering
- Strongly connected components
- Binary search
- Introduction to Divide and Conquer algorithms
Topological ordering

- Given a directed graph $G = (V,E)$ with $|V|=n$, assign labels 1,...,$n$ to $v_i \in V$ s.t. if $v$ has label $k$, all vertices reachable from $v$ have labels $> k$
Topological ordering

• Given a directed graph $G = (V,E)$ with $|V|=n$, assign labels $1,...,n$ to $v_i \in V$ s.t. if $v$ has label $k$, all vertices reachable from $v$ have labels $> k$

• Pictorially

(only forward edges if vertices arranged in increasing order of labels)
Topological ordering

• If G has a directed cycle => no topological ordering
  – Why?

• **Theorem** – Every directed graph without a directed cycle (DAG) has a topological ordering

• Revised problem: Given a directed **acyclic** graph G = (V,E) with |V|=n, assign labels 1,...,n to \( v_i \in V \) s.t. if v has label k, all vertices reachable from v have labels > k
Topological ordering – inductive approach

• How would you approach this problem?
Topological ordering – inductive approach

• How would you approach this problem?
  – Find a vertex which you know can be labelled 1
  – What are the properties of such a “starting” vertex?
Topological ordering – inductive approach

• How would you approach this problem?
  – Find a vertex which you know can be labelled 1
  – What are the properties of such a “starting” vertex?

• Claim: A DAG G always has some vertex with indegree = 0
  – Take an arbitrary vertex v. If v doesn’t have indegree = 0, traverse
    any incoming edge to reach a predecessor of v. If this vertex doesn’t
    have indegree = 0, traverse any incoming edge to reach a
    predecessor, etc.
  – Eventually, this process will either Identify a vertex with indegree = 0, or else reach a vertex that has been reached previously (a contradiction, given that G is acyclic)
Topological ordering – inductive approach

• How would you approach this problem?
  – Find a vertex which you know can be labelled 1
  – What are the properties of such a “starting” vertex?

• Inductive (or recursive) approach
  – Find a vertex $v$ with $\text{indegree}(v) = 0$; give it lowest available label;
  – Delete $v$ (and incident edges, update degrees of remaining edges)
  – Repeat

• (Bonus) – can you do the same, beginning with an “ending” vertex?
Topological ordering – using DFS

• Claim: In a DAG, every edge leads to a vertex with lower post number.
• Why?
Topological ordering – using DFS

- Claim: In a DAG, every edge leads to a vertex with lower post number.
- Why?
  - Any edge \((u,v)\) for which \(\text{post}(v) > \text{post}(u)\) is a back edge. But a DAG, being acyclic, has no back edges.
Topological ordering – using DFS

- Claim: In a DAG, every edge leads to a vertex with lower post number.
- Why?
  - Any edge \((u,v)\) for which \(\text{post}(v) > \text{post}(u)\) is a back edge. But a DAG, being acyclic, has no back edges.

- Obvious solution:
  - Run DFS, then perform tasks in decreasing order of post numbers
  - Linear running time!
Topological ordering – example

A → B → C
Topological ordering – example

A → B → C
Topological ordering – example

1, 2, 3
Topological ordering – example

A → B → C

1, 2, 3 → 4, 5
Topological ordering – example

A -> B -> C

1,6
2,3
4,5
Topological ordering – example

Vertices explored in alphabetical order

Topological order: A-C-B
Topological ordering – example

Vertices explored in order: B-C-A

Vertices explored in alphabetical order

Topological order: A-C-B
Topological ordering – example

Topological order: A-C-B

Vertices explored in alphabetical order

Vertices explored in order: B-C-A
Topological ordering – example

Vertices explored in alphabetical order

Vertices explored in order: B-C-A
Topological ordering – example

Vertices explored in order: B-C-A

Topological order: A-C-B
Topological ordering – example

Vertices explored in alphabetical order

Topological order: A-C-B

Vertices explored in order: B-C-A
Topological ordering – example

Vertices explored in alphabetical order

Vertices explored in order: B-C-A

Topological order: A-C-B
Topological ordering – example

Vertices explored in alphabetical order:

Topological order: A-C-B

Vertices explored in order: B-C-A

Topological order: A-C-B
Strongly Connected Components

- Two vertices $u$ and $v$ of a directed graph are connected if there is a path from $u$ to $v$ and a path from $v$ to $u$. 
Strongly Connected Components

- Two vertices $u$ and $v$ of a directed graph are connected if there is a path from $u$ to $v$ and a path from $v$ to $u$.
- This definition leads to a partition of a directed graph into disjoint sets of vertices -> strongly connected components
Meta graph

• Meta Graph – The graph obtained by shrinking each strongly connected component down to a single meta-node and drawing an edge from one meta-node to another if there is an edge between their respective components.

