CMPSCI 311: Introduction to Algorithms

Lecture 9: Divide and Conquer

Akshay Krishnamurthy

University of Massachusetts

Last Compiled: February 27, 2018

Announcements

- Midterm Wednesday 7-9pm ISB 135
- Homework 3 due next week
- No discussion this week, yes quiz
- Ibrahim's office hours change: Tuesday 12-1 CS 207
- HW 1 graded (submit regrade request with issues)

Recap

- Greedy algorithms
  - Schedule problems
  - Shortest paths (Dijkstra's algorithm)
  - MST (Prim, Kruskal)
    - Efficient implementation with union-find data structure.

Algorithm Design Techniques

- Greedy
- Divide and Conquer
- Dynamic Programming
- Network Flows

Comparison

	Greedy	Divide and Conquer
Formulate problem	?	?
Design algorithm	easy	hard
Prove correctness	hard	easy
Analyze running time	easy	hard

Divide and Conquer: Recipe

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

- Common example
  - Problem of size $n \rightarrow$ two parts of size $n/2$.
  - Combine solutions in $O(n)$ time.

Example: Mergesort

```
MergeSort(Arr)
  if length(Arr) ≤ 2 then                          ▷ Base case
     Sort however you like, return sorted list.
  else
     middle = length(Arr)/2                        ▷ Recursive Steps
     L = MergeSort(Arr[0:middle])
     R = MergeSort(Arr[middle:length(Arr)])
     Return Merge(L, R)                            ▷ Combine Step
  end if
```

Mergesort Running time

- Base Case: $O(1)$.
- Recursive step: $O(1)$ + ???
- Merge step: $O(n)$.

Recurrence Relations

Let $T(n)$ be running time for inputs of length n.

$$T(n) \leq 2T(n/2) + cn \qquad \text{when } n \geq 2$$
$$T(0), T(1), T(2) \leq c$$

How do we solve for $T(n)$?

Solving recurrences

$$T(n) \leq 2T(n/2) + cn, \qquad T(2) \leq c.$$

- Unravel recurrence
- Guess and check
- Partial substitution

Mergesort runtime: $O(n \log_2 n)$.

Maximum Subsequence Sum (MSS)

Input: array A of n numbers

Find: value of the largest subsequence sum

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

(Note: empty subsequence ($j < i$) is allowed and has sum zero)

What is a simple algorithm for MSS?

```
MSS(A)
  Initialize all entries of n × n array B to zero
  for i = 1 to n do
     sum = 0
     for j = i to n do
        sum += A[j]
        B[i, j] = sum
     end for
  end for
  Return maximum entry of 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

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

```
MSS(Arr)
  if length(Arr) == 1 then                                  ▷ Base case
      return max(A[0], 0)
  end if
  mid = length(Arr)/2
  L = MSS(Arr[0:mid]), R = MSS(Arr[mid:length(Arr)])        ▷ Recursive Steps
  Set sum = 0, L' = 0.                                      ▷ Compute Left
  for i = mid-1 down to 0 do
      sum += Arr[i], L' = max(L', sum).
  end for
  Set sum = 0, R' = 0.                                      ▷ Compute Right
  for i = mid up to length(Arr)-1 do
      sum += Arr[i], R' = max(R', sum).
  end for
  return max(L, R, L' + R').                                ▷ Output max
```

MSS Correctness?

- If MSS is contained in left half, then by induction we are correct
- If MSS is contained in right half, then by induction we are correct
- Otherwise MSS spans midpoint.
  - L' = MSS on left half ending at midpoint
  - R' = MSS on right half starting at midpoint
  - L'+R' = MSS spanning midpoint.

MSS running time

- Base Case: $O(1)$.
- Recursive step: $O(1)$ + ???
- Merge step: $O(n)$.

Recurrence:

$$T(n) \leq 2T(n/2) + cn, \qquad T(1) \leq c.$$

Solves to $O(n \log_2 n)$ just like Mergesort.

More recurrences

- Problem of size $n \to q$ parts of size $n/2$.
- Combine solutions in $O(n)$ time.

Recurrence

$$T(n) \leq qT(n/2) + cn, \qquad T(1) \leq c.$$

Qualitatively different behavior $q = 1$, $q = 2$, and $q > 2$.

- If $q = 1$, $T(n) = O(n)$.
- If $q = 2$, $T(n) = O(n \log n)$.
- If $q > 2$, $T(n) = O(n^{\log_2(q)})$.

Proof for $q = 1$

$$T(n) \leq T(n/2) + cn, \qquad T(1) \leq c.$$

- Unravel the recurrence

$$\begin{aligned}
T(n) &\leq T(n/2) + cn \\
&\leq T(n/4) + cn/2 + cn \\
&\leq T(n/8) + cn/4 + cn/2 + cn \\
&\dots \\
&\leq \sum_{i=0}^{\log_2 n - 1} cn/2^i \\
&\leq 2cn
\end{aligned}$$

Proof for $q = 1$

$$T(n) \leq T(n/2) + cn, \qquad T(1) \leq c.$$

- Partial substitution (with guess $T(n) \leq kn^d$)

$$\begin{aligned}
T(n) &\leq T(n/2) + cn \\
&\leq k(n/2)^d + cn \\
&= \frac{k}{2^d} n^d + cn
\end{aligned}$$

Set $d = 1$ $k = 2c$ to get

$$T(n) \leq \frac{k}{2} n + cn = kn$$

Proof for $q > 2$

$$T(n) \leq qT(n/2) + cn, \qquad T(1) \leq c.$$

- Unravel the recurrence

$$\begin{aligned} T(n) &\leq qT(n/2) + cn \\ &\leq q^2T(n/4) + cqn/2 + cn \\ &\leq q^3T(n/8) + cq^2n/4 + cqn/2 + cn \\ &\dots \\ &\leq \sum_{i=0}^{\log_2 n - 1} cn \left(\frac{q}{2}\right)^i \end{aligned}$$

Final calculations

- Use geometric series $(\sum_{k=0}^{n-1} r^k = (r^n - 1)/(r-1))$

$$\begin{aligned} T(n) &\leq cn \sum_{i=0}^{\log_2 n - 1} (q/2)^i = cn \left(\frac{(q/2)^{\log_2 n} - 1}{q/2 - 1} \right) \\ &\leq \frac{c}{q/2 - 1} n (q/2)^{\log_2 n} \\ &= \frac{c}{q/2 - 1} n n^{\log_2 (q/2)} \\ &= \frac{c}{q/2 - 1} n n^{\log_2 (q) - 1} \\ &= \frac{c}{q/2 - 1} n^{\log_2 (q)} = O(n^{\log_2 (q)}) \end{aligned}$$

Summary

With recurrence

$$T(n) \leq qT(n/2) + cn, \qquad T(1) \leq c.$$

Always get

$$T(n) \leq cn \sum_{i=0}^{\log_2 (n) - 1} (q/2)^i$$

- But series behaves differently for $q < 2, q = 2, q > 2$.