COMPSCI 311: Introduction to Algorithms

Lecture 10: Divide and Conquer

Marius Minea

University of Massachusetts Amherst

slides credit: Dan Sheldon

25 February 2019

Divide and Conquer: Recipe

- Divide problem into several parts
- Solve each part recursively
- Combine solutions to sub-problems into overall solution

Learning Goals

	Greedy	Divide and Conquer
Formulate problem Design algorithm		✓
Prove correctness	✓	
Analyze running time		✓
Specific algorithms	Dijkstra, MST	

Motivating Problem: Maximum Subsequence Sum (MSS)

- **Input**: array A of n numbers, e.g.

$$A = 4, -3, 5, -2, -1, 2, 6, -2$$

- **Find**: value of the largest subsequence sum

$$A[i] + A[i+1] + \ldots + A[j]$$

  - (empty subsequence allowed and has sum zero)
- MSS in example? 11 (first 7 elements)
- Can we extract any observations?

Clicker Question 3

Which of the following is true for a maximum-sum subsequence?

A. It has more positive than negative numbers

B. It does not start nor end with a negative number

C. Any maximal sequence of negative numbers is bordered by a sequence of positive numbers with sum larger in absolute value

A Simple MSS Algorithm

A Problem from HW1 (replace max with sum)

MSS(A)
  Initialize all entries of $n \times n$ array B to zero
  for $i = 1$ to n **do**
    sum $= 0$
    for $j = i$ to n **do**
      compute sum of $A[i]$ ... $A[j]$ ▷ HW1: max
      $B[i, j] =$ sum
    end for
  end for
  Return maximum value among all $B[i, j]$

Running time? $O(n^2)$. Can we do better?

Divide-and-conquer for MSS

- Recursive solution for MSS
- **Idea**:
  - Find MSS L in left half of array
  - Find MSS R in right half of array
  - Find MSS M for sequence that crosses the midpoint

$$A = \overbrace{\underbrace{4, -3, 5}_{L=6}, -2, -1, \underbrace{2, 6}_{R=8}}^{M=11}, -2$$

- Return $\max(L, R, M)$
- Exercise: change one entry to make MSS=R.
  decrease one of -2, -1 by more than 3.
- How to find L, R, M?

MSS(A, left, right)
  if right $-$ left ≤ 2 **then**
    Solve directly and return MSS
  end if

  mid $= \lfloor \frac{\text{left+right}}{2} \rfloor$
  L = MSS(A, left, mid)
  R = MSS(A, mid+1, right)

  Set sum $= 0$ and $L' = 0$
  for i = mid down to 1 **do**
    sum $+= A[i]$
    $L' = \max(L', \text{sum})$
  end for

  Set sum $= 0$ and $R' = 0$
  for i mid+1 to right **do**
    sum $+= A[i]$
    $R' = \max(R', \text{sum})$
  end for
  $M = L' + R'$

  return $\max(L, R, M)$

▷ Base case

▷ Recurse on left and right halves

▷ Compute L' (left part of M)

▷ Compute R' (right part of M)

▷ Compute M

▷ Return max

MSS(A, left, right)
  if right $-$ left ≤ 2 **then**
    Solve directly and return MSS
  end if

  mid $= \lfloor \frac{\text{left+right}}{2} \rfloor$
  L = MSS(A, left, mid)
  R = MSS(A, mid+1, right)

  Set sum $= 0$ and $L' = 0$
  for i = mid down to 1 **do**
    sum $+= A[i]$
    $L' = \max(L', \text{sum})$
  end for

  Set sum $= 0$ and $R' = 0$
  for i mid+1 to right **do**
    sum $+= A[i]$
    $R' = \max(R', \text{sum})$
  end for
  $M = L' + R'$

  return $\max(L, R, M)$

Running time?

- Let $T(n)$ be running time of MSS on array of size n
- Two recursive calls on arrays of size $n/2$: $2T(n/2)$
- Work outside of recursive calls: $O(n)$
- Running time

$$T(n) = 2T(n/2) + O(n)$$

Recurrence

- Recurrence (with convenient base case)

$$T(n) = 2T(n/2) + O(n)$$
$$T(1) = O(1)$$

  could choose other convenient base case(s): T(0), T(2), ...
- Goal: *solve* the recurrence = find simple expression for $T(n)$
- First, let's use definition of Big-O:

$$T(n) \leq 2T(n/2) + cn$$
$$T(1) \leq c_1$$

- What next?

