Lecture P7: Advanced Recursion

Overview

What is recursion?
- When one function calls ITSELF directly or indirectly.

Why learn recursion?
- New mode of thinking.
- Powerful programming tool to solve a problem by breaking it up into one (or more) smaller problems of similar structure.
  - "Divide et impera"
  - "Veni, vidi, vici"

Julius Caesar (100 BCE - 44 BCE)

Quicksort

Quicksort.
- Partition array so that:
  - some partitioning element $a[m]$ is in its final position
  - no larger element to the left of $m$
  - no smaller element to the right of $m$

Quicksort

Quicksort.
- Partition array so that:
  - some partitioning element $a[m]$ is in its final position
  - no larger element to the left of $m$
  - no smaller element to the right of $m$
- Sort each "half" recursively.
Quicksort

Partition array so that:
- some partitioning element \( a[m] \) is in its final position
- no larger element to the left of \( m \)
- no smaller element to the right of \( m \)
Sort each “half” recursively.

void quicksort(char a[], int left, int right) {
    int m;
    if (right > left) {
        m = partition(a, left, right);
        quicksort(a, left, m-1);
        quicksort(a, m+1, right);
    }
}

quicksort.c (see Sedgewick Program 7.1)

int partition(char a[], int left, int right) {
    int i = left-1; /* left to right pointer */
    int j = right; /* right to left pointer */
    while(1) {
        while (a[++i] < a[right])
            ;
        while (a[right] < a[--j])
            if (j == left)
                break;
        if (i >= j)
            break;
        swap(a, i, j);
    }
    swap(a, i, right);
    return i;
}

partition (see Sedgewick Program 7.2)

main()
#include <stdio.h>
#define N 14
int main(void) {
    char a[] = "pseudomythical";
    printf("Before: %s\n", a);
    quicksort(a, 0, N-1);
    printf("After: %s\n", a);
    return 0;
}

void swap(char a[], int i, int j) {
    char t;
    t = a[i]; a[i] = a[j]; a[j] = t;
}
Quicksort: Performance

Quicksort vs. Insertion sort.

Insertion Sort

<table>
<thead>
<tr>
<th>computer</th>
<th>thousand</th>
<th>million</th>
<th>billion</th>
</tr>
</thead>
<tbody>
<tr>
<td>home pc</td>
<td>instant</td>
<td>2 hour</td>
<td>310 years</td>
</tr>
<tr>
<td>super</td>
<td>instant</td>
<td>1 sec</td>
<td>1.6 weeks</td>
</tr>
</tbody>
</table>

Quicksort

<table>
<thead>
<tr>
<th>thousand</th>
<th>million</th>
<th>billion</th>
</tr>
</thead>
<tbody>
<tr>
<td>instant</td>
<td>0.3 sec</td>
<td>6 min</td>
</tr>
</tbody>
</table>

Super: instant 0.3 sec 6 min

Stay tuned: Lecture T4.

Dragon (Jurassic Park) Curve

Fold a wire in half \( n \) times. Unfold to right angles.

\[
\begin{array}{c|c}
\text{Insertion Sort} & \text{Quicksort} \\
\hline
\text{computer} & \text{thousand} & \text{million} & \text{billion} \\
\hline
\text{home pc} & \text{instant} & 2 \text{ hour} & 310 \text{ years} \\
\text{super} & \text{instant} & 1 \text{ sec} & 1.6 \text{ weeks} \\
\hline
\text{thousand} & \text{million} & \text{billion} \\
\hline
\text{instant} & 0.3 \text{ sec} & 6 \text{ min} \\
\text{instant} & \text{instant} & \text{instant} \\
\hline
\end{array}
\]

Drawing a Dragon Curve

Use simple "turtle graphics."

- \( F \): move turtle forward one step (pen down).
- \( L \): turn left 90°.
- \( R \): turn right 90°.

Example.

- \( F L F L F R F \)
Drawing a Dragon Curve

Use simple “turtle graphics.”
- F: move turtle forward one step (pen down).
- L: turn left 90°.
- R: turn right 90°.

Example.
- dragon(0): F
- dragon(1): F L F
- dragon(2): F L F L F R F
- dragon(3): F L F L F L F L F R F F F

"inverted" dragon(3): reverse string, switch L and R

Recursive Dragon Curve Program

A dragon curve of order n is:
- Dragon curve of order n-1.
- Move left.
- Inverted dragon curve of order n-1.
  - backwards, switch L and R

```
void dragon(int n) {
  if (n == 0)
    F();
  else {
    dragon(n-1);
    L();
    nogard(n-1);
  }
}
```

void F(void) {
  printf("10 0 rlineto\n");
}

void L(void) {
  printf("90 rotate\n");
}

void R(void) {
  printf("-90 rotate\n");
}

drawing in PostScript

```
void dragon(int n) {
  if (n == 0)
    F();
  else {
    dragon(n-1);
    L();
    nogard(n-1);
  }
}
```

Need implementation of nogard().

Drawing a Dragon Curve

Observation: nogard(n) is same as dragon(n), except middle move is R not L

Justification: by definition, nogard(n) is inverted dragon(n).

```
void nogard(int n) {
  if (n == 0)
    F();
  else {
    dragon(n-1);
    R();
    nogard(n-1);
  }
}
```

Nonrecursive Dragon Curve

To write down the whole dragon curve sequence:
1. Put F in every other space.
2. Put L, R (alternating) in every other remaining space.
3. Repeat Step 2 until done.

Proof.

D(0) L N(0) D(0) R N(0) D(0) L N(0) D(0) R N(0)

Drawback: requires excessive memory.
Sequential Dragon Curve

The dragon curve and binary integers.

- The kth turtle turn (ignore Fs) depends on the bit to the left of the rightmost 1 in the binary representation of k.
  - L if bit = 0, R if bit = 1

Proof: (by induction on order of curve)

- Base case: dragon(1) = F L F.
- Assume true for dragon(n), and consider dragon(n+1).
- Recall: only difference between top and bottom halves is their middle moves.

Consequence: simple iterative algorithm that requires little storage.

---

Summary

Why learn recursion?

- New mode of thinking.
- Powerful programming tool to solve a problem.
  - Break it up into smaller subproblems of similar structure
  - Solve subproblems using the same method; then combine results
  - But don't forget the base case!

Examples

- Quicksort.
- Dragon curve.

Many other problems have elegant divide-and-conquer solutions.

- Searching a sorted list (binary search)
- Trees (stay tuned for Lecture P10)