May work with a partner.
This will be written. HW due.
Due next Friday.
Look for HW tomorrow - will be.
HW due.
Announcement.
CS 314: Pseudo code.
Want to support:

Items that have priorities.

A data structure which stores

What is a priority queue?

Introducing pseudo code.

Today: Priority Queue (see 2.5 in book).
- Sort the list

- Find min: O(n) (if in terms of list)
- Insert - insert now item at end: O(1)
- Insert to sorted in sorted list: O(log n + C)
- How long: O(n)
- Down

What is a simple way to do this?
Ex: [Diagram of a binary tree]

- Balanced binary tree, there is a key.
- An empty tree needs a level.
- All leaves are at the same level.
- Breadth-first traversal.
- Complete binary tree.

(Min, Max) Heap:

- Complete balanced tree.
- The left and right children are ordered.
- The root is the maximum element.
Here we can do better. Store in an array.

Node/Parent structure.

How to implement?
This method is not space efficient unless the free list is a complete binary tree.
Adding a new element:

Then what?

Put new value in [Ⅳ]+[Ⅳ]

Insert (Ⅳ)

Want to happy

Bubble up the

element in front

until heap property

is satisfied.
Heapy-up(H[i])

if H[i] < H[j[i]]

let j' = j[i]

end if

end if

function

Pseudo code:

Heapy-up(H[i])

if H[i] < H[j[i]]

let j' = j[i]

end if

end if

set j' equal to i/2

end while

end function
$D$: Spend all time finding & checking

$H_0$: Heavily-up([h, k]) works as specified for k > k_

$H_A$: $f(k) = \frac{1}{k}$ is an already a heap

Since $H_A$ is the next:

Base case: $\sum_{i=1}^{n} = 1$.

$\forall k: \sum_{i=1}^{k} f(k)$ exists. The heap construction is already a heap.

Proof: Heavily-up([1, 1])
What about deleting?

Delete $(H, i)$

$\uparrow$ position in array

Delete $(H, 2)$

Heapify!

bubble down in tree
Heapify-down(H, i):
    n ← length(H)
    if 2i > n ← at a leaf
        terminate with H unchanged
    else if 2i < n
        left ← 2i
        right ← 2i + 1
        j ← min( H[left], H[right] )
    else if 2i = n
        j ← 2i
    end if

    if H[j] < H[i]
        swap H[i] and H[j]
        Heapify-down(H, j)
    end if
Prop: The procedure `Heapsify-down(H, i)` fixes the heap property in $O(\log n)$ time (assuming $H[i]$ is a heap with $H[i]$ too big).

Proof: (Similar - see p. 64 in text)
Use heathy-down

(heathy-down)

There will be three categories where all operations will be performed. Primary queues can be implemented.