Prof. Alexandra Nilles

Western Washington University

4 November

Announcements

- Mid-term is underway!
- If in DAC testing center, message me your scheduled time
- HW 4 due this Friday
- Delayed office hours today

Heaps

- Definition: a complete binary tree with vertical ordering
- vertical ordering: for every element X with parent P, P <= X
- Complete: every "level" is full except last level, which is filled left to right. Guarantees height goes like O(log N).
- "Min heap": minimum element is always the root, parents always smaller than children
- "Max heap": reverse ordering, max value at root

Add element to heaps

```
               10
          /          \
        20              80
   /        \         /    \
  40         60      85     99
 /   \         /
50    700    65
```

1. Add 15: right child of 60
2. Restore heap ordering: new element is shifted up tree to proper place
    - "percolate" / "bubbling" / "sifting"
    - swap with parent

Remove element from heap

- Case: remove root
- replace with rightmost leaf (to maintain completeness)
    - but this breaks our ordering property!
- Solution: bubble *down*: swap with smaller child (why?)
- log(n) bubbles up or down for adding / removing

Array representation

- Since we know we have a *complete* tree, with careful indexing we can implement a heap in an array
- Let index of root = 1
- For any node n at index i:
    - index of `n.left` = 2i
    - index of `n.right` = 2i+1
    - Parent index of n?
        - floor of i/2
- If array runs out of space, we can copy data into larger array

Heap Sort

- Basic idea: add each element to heap, then remove top element N times
- "Selection sort with the right data structure"
- In-place solution: maintain max-heap in array, build sorted array from back to front of array
- Algorithm performance? O(n lg n)
- Unstable, in-place
- Good bound on worst-case scenarios, makes it well suited for real-time aapplications
- Not easily parallelizable

Priority Queue

- "Abstract Data Type" usually implemented with heaps
- ADT vs data structure?
  - ADT is more formal, explicitly a mathematical model, defined by semantics: in, out, and invariants / guarantees
  - data structure refers to a particular implementation

6 November

Intro to Graphs

Vocab / structure:

- network
- nodes / vertices / entities
- edges (links, relations)
- directed vs undirected
- node data
- edge weights
- vertex degree
  - k-regular graph all vertices have degree k
  - in-degree, out-degree
- order of graph is number of vertices
- size of graph is number of edges
- subgraphs
- tree is a directed acyclic graph (DAG)
- path in graph: sequence of vertices with pairwise edge connections
- cycle: path that ends at originating node

Why graphs?

- Highly expressive encoding, maps to many problem domains
- independent set problem: pairwise nonadjacent vertices
- epidemiology, social network
- connectivity of graphs:
  - strong connectivity: (u,v) path for each pair u, v of vertices
  - k-connectivity: can deleting k edges create two disconnected components?
- Eulerian circuits: can we make a loop in graph while crossing each bridge only once?
  - traffic planning, bus network routing
- Neural networks
  - Graph neural networks: input is graph (instead of vector), predict properties of entire graph
  - Human brain: 100 billion neurons, 100 trillion synaptic connections (approx 1k-5k connections per neuron)
  - ChatGPT: 176 billion neurons

    * 85,000 nodes in layer connected to all 85,000 nodes in next layer, quadrillion connections? $10 * *16$

- Semantics of visual scenes
- Isomorphisms, graph symmetry: "am I seeing the same arrangement of entities as I saw before"
- Lots of difficult but beautiful problems

Representing graphs:

- drawings, planar graphs, embeddings
- connections to automata, autopilot
- adjacency list, adjacency matrix