CSCI 241

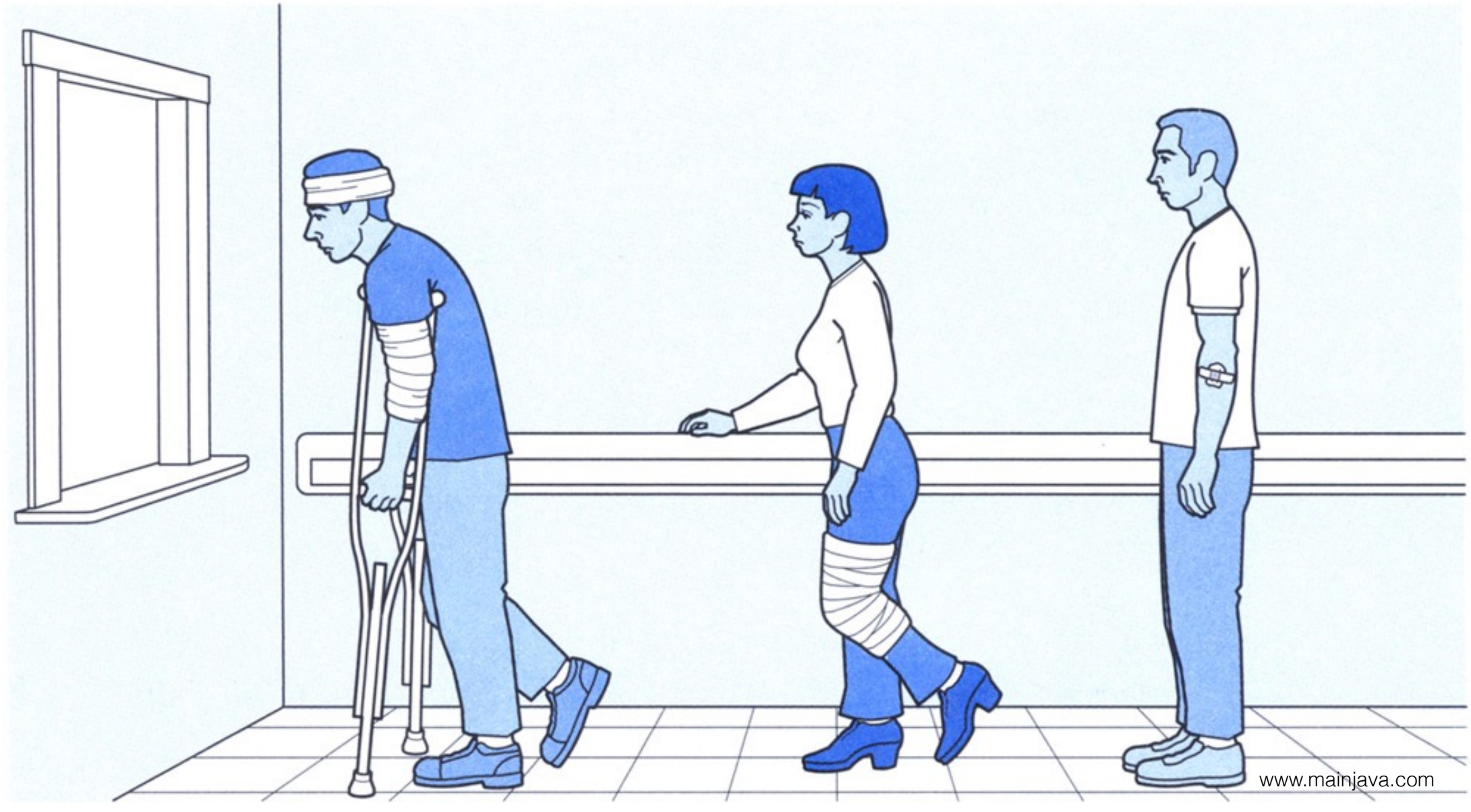
Scott Wehrwein

The Priority Queue ADT

Goals

Understand the purpose and interface of the Priority Queue ADT.