Solving the Recurrence

- Same recurrence with change of variable

$$T(m) \leq 2T(m/2) + cm, \quad m \geq 2$$
$$T(1) \leq c$$

  - no difference, but sometimes helpful conceptually
  - n = original input size, m = generic input size
- Three approaches to solve it
  1. Unrolling
  2. Recursion tree (another version of unrolling)
  3. Guess and verify (proof by induction)

Recurrence Solving (1): Unrolling

- **Idea 1**: "unroll" the recurrence

$$\begin{aligned}
T(n) &\leq 2T(n/2) + cn && \text{equation in } n \\
&\leq 2\left[2T(n/2^2) + c(n/2)\right] + cn && n \to n/2 \\
&= 2^2 T(n/2^2) + 2cn \\
&\leq 2^2\left[2T(n/2^3) + c(n/2^2)\right] + 2cn && n \to n/4 \\
&= 2^3 T(n/2^3) + 3cn \\
&\leq \dots
\end{aligned}$$

- Do you see a pattern? $2^k T(n/2^k) + k \cdot cn$
- When does this stop?
  Base case: $n/2^k = 1 \Rightarrow k = \log n$ unrollings

$$T(n) \leq 2^{\log n} \cdot T(1) + \log n \cdot cn = c_1 n + cn \log n = O(n \log n)$$

Clicker Question 2

Suppose we have the recurrence $T(n) = T(n/2) + T(n/3)$.
What do we get after two unrollings?

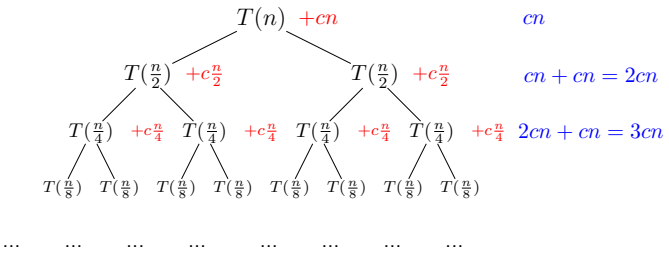
A. $T(n/4) + T(n/9)$

B. $T(n/4) + 2T(n/6) + T(n/9)$

C. $2T(n/6)$

D. $T(n/4) + T(n/6) + T(n/9)$

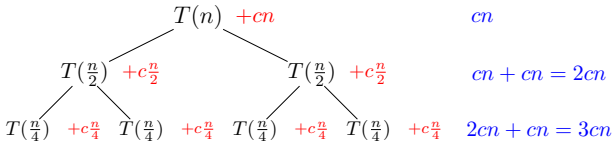
Recurrence Solving (2): Recursion Tree



... ...

- Each level adds work to expand nodes
- **Conclusion:** $T(n) \leq cn \log n$

Clicker Question 3



... ...

Suppose we have the recurrence $T(n) = 2T(n/2) + cn$.
What is the work done at each level of recursive calls?

A. cn

B. $2^k T(n/2^k)$

C. $2^k T(n/2^k) + cn$

D. cn, except $cn \log n$ for all the leaves.

Recurrence Solving (3): Guess and Verify

- Guess solution
- Prove by (strong) induction

$$T(n) \leq 2T(n/2) + cn$$

Guess solution $T(n) \leq kn \log n$, find k so it works.

Choose base case $n = 2 \Rightarrow T(2) \leq 2k \log 2$, $k \geq T(2)/2$.

$n = 1$ won't work as base case, since $\log 1 = 0$
(but small n does not matter for big-O)

Guess and Verify: Induction Step

Strong induction:
Assume $T(m) \leq k \cdot m \log m$ for all $m < n$, prove for n

$$\begin{aligned} T(n) &\leq 2 \cdot T(n/2) + cn \\ &\leq 2 \cdot k(n/2) \log(n/2) + cn \\ &= kn(\log n - 1) + cn \\ &= kn \log n + (c - k)n \qquad \leq kn \log n \text{ if } k \geq c \end{aligned}$$

$\Rightarrow$ choose $k = \max(c, T(2)/2)$.

The induction proof is complete, $T(n) \leq kn \log n$

A More General Recurrence

$$\boxed{T(n) \leq q \cdot T(n/2) + cn}$$

- What does the algorithm look like?
  - q recursive calls to itself on problems of half the size
  - $O(n)$ work outside of the recursive calls
- Exercises: $q = 1$, $q > 2$
- **Useful fact** (geometric sum): if $r \neq 1$ then

$$1 + r + r^2 + \ldots + r^d = \frac{1 - r^{d+1}}{1 - r} = \frac{r^{d+1} - 1}{r - 1}$$