Divide and Conquer

COMP 215 Lecture 4
The Divide and Conquer Approach

• Divide and conquer algorithms have three components:
  – DIVIDE the problem instance into one or more smaller instances.
  – CONQUER each of the smaller instances (recursively)
  – If necessary, COMBINE the solutions to the smaller instances.

• We will analyze three D&C algorithms over the next couple of days:
  – Mergesort – today.
  – Quicksort – today and Friday.
  – Convex Hull – Wednesday.

• Let's talk about mergesort...
/output: sorted S.

void mergesort(int n, keytype S[]) {
    if (n > 1) {
        int h = floor(n/2), m = n - h;
        keytype U[1..h], V[1..m];
        copy S[1] through S[h] to U;
        copy S[h+1] to S[n] to V;
        mergesort(h, U);
        mergesort(m, V);
        merge(h, m, U, V, S);
    }
}
Mergesort Complexity Analysis

- We will count comparisons. (Assignments would also be reasonable)
- Worst case number of comparisons for the merge operation. $W(h,m) = ???$
- First, what would the best case be?
- Now let's come up with a recurrence for the total running time. (Assume $n$ is a power of 2.)
Mergesort Analysis

- Mergesort recurrence: \( W(n) = W(n/2) + W(n/2) + n - 1. \)
- How can we solve this recurrence?
- What about \( n \) not a power of 2?
Non-Decreasing Proof

- Show that $W(n) = W([n/2]) + W(\lceil n/2 \rceil) + n - 1$ is eventually non-decreasing.

- **Base:** $W(1) = 0$, $W(2) = 1$

- **Induction Hypothesis:** Assume for all $m \leq n$, if $k < m$, $W(k) \leq W(m)$

- We need to show that $W(n) \leq W(n+1)$
• Plugging both sides into recurrence:

\[ W(\lfloor \frac{n}{2} \rfloor) + W(\lceil \frac{n}{2} \rceil) + n - 1 \leq W(\lfloor \frac{n+1}{2} \rfloor) + W(\lceil \frac{n+1}{2} \rceil) + n \]

• Clearly \( \lfloor \frac{n}{2} \rfloor \leq \lfloor \frac{n+1}{2} \rfloor \leq n \)

• So, by induction Hypothesis: \( W(\lfloor \frac{n}{2} \rfloor) \leq W(\lfloor \frac{n+1}{2} \rfloor) \)

• We can make a similar argument to show that

\[ W(\lfloor \frac{n}{2} \rfloor) \leq W(\lceil \frac{n+1}{2} \rceil) \]

• It is easy to see that \( n-1 \leq n \)
Merge Sort Space Analysis

- Let's discuss heaps and stacks.
- Let's draw the recursion tree.
- Let's look at the mergesort code.
- Can we do better?
void mergesort2(int low, int high, keytype S[]){
    if (low < high) {
        int mid = floor((low + high)/2);
        mergesort2(low, mid, S);
        mergesort2(mid+1, high, S);
        merge2(low, mid, high, S);
    }
}

• A note on global variables in the book's pseudocode...
Mergesort2 Space Usage

- How much space does it use?
Quicksort

```c
void qsort(index low, index high, keytype S[]){
    if (low < high) {
        index pivotpoint;
        pivotpoint = partition(low, high, S);
        quicksort(low, pivotpoint-1);
        quicksort(pivotpoint+1, high);
    }
}
```
partition(index low, index high, keytype S[]){
    index i, j;
    keytype pivotitem;
    pivotitem = S[low];
    j = low;
    for (i = low +1; i<= high; i++) {
        if (S[i] < pivotitem) {
            j++;
            exchange S[i] and S[j];
        }
    }
    pivotpoint = j;
    exchange S[low] and S[pivotpoint];
    return pivotpoint;
}
Quicksort Analysis

- Partition?
- Best-case sort?
- Worst-case sort?
Quicksort Analysis

- Partition?
  \[ T(n) = n-1 \]

- Best-case sort?
  \[ B(n) = B\left(\left\lfloor \frac{n-1}{2} \right\rfloor \right) + B\left(\left\lceil \frac{n-1}{2} \right\rceil \right) + n - 1 \]
  \[ B(n) \in \Theta(n \lg n) \]

- Worst-case sort?
  \[ W(n) = W(n-1) + n - 1 \]
  \[ W(n) = \frac{n(n-1)}{2} \in \Theta(n^2) \]
Quicksoort Average Case

- Average case recurrence:

\[ A(n) = \sum_{p=1}^{n} \frac{1}{n} [A(p-1) + A(n-p)] + n - 1 \]

- Solution:

\[ A(n) \approx 1.38(n+1)\lg n \in \Theta(n\lg n) \]

- What about space?