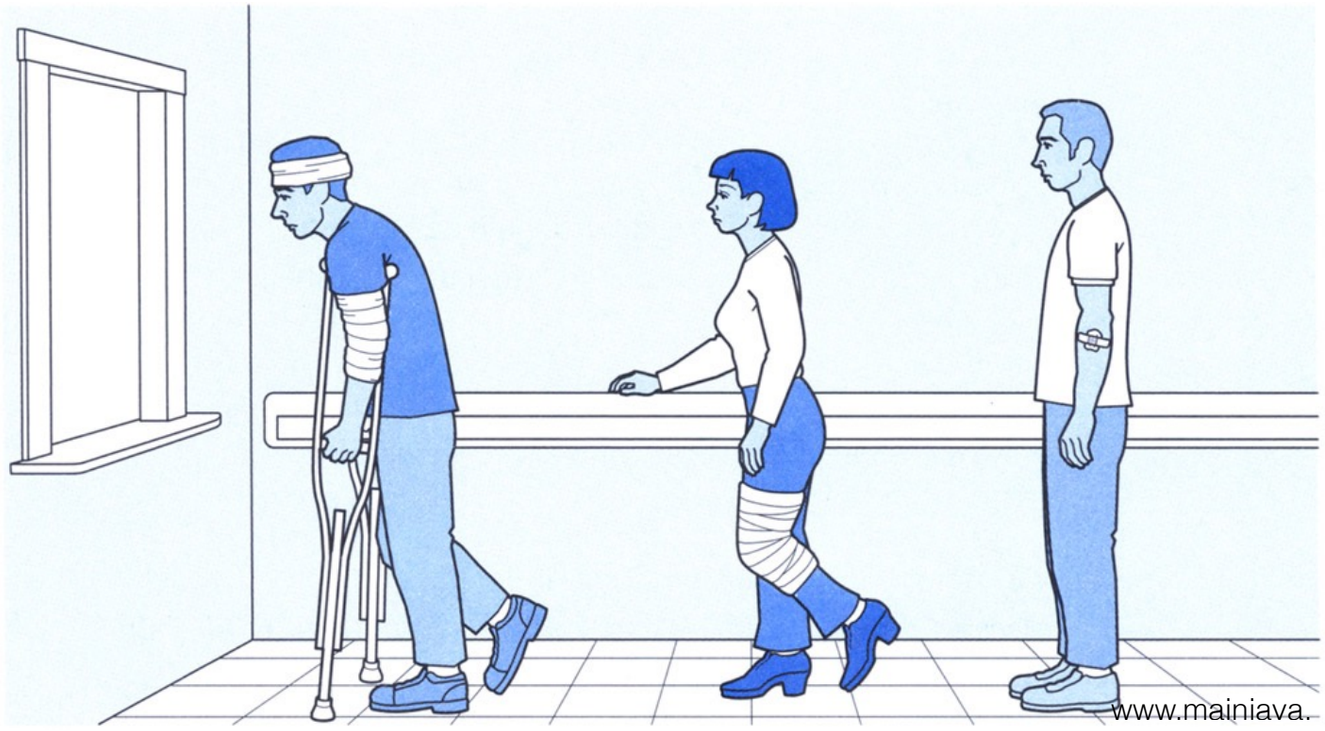
Priority Queue



Queue vs Priority Queue

add (`enqueue`):
  insert an item into the queue

remove (`dequeue`):
  remove the **first item** to be inserted

