

CS3000: Algorithms & Data

Paul Hand

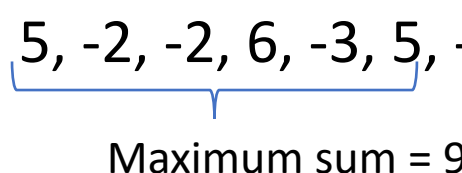
Lecture 7:

- Divide and Conquer Example – Similar to HW
- Binary Search
- Another Example

Jan 30, 2019

Practice Problem:
Maximum Sum Subarray Problem

Maximum Sum Subarray Problem

- **Input:** Array $A[1:n]$ of integers
- **Problem:** Find a subarray $A[i:j]$ with the largest possible sum
- **Example:** $A = [3, -4, 5, -2, -2, 6, -3, 5, -3, 2]$


Maximum sum = 9

- **Input:** Array $A[1:n]$ of integers
- **Problem:** Find a subarray $A[i:j]$ with the largest possible sum
- **Task:** Devise a divide and conquer algorithm to solve this problem. Consider an algorithm that divides A into two halves.
- **Task:** Devise a divide and conquer algorithm to solve this problem. Consider an algorithm that divides A into two halves.

Divide-and-Conquer: Binary Search

Binary Search

Is 28 in this list?

2	3	8	11	15	17	28	42
---	---	---	----	----	----	----	----

A

Binary Search

```
Search(A, t) :
```

```
  // A[1:n] sorted in ascending order
```

```
  Return BS(A, 1, n, t)
```

```
BS(A, ℓ, r, t) :
```

```
  If (ℓ > r) : return FALSE
```

$$m \leftarrow \ell + \left\lfloor \frac{r - \ell}{2} \right\rfloor$$

```
  If (A[m] = t) : Return m
```

```
  ElseIf (A[m] > t) : Return BS(A, ℓ, m-1, t)
```

```
  Else : Return BS(A, m+1, r, t)
```


Activity

- What is the running time of binary search?
 - What is the recurrence?
 - What is the solution to the recurrence?

```
Search(A, t) :  
  // A[1:n] sorted in ascending order  
  Return BS(A, 1, n, t)
```

```
BS(A, ℓ, r, t) :  
  If(ℓ > r) : return FALSE  
  
  m ← ℓ + ⌊ $\frac{r-\ell}{2}$ ⌋  
  
  If(A[m] = t) : Return m  
  ElseIf(A[m] > t) : Return BS(A, ℓ, m-1, t)  
  Else: Return BS(A, m+1, r, t)
```

Proof of Correctness for Binary Search

Proof of Correctness of Binary Search

Clm: $\forall n \in \mathbb{N} \quad \forall l, r \text{ s.t. } r-l \leq n, \forall A, \forall t$
$$\text{BS}(A, l, r, t) = \begin{cases} i \text{ s.t. } A[i] = t \\ \perp \text{ if } t \notin A \end{cases}$$
 $H(n)$
Inductive Hyp

Base Case: $H(0) \dots$
 $H(1)$ the algorithm is correct

Inductive Step: Assume $H(n)$ is true

Suppose that we get $\text{BS}(A, l, r, t)$ and $r-l \leq n+1$

$$m \leftarrow l + \lfloor \frac{r-l}{2} \rfloor$$

Search(A, t):

// A[1:n] sorted in ascending order

Return BS(A, 1, n, t)

BS(A, l, r, t):

If ($l > r$): return FALSE

$$m \leftarrow l + \lfloor \frac{r-l}{2} \rfloor$$

If ($A[m] = t$): Return m

ElseIf ($A[m] > t$): Return BS(A, l, m-1, t)

Else: Return BS(A, m+1, r, t)

Binary Search Wrapup

- Search a sorted array in time $O(\log n)$
- Divide-and-conquer approach
 - Find the middle of the list, recursively search half the list
 - **Key Fact:** eliminate half the list each time
- Prove correctness via induction
- Analyze running time via recurrence
 - $T(n) = T(n/2) + C$

Practice Problem:
Finding maximum of unimodal list

Max of Unimodal List

- **Input:** Array $A[1:n]$ of integers. $A[1:i]$ is increasing. And $A[i+1:n]$ is decreasing.
- **Problem:** Find largest element in $O(\log(n))$ time
- **Examples:**
 - $A = [1,4,5,3,0]$
 - $A = [5,2,1,0,-2]$
 - $A = [2,4,7,9]$

Max of Unimodal List

- **Input:** Array $A[1:n]$ of integers. $A[1:i]$ is increasing. And $A[i+1:n]$ is decreasing.
- **Problem:** Find largest element in $O(\log(n))$ time
- **Examples:**
 - $A = [1,4,5,3,0]$
 - $A = [5,2,1,0,-2]$
 - $A = [2,4,7,9]$