Data Structures
  Linked Data Structs

Implementing linked structs using arrays (linear of simple datatype)
Records with arrays

\[ \begin{array}{c}
\text{data}_1 \\
\text{data}_2 \\
\text{data}_3 \\
\text{data}_4 \\
\end{array} \quad \begin{array}{c}
\text{data}_1 \\
\text{data}_2 \\
\text{data}_3 \\
\text{data}_4 \\
\end{array} \quad \cdots \\
\text{data}_1 \quad \text{data}_2 \quad \text{data}_3 \quad \text{data}_4 \\
\vdots \quad \vdots \quad \vdots \quad \vdots \\
\end{array} \]

n records: arrays of size n. Store dataj component of the ith record in dataj[i]

Pointers:

\[
\begin{array}{c}
\text{i}^{th} \text{ record} \\
\text{j}^{th} \text{ record} \\
\end{array} \]

pointers = indices into arrays.
Do-it-yourself memory management:
- `new` produces a new record
- `delete` de-allocates record

Reuse: delete puts record on a linked list of deallocated records.

Linked structures
- **Linked lists**
  - **Stack**
    - Simple (single forward link, null pointer at end)
  - **Circular** (simple except last link points to first record)

Linked lists

Queue

(i)  \[ \text{front} \rightarrow \text{record} \rightarrow \cdots \rightarrow \text{rear} \]
doubly linked circular or not

Deque (Double-ended queue)
insert/delete from either end

Doubly linked list advantages
traverse forwards and backwards in passive way (data struct is unchanged).

traversing a singly linked list in both directions:
Deque with one link per node. (Also list that can be traversed passively in both directions)

\[
\text{prev} := \text{prev}
\]

\[
\text{prev} := \text{cur}
\]

\[
\text{cur} := (\text{cur} \rightarrow \text{link}) \oplus \text{temp}
\]

\[
A \oplus C = \text{cur} \rightarrow \text{link}
\]

\[
\text{temp} := A
\]

\[
\text{cur} := (A \oplus C) \oplus A
\]

Trees
A tree is an undirected connected, acyclic graph.
Size of \(T\) = \# of vertices.
Any tree of size \(n > 0\) has exactly \(n-1\) edges.
Proof (by induction on \(n\))
\(n = 1:\)
- 1 vertex
- 0 edges
\]
$n > 1$: (Assume tree for trees of size $n-1$ (n-2 edges))

Note: In any tree $T$ of size $n$, there is a node of degree 1:
1st: no vertex has degree 0 (because $T$ is connected)
2nd: Assume no $v$ has degree 1. So all $v$'s have degree $\geq 2$.

Can find a path of arbitrary length, so must encounter (eventually) some node twice.
Thus there is a cycle. But $T$ is a tree and has no cycles.
So $\exists v$ of degree 1.

Remove $v$ and its edge. Result is another tree $T'$ of size $n-1$.
Inductive hypothesis: $T'$ has
Thus $T$ has exactly $n-1$ edges. QED.

Facts

1. Any graph with $n$ vertices and $\geq n$ edges must have a cycle.

2. Any graph with $n$ vertices and $< n-1$ edges must not be connected.

A rooted tree is a tree with a distinguished node (the root).

 EXISTS UNIQUE ORIENTATION OF THE EDGES (ARROWS) SO THAT NO EDGES GO INTO THE ROOT, AND EVERY NONROOT NODE HAS EXACTLY ONE EDGE COMING INTO IT.