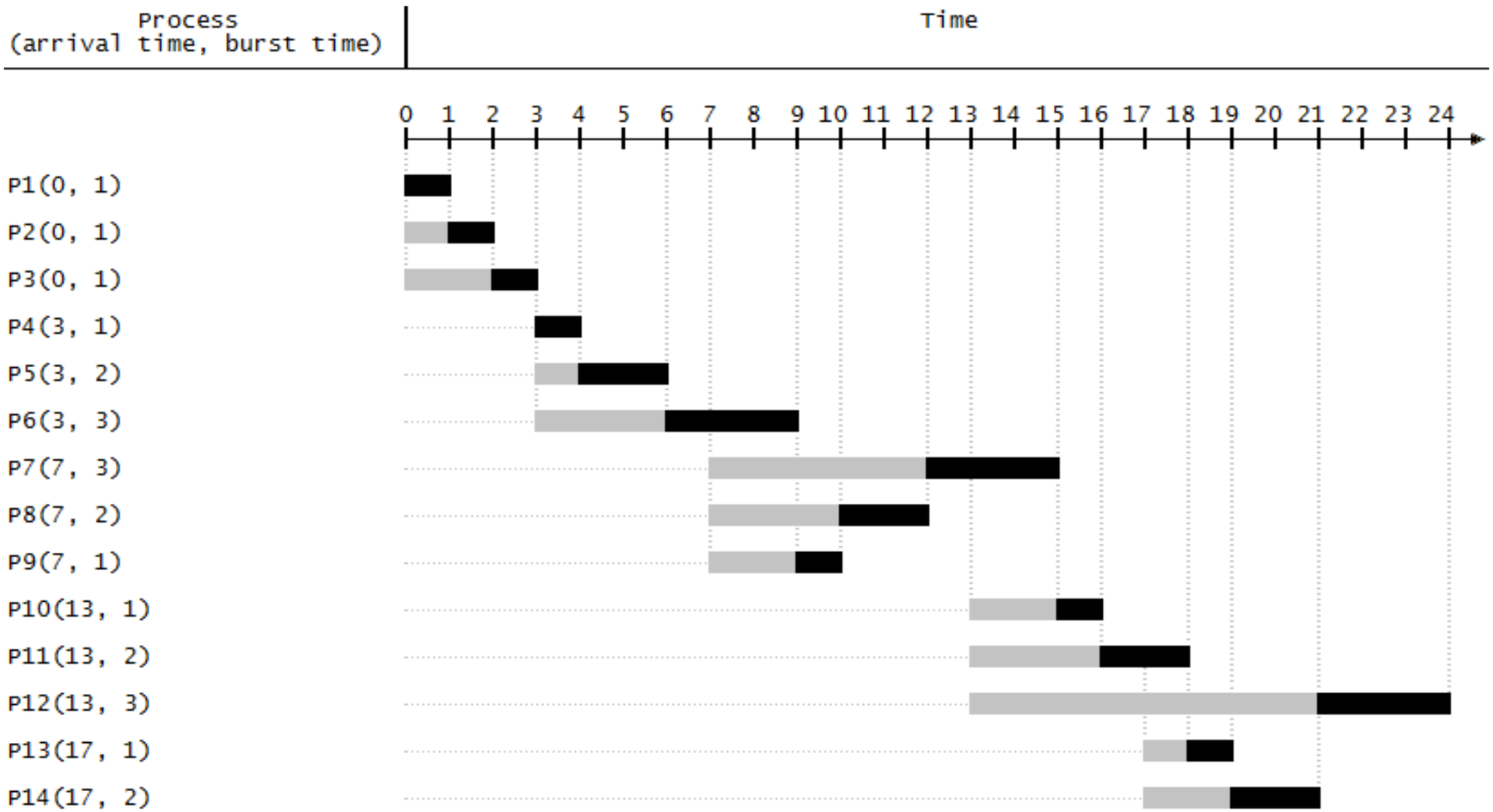


add (`add`):
  insert an item into the queue

remove (`poll`):
  remove the **highest-priority** item in the queue

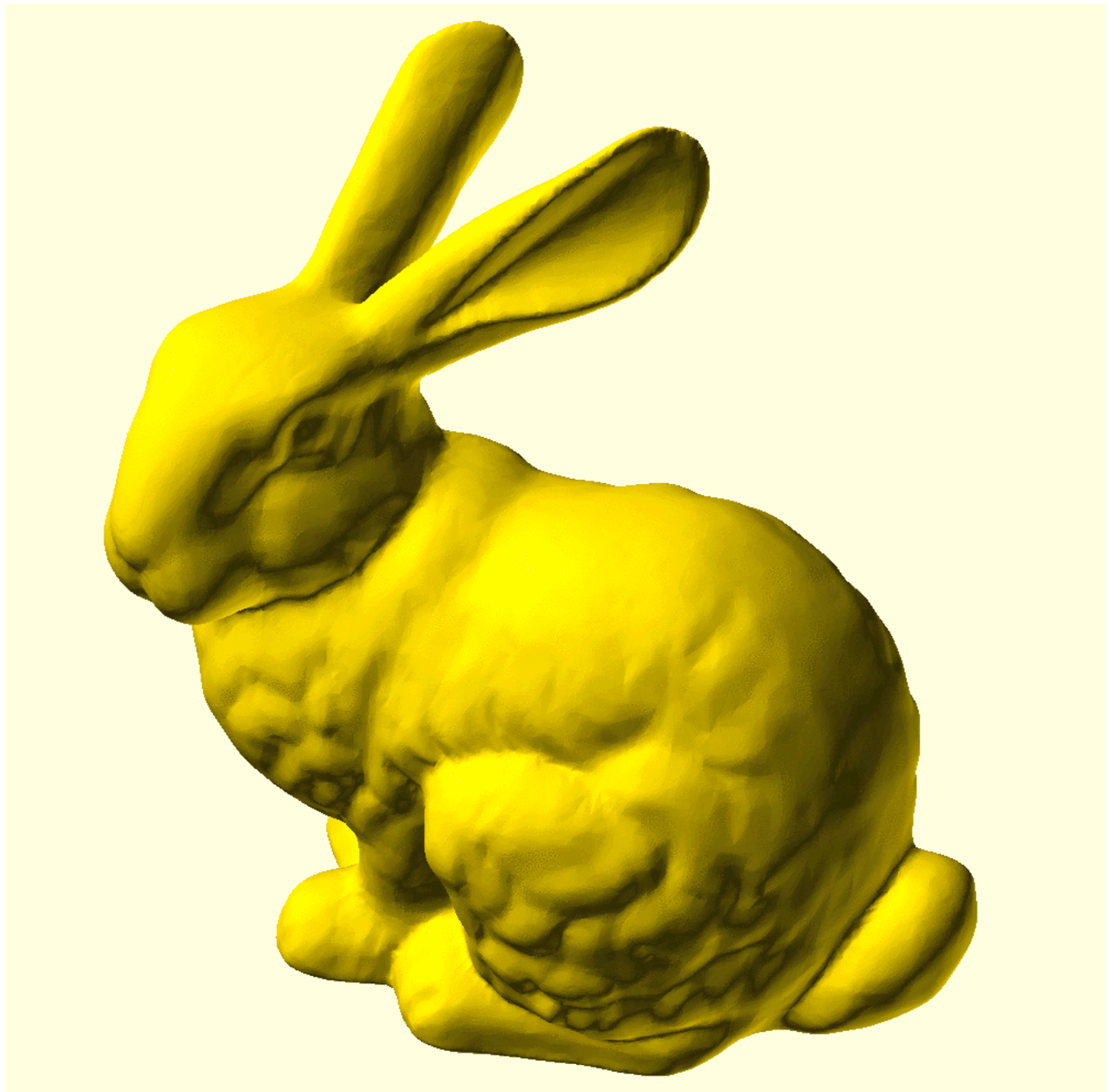
What is this good for?

