Recurrence Relations

Background

- Recurrence relations are a 'natural' way to describe recursive algorithms
- Complexity of recursive algorithms is derived from solution of recurrence relations
- Really just the application of some common techniques and mathematical 'tricks'

Approach

- Observe algorithm
- Derive recurrence relation from algorithm
- Solve recurrence relation
  - often mathematical 'trick'
  - however, several common methods
- Determine complexity based upon solution to recurrence relation
Ex. 1b: Findmax in Unsorted Array

Recursive: Little Steps

```c
int findmaxR2(int a[], int left, int right)
if (left == right) return a[left]
return max(a[left], findmaxR2(a, left+1, right))
```

General Form

Recurrence Relation

\[ T(0) = T(1) = 0; \quad T(N) = 1 + T(N - 1) \]

Solution of the Form:

\[ T(0) = T(1) = 0; \quad T(N) = 1 + T(N - 1) \]

Ex. 1a: Findmax in Unsorted Array

Recursive: Divide and Conquer

```c
int findmaxR1(int a[], int left, int right)
if (left == right) return a[left]
int mid = (left+right)/2
return max(findmaxR(a, left, mid), findmaxR(a, mid+1, right))
```

General Form

Recurrence Relation

\[ T(0) = T(1) = 0; \quad T(N) = 1 + 2T(N/2) \]
Solution of the Form:
\[ T(0) = T(1) = 0; \ T(N) = 1 + 2T(N/2) \]
- Formula 2.5

Example 2: Quicksort

```c
quicksort(a[], left, right)
if (left >= right) return
int pivot = partition(a, left, right)
quicksort(a, left, pivot-1)
quicksort(a, pivot+1, right)
```

**General Form**

**Recurrence Relation**

\[ T(0) = T(1) = 0; \ T(N) = N + T(N - 1) \] (worst case)
\[ T(N) = N + 2T(N/2) \] (best case)

Solution of the Form:
\[ T(0) = T(1) = 0; \ T(N) = N + T(N - 1) \]
- Formula 2.1
Solution of the Form:
\[ T(0) = T(1) = 0; \quad T(N) = N + 2T(N/2) \]

- Formula 2.4
- Methods to solve
  - traditional (substitution of \(2^n\) for \(N\) \((n = \log N)\))
  - telescoping

Example 3: Mergesort
```c
void mergesort(Item a[], int left, int right)
if (right <= left) return;
int mid = (right + left)/2;
mergesort(a, left, mid);
mergesort(a, mid+1, right);
merge(a, left, mid, right);
```

**General Form**

**Recurrence Relation**
\[ T(0) = T(1) = 0; \quad T(N) = N + 2T(N/2) \]
Recurrence Relations

Three Basic Forms

- \( T(\text{base}) = 0,1; \ T(n) = \text{work} + T(n-1) \)
  - work = 1: did in class
  - work = n: formula 2.1
- \( T(\text{base}) = 0,1; \ T(n) = \text{work} + 2T(n/2) \)
  - work = 1: formula 2.5
  - work = n: formula 2.4
- \( T(\text{base}) = 0,1; \ T(n) = \text{work} + T(n/2) \)
  - work = 1: formula 2.2
  - work = n: formula 2.3

Summary: Recurrence Relations

- Observe algorithm
- Derive recurrence relation from algorithm
- Solve recurrence relation
  - often mathematical ‘trick’
  - however, several common methods
- Determine complexity based upon solution to recurrence relation