The Heap

Topics for this week:
- Heap
  - a new data structure
  - can be used to implement a priority queue, i.e. can efficiently support these operations:
    - insert, removeMin (and update)
- HeapSort - another $O(n \log n)$ sort, non-recursive
Problem: A Traffic Simulation

We are studying traffic patterns: our simulation software simulates the trajectories of individual cars and measures the quantities of interest (congestion, speed, etc). One part of the problem studies toll booths and in order for the simulation to proceed, we need to be able to figure out which car arrives at the toll booth next.

Event queue:

Need to be able to:
- insert (a new car with its estimated time of arrival)
- removeMin (which car to process next at the toll booth?)
Priority Queue

A data structure that supports:
- **insert** a new element (with its associate value)
- **removeMin**, returns the element with the smallest value
- possibly **updateValue** of an existing element

Let’s look at previous implementation approaches:

- **insert**
  - **unsorted list**: $(C_1, 10:00), (C_2, 9:30), (C_3, 11:00), (C_4, 9:33)$
  - **sorted list, resort after insert**: $(C_4, 9:33), (C_2, 9:30), (C_1, 10:00), (C_3, 11:00)$
  - **sorted list, insert at proper position**: $(C_4, 9:33), (C_2, 9:30), (C_1, 10:00), (C_3, 11:00)$

- **removeMin**
  - **unsorted list**: $O(n)$
  - **sorted list, resort after insert**: $O(n \log n)$
  - **sorted list, insert at proper position**: $O(n)$

- **not clever, rather do**
  - $(C_4, 9:33), (C_2, 9:30), (C_1, 10:00), (C_3, 11:00)$

- **$0(1)$**
Priority Queue via the Heap

The heap:
- a balanced binary tree
- every parent’s value smaller ($\leq$) than both children’s values

Example: a heap for arriving times 3, 6, 5, 9, 7, 4, 1?

Notice that a heap of $n$ elements has height $\approx \log_2 n$. 
How to **insert** to a heap?

- place the element as the last element in the last row (or start a new row if the last row is already full)

- then what?

  idea: if parent's value > the current child's value, swap them

Pseudo code:

```python
def insert(heap, x):
    current = None
    new_element = Node(x)
    heap.append(new_element)
    while current is not the root and current's parent's value > current's value:
        swap current's value with its parent's value
        current = current's parent
```

Priority Queue via the Heap
Priority Queue via the Heap

How to **removeMin** from a (min-)heap?

Pseudo code:

```python
def removeMin(heap):
    minvalue = root's value
    root's value = last element's value
    remove last elem (its parent points to None)
    current = root
    while current's value > left child's value or right child's value:
        let minchild = child with the smaller value
        swap current's value with minchild's value
        current = minchild
```

Time complexity: \( O(\log n) \) for both insert and removeMin
Implementing Heaps

As a linked data structure:

- node object with slots:
  - value
  - parent
  - left child
  - right child
  - possibly also name, etc. \(<\) does not influence the comparisons

E.g., like a car licence plate

Via arrays:

- left child \((k) = 2k + 1\)
- right child \((k) = 2k + 2\)
- parent \((k) = \lfloor \frac{k-1}{2} \rfloor\)

Index: 0 1 2 3 4 5 6 7 8

Values: 1 5 3 7 9 6 4 8 11

Diagram:

```
  1
 / \  
5—\—3
 \   
7   9   6   4
    \  
   11
```

```
Heap Sort

How to sort with heaps?

3, 1, 7, 14, 2, 8, 9, 4, 6

Heapify:
- Create empty heap
- For x in lst:
  - Insert(Heap, x)
- While heap is not empty:
  - Print(removeMin(heap))

Heapify: \(O(\log n)\) times insert:
- \(O(\log n)\) + \(O(\log n)\) for each insertion
- Cannot be done in \(O(n)\)

The rest of the code clearly: \(O(n \log \log n)\)

Overall: \(O(n \log \log n)\)