Systems - CPU scheduling, Virtual memory paging



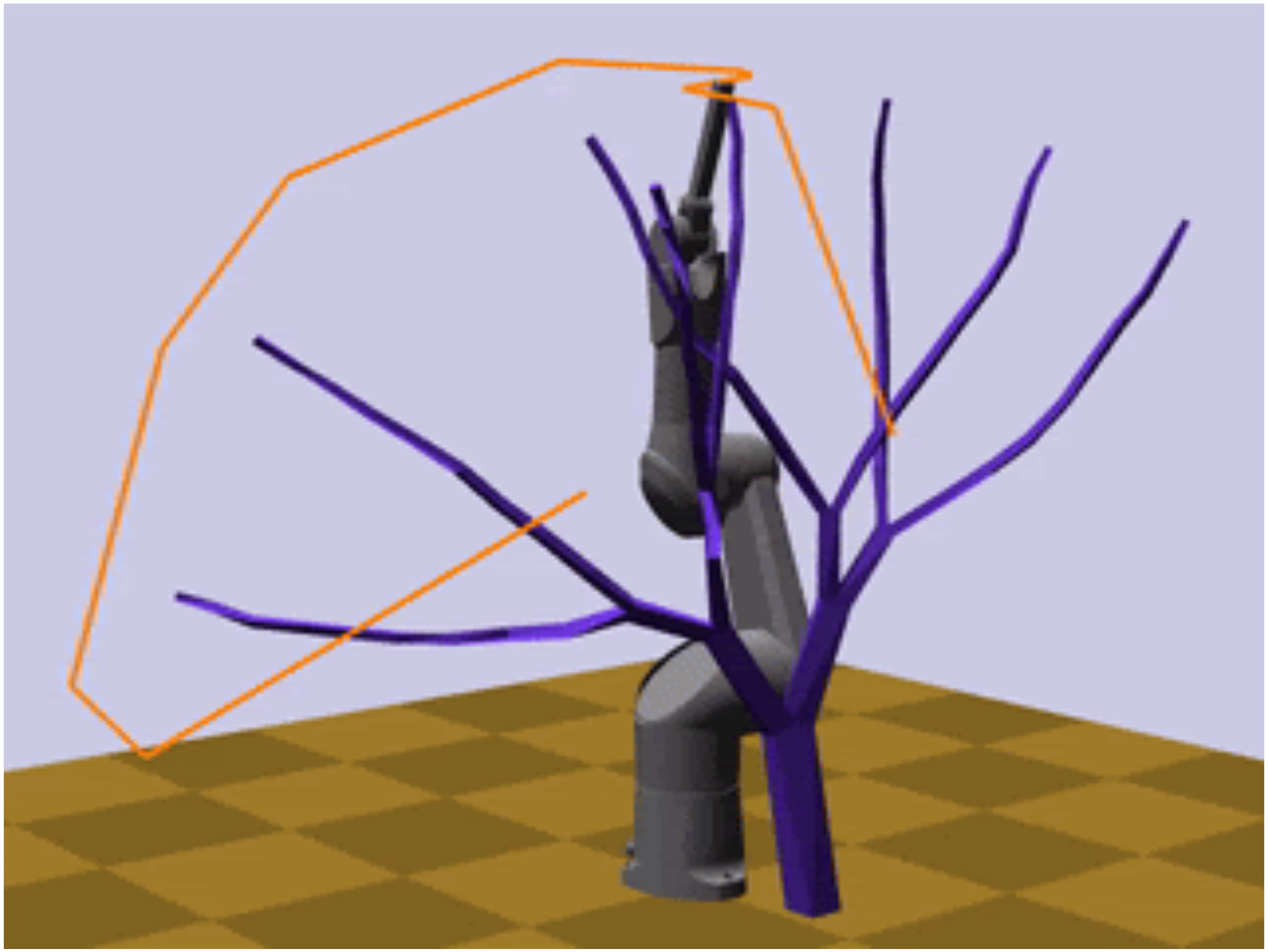
What is this good for?