• Meta Graph is a DAG. (Why?)

• In other words, every directed graph is a dag of its strongly connected components.

![Diagram of Meta Graph]

(A) (B) (C) (D) (E) (F)
Meta graph

- Meta Graph – The graph obtained by shrinking each strongly connected component down to a single meta-node and drawing an edge from one meta-node to another if there is an edge between their respective components.
- Meta Graph is a DAG. (Why?)
- In other words, every directed graph is a dag of its strongly connected components.
Meta graph

• Meta Graph – The graph obtained by shrinking each strongly connected component down to a single meta-node and drawing an edge from one meta-node to another if there is an edge between their respective components.
• Meta Graph is a DAG. (Why?)
• In other words, every directed graph is a dag of its strongly connected components.
Binary search

- Looking up a word in the dictionary

- Problem: Find x in a sorted array A[1…n]

- Algorithm:
  - Compare x with middle element of array A
  - Recursively find x in left subarray or right subarray
Binary search pseudocode

Binary_search(A[], key, first, last)

    if (first > last) return not_found

    mid = (first + last)/2

    if (key == A[mid]) return mid

    if (key < A[mid])
        return Binary_search(A[], key, first, mid-1)

    if (key > A[mid])
        return Binary_search(A[], key, mid+1, last)
Divide and Conquer

• Divide – the problem (instance) into one or more subproblems

• Conquer – each subproblem recursively

• Combine – separate solutions
Binary search pseudocode

Binary_search(A[], key, first, last)

    if (first > last) return not_found  //base case

    mid = (first + last)/2

    if (key == A[mid]) return mid

    //divide & conquer
    if (key < A[mid])
        return Binary_search(A[], key, first, mid-1)

    if (key > A[mid])
        return Binary_search(A[], key, mid+1, last)
Binary search revisited

• Divide: compare x with middle element
• Conquer: recurse in one subarray
• Combine: trivial

• Running time
  – $T(n) = T(n/2) + O(1)$
  – **Master theorem:** $T(n) \leq a \cdot T(n/b) + O(n^d)$
    1. $T(n) = O(n^d)$ if $a < b^d$
    2. $T(n) = O(n^d \log n)$ if $a = b^d$
    3. $T(n) = O(n \log_{b^d} a)$ if $a > b^d$

  – Using master theorem, $T(n) = O(\log n)$ (Case 2)
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- a) Indicate the pre and post numbers of the nodes.

- Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.

• Solution
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

![Graph Diagram]

- a) Indicate the pre and post numbers of the nodes.

Solution

![Solution Diagram]
Solved Exercise

Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

a) Indicate the pre and post numbers of the nodes.

Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.

• Solution
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- a) Indicate the pre and post numbers of the nodes.
- Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.

• Solution
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

```
A --> C --> I
|     |     |
D --> F --> G
|     |     |
E --> H
```

a) Indicate the pre and post numbers of the nodes.

Solution

```
A 1, 2, 3, 4, 5, 6
C 2, 3, 4, 5, 6
E 4, 5, 6
D 3, 4, 5, 6
F 5, 6
H 7,
```

G 5, 6
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.
• Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.
• Solution
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- a) Indicate the pre and post numbers of the nodes.
  - Solution

```plaintext
A  D  G
  |   |
  C  F
  |
I  E  H
```

1, 3, 2, 4, 9, 5, 6, 7, 8
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- a) Indicate the pre and post numbers of the nodes.
- Solution
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- a) Indicate the pre and post numbers of the nodes.

Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.

• Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

![Graph Diagram]

• a) Indicate the pre and post numbers of the nodes.
• Solution

![Solution Diagram]
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.

• Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.
• Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• a) Indicate the pre and post numbers of the nodes.

• Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• b) What are the sources and sinks of the graph?
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- b) What are the sources and sinks of the graph?
- Solution
  - From the DFS performed in the previous step we are aware that vertex A is a source node since it has the highest post number.
Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

b) What are the sources and sinks of the graph?

Solution

- From the DFS performed in the previous step, we are aware that vertex G has the lowest post number. Thus, vertex G is a sink.
Solved Exercise

- Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

- b) What are the sources and sinks of the graph?

- Solution
  - From the DFS performed in the previous step, we are aware that vertex G has the lowest post number. Thus, vertex G is a sink.
  - Also, in the above DFS while visiting the vertices in the adjacency list of vertex F we could have visited vertex H before vertex G. This would have lead to vertex H having the least post number. Thus, vertex H can be a sink as well.
• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• c) What topological ordering is found by the algorithm?

• Solution
Solved Exercise

• Run the DFS-based topological ordering algorithm on the following graph. Whenever you have a choice of vertices to explore, always pick the one that is alphabetically first.

• c) What topological ordering is found by the algorithm?
• Solution

A → I → C → E → D → F → H → G