JavaHyperText Topics

“Graphs”, topics:
- 4: DAGs, topological sort
- 5: Planarity
- 6: Graph coloring

Announcements

Monday after Spring Break there will be a CMS quiz about “Shortest Path” tab of JavaHyperText. To prepare:
- Watch the videos (< 15 min) and their associated PDFs (in total 5 pages)
- Especially try to understand the loop invariant and the development of the algorithm

Yesterday, 2018 Turing Award Winners announced

- Won for deep learning with neural networks
  - Facial recognition
  - Talking digital assistants
  - Warehouse robots
  - Self-driving cars
  - …see NYTimes article

Neural networks are graphs!

Neural Network

Neurons in brain receive input, maybe fire and activate other neurons

Neural Network

Input layer

Hidden layers

Output layer
Sorting

CS core course prerequisites

Problem: find an order in which you can take courses without violating prerequisites

e.g. 1110, 2110, 2800, 3110, 3410, 4410, 4820

Topological order

A topological order of directed graph G is an ordering of its vertices as \( v_1, v_2, \ldots, v_n \) such that for every edge \( (v_i, v_j) \), it holds that \( i < j \).

Intuition: line up the vertices with all edges pointing left to right.

Cycles

- A directed graph can be topologically ordered if and only if it has no cycles
- A cycle is a path \( v_0, v_1, \ldots, v_p \) such that \( v_0 = v_p \)
- A graph is acyclic if it has no cycles
- A directed acyclic graph is a DAG

Is this graph a DAG?

- Deleting a vertex with indegree zero would not remove any cycles
- Keep deleting such vertices and see whether graph "disappears"

And the order in which we removed vertices was a topological order!

Algorithm: topological sort

```java
int k = 0;
// inv: k nodes have been given numbers in 1..k in such a way that
// if n1 <= n2, there is no edge from n2 to n1.
while (there is a node of in-degree 0) {
    Let n be a node of in-degree 0;
    Give it number k;
    Delete n and all edges leaving it from the graph.
    k = k + 1;
}
```

JavaHyperText shows how to implement efficiently:

\( O(V+E) \) running time.
Graph Coloring

How many colors are needed to ensure adjacent states have different colors?

Map coloring

Uses of graph coloring

How to color a graph

Assume colors are integers 0, 1, ...

Assume colors are integers 0, 1, ...

If there are d vertices, need at most d+1 available colors
Analysis

```java
void color() {
    for each vertex v in graph:
        c = find_color(neighbors of v);
        color v with c;
}
```

```java
int find_color(vs) {
    int[] used = new int[vs.length() + 1];
    for each vertex v in vs:
        if color(v) <= vs.length():
            used[color(v)]++;
    return smallest c such that used[c] == 0;
}
```

**Time:** $O(\text{vs.length}())$

**Time:** $O(\# \text{neighbors of } v)$

**Total time:** $O(E)$

Use the minimum number of colors?
Maybe! Depends on order vertices processed.

---

**Bipartite graphs**

**Bipartite:** vertices can be partitioned into two sets such that no edge connects two vertices in the same set.

**Matching problems:**
- Med students & hospital residencies
- TAs to discussion sections
- Football players to teams

Fact: $G$ is bipartite iff $G$ is 2-colorable

---

**Four Color Theorem**

Every “map-like” graph is 4-colorable
[Appel & Haken, 1976]
Four Color Theorem

Every “map-like” graph is 4-colorable
[Appel & Haken, 1976]

“map-like”? = planar

Planarity

A graph is planar if it can be drawn in the plane without any edges crossing

Discuss: Is this graph planar?

Planarity

A graph is planar if it can be drawn in the plane without any edges crossing

Discuss: Is this graph planar?

Planarity

A graph is planar if it can be drawn in the plane without any edges crossing

Discuss: Is this graph planar?

YES!

Detecting Planarity

Kuratowski’s Theorem:

A graph is planar if and only if it does not contain a copy of $K_5$ or $K_{3,3}$ (possibly with other nodes along the edges shown).
John Hopcroft & Robert Tarjan

- Turing Award in 1986 “for fundamental achievements in the design and analysis of algorithms and data structures”

- One of their fundamental achievements was a $O(V)$ algorithm for determining whether a graph is planar.

David Gries & Jinyun Xue

Tech Report, 1988

Abstract: We give a rigorous, yet, we hope, readable, presentation of the Hopcroft-Tarjan linear algorithm for testing the planarity of a graph, using more modern principles and techniques for developing and presenting algorithms that have been developed in the past 10-12 years (their algorithm appeared in the early 1970’s). Our algorithm not only tests planarity but also constructs a planar embedding, and in a fairly straightforward manner. The paper concludes with a short discussion of the advantages of our approach.

Happy Spring Break!

Java Island, Southeast Asia