06-0: Finding Max & Min

- What is the smallest exact number of comparisons required to find the maximum element of a list with \( n \) elements?
- What is the smallest exact number of comparisons required to find the minimum element of a list of \( n \) elements?
- What is the smallest number of comparisons required to find the maximum and minimum elements of a list?

06-1: Finding Max & Min

- What is the smallest number of comparisons required to find the maximum element of a list? \( (n - 1) \)
- What is the smallest number of comparisons required to find the minimum element of a list? \( (n - 1) \)
- What is the smallest number of comparisons required to find the maximum and minimum elements of a list?
  - Compare pairs: then compare the largest to the current largest, and smallest to the current smallest \( \lceil \frac{n}{2} \rceil + (n - 2) \)

06-2: Selection Problem

- What if we want to find the \( k \)th smallest element?
  - Median: \( k = \lceil \frac{n}{2} \rceil \)th smallest element, for odd \( n \)
- What is the obvious method?
- Can we do better?

06-3: Selection Problem

- What if we want to find the \( k \)th smallest element?
  - Median: \( k = \lceil \frac{n}{2} \rceil \)th smallest element, for odd \( n \)
- What is the obvious method?
  - Sort the list, select the element at index \( k \)
- Can we do better?

06-4: Selection Problem

```plaintext
Quicksort(A, low, high)
if (low < high) then
   pivotindex $\leftarrow$ Partition(A, low, high)
   Quicksort(A, low, pivotindex - 1)
   Quicksort(A, pivotindex + 1, high)

Partition(A, low, high)
 pivot $\leftarrow$ A[high]
 i $\leftarrow$ low - 1
 for j $\leftarrow$ low to high - 1 do
   if (A[j] $\leq$ pivot) then
     i $\leftarrow$ i + 1
     swap A[i] $\leftrightarrow$ A[j]
 swap A[i+1] $\leftrightarrow$ A[high]
 return i+1
```

06-5: Selection Problem
Select(A, low, high, k)
    if (low = high)
        return A[low]
    pivot = Partition(A, low, high)
    adj_pivot = piv - low + 1
    if (k = adj_pivot) then
        return A[pivot]
    if (k < adj_pivot) then
        return Select(A, low, pivot - 1, k)
    else
        return Select(A, pivot + 1, high, k - adj_pivot)

Running time (Best and worst case)?

06-6: Selection Problem

- Best case time:
  \[ T(n) = T(n/2) + c \cdot n \in \Theta(n) \]

- Worst case time:
  \[ T(n) = T(n - 1) + c \cdot n \in \Theta(n^2) \]

- Average case time turns out to be \( \Theta(n) \), but we’d like to get the worst-case time down.

06-7: Selection Problem

- Improving worst-case time for selection
  - We need to guarantee a “good” pivot to get \( \Theta(n) \) time for selection
  - How much time can we spend to find a good pivot, and still get \( \Theta(n) \) time for selection?

06-8: Selection Problem

- Improving worst-case time for selection
  - We need to guarantee a “good” pivot to get \( \Theta(n) \) time for selection
  - How much time can we spend to find a good pivot, and still get \( \Theta(n) \) time for selection?
  - \( O(n) \)!

06-9: Selection Problem

- Finding a “Good” pivot (one that is near the median) in linear time:
  - Split the list into \( \frac{n}{5} \) list of length 5
  - Do an insertion sort on each of the \( \frac{n}{5} \) lists to find the median of each of these lists
  - Call select recursively to find the median of the \( \frac{n}{5} \) medians
06-10: Selection Problem

• How good is the pivot chosen by this method? How many elements are guaranteed to be less than the pivot?
  • Each row has 5 elements
  • Half of the rows will have a median less than the pivot
  • Each of these rows will have 3 elements less than the pivot

\[3 \times \left\lceil \left\lceil \frac{n}{5} \right\rceil \frac{1}{2} \right\rceil\]

06-11: Selection Problem

• Not all of those rows have exactly 3 elements less than the pivot:
  • The total number of elements might not be divisible by 5 (so one row would have < 5 elements)
  • The row containing the pivot itself only has 2 elements less than the pivot (not 3)

\[3 \times \left( \left\lceil \frac{n}{5} \right\rceil \frac{1}{2} - 2 \right) \geq \frac{3n}{10} - 6\]
06-15: Selection Problem

- Worst case time for selection for a problem of size $n$:
  - $\Theta(n)$ time to do partition, $n/5$ insertion sorts
  - Time to find the median of medians (looking at $n/5$ elements)
  - Time to make the recursive call (to a problem of no more than size $7n/10 + 6$

06-16: Selection Problem

$$T(n) \leq \begin{cases} 
C_1 + T\left(\left\lceil \frac{n}{5} \right\rceil \right) + T(\left\lceil \frac{7n}{10} \right\rceil + 6) + C_2 \times n & \text{if } n < 140 \\
\text{otherwise} 
\end{cases}$$

06-17: Selection Problem

$$T(n) \leq \frac{C \times n}{5} + C + \frac{7 \times C \times n}{10} + 6 \times C + C_2 \times n$$

$$= \frac{9 \times C \times n}{10} + 7 \times C + C_2 \times n$$

$$= C \times n + \left(7 \times C + C_2 \times n - \frac{C \times n}{10} \right)$$

06-18: Selection Problem

$$7 \times C + C_2 \times n - \frac{C \times n}{10} \leq 0$$

$$C \times \left(7 - \frac{n}{10}\right) \leq -C_2 \times n$$

$$C \times \left(\frac{n}{10} - 7\right) \geq C_2 \times n$$

$$\frac{n}{10 \times C} \geq C_2 \times (n/10 - 7)$$

Note that we must insist that $n > 70$. If $n \geq 140$, then this is true if $C > 20 \times C_2$

06-19: Selection Problem

- Selection takes time $O(n)$
  - in fact, $\Theta(n)$, since each recursion steps takes time $\Omega(n)$

- So, we can use Selection to make Quicksort take time $\Theta(n \log n)$ worst case
  - Would that be a good idea?