Computer Graphics - Mesh simplification, Collision detection



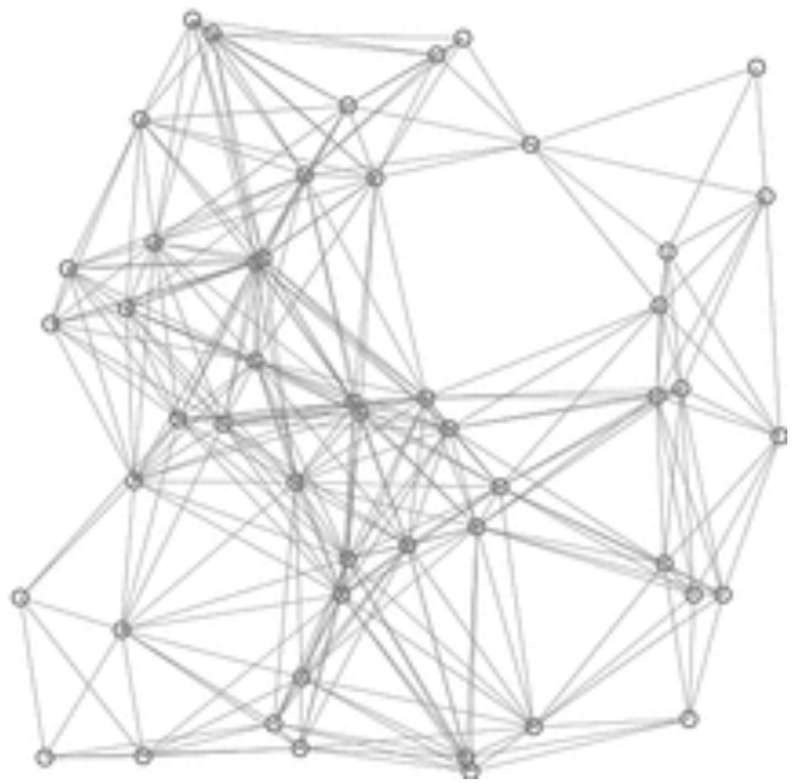
What is this good for?

AI - Robot Navigation and Path Planning



What is this good for?

Statistics (largest M values)
Computer Science (bin packing, graph search)



In fact, you'll use your Priority Queue (A3)
to implement a graph algorithm (A4)!

Priority Queue

Like a Queue, but:

- Each item in the queue has an associated **priority** which implements **`Comparable`**
- Removing an element (**`poll`**) returns the item with the "highest priority"

defined in this class as: the element with the "*smallest*" associated priority value

Take note:
this is easy to get mixed up

Ties are broken arbitrarily

Priority Queue: Java

```
interface PriorityQueue<E> {
 boolean add(E e); // insert e
 E peek(); // return min element
 E poll(); // remove/return min element
 void clear();
 boolean contains(E e);
 boolean remove(E e);
 int size();
 Iterator<E> iterator();
}
```

`E` represents the value and is *also* `Comparable`.

i.e., determines the priority

Priority Queue: A3

```
interface PQ<V, P extends Comparable<P>> {
 boolean add(V v, P p); // add v w/ priority p
 V peek(); // return highest-priority val
 V poll(); // remove/return highest-priority val
 void clear();
 boolean contains(V val);
 void changePriority(V v, P newP)
 int size();
}
```