Algorithms

Dana Shapira
Lesson #1:
Divide and Conquer

Administration

- Grade:
  - exam – 90 % exercises – 10% (about 8)
- Email – Nethanel Gelernter: <nethanelbiu@gmail.com>
- Web site - http://u.cs.biu.ac.il/~gelernn/83222/
- Recommended Book:
  - Introduction to Algorithms - Cormen, Leiserson, Rivest and Stein (version 1 is without Stein).
  - אלגוריתמיקה . דוד הראל (האוניברסיטה הפתוחה)

Syllabus

- Divide and Conquer:
  recurrences, Boolean multiplication, Strassen's matrix multiplication, median and order statistics.
- Greedy algorithms:
  Huffman codes, more under graph algorithms.
- Dynamic programming:
  matrix-chain multiplication, longest common subsequence, edit distance, all-pairs shortest path (Floyd-Warshall).
- Graph algorithms:
  minimum spanning tree (Kruskal, Prim), shortest path (Dijkstra), maximum flow and minimum cuts (Ford-Fulkerson method, Edmonds-Karp),
- Polynomials and the FFT:
  representation of polynomials, evaluation and interpolation, (optional topic: chinese remainder theorem).
- Introduction to NP Completeness:
  Vertex Cover Problem, Clique Problem, Satisfiability Problem, Hitting Set
- [Additional optional topics: shortest path (Bellman-Ford), maximum bipartite matching ((a) using flow, (b) augmenting path method)

Divide and Conquer

Let A be an unsorted Array with n elements:

- Find the maximum element of A
  - Exact number of comparisons – n-1 (why?)
- Find the minimum element of A
- Find maximum and minimum element of A

MinMax(i,j)

```
MinMax(i,j){
if (j=i)
       return(A[i],A[i])
if (j=i+1) then
   if (A[i]<A[j]) then
       return(A[i],A[j])
   else return(A[j],A[i])
else {
```

$k \leftarrow \left\lfloor \frac{i+j}{2} \right\rfloor$

```
   (m1,M1)=MinMax(i,k)
   (m2,M2)=MinMax(k+1,j)
   return(min(m1,m2),max(M1,M2))
}
```

What is the number of comparisons?

$$T(n) = \begin{cases} 1 & n = 2 \\ 2T\left(\frac{n}{2}\right) + 2 & n > 2 \end{cases}$$

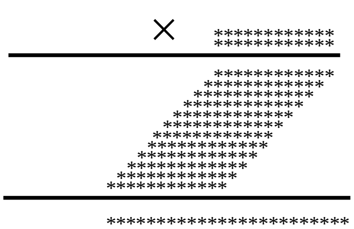
Boolean multiplication

X:

Y:

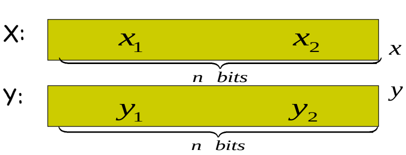
Question: How many bit operations needed for multiplying X and Y?

Boolean multiplication



$\Theta(n^2)$ bit operations

Boolean multiplication



$$x = x_2 + 2^{n/2} \cdot x_1$$

$$y = y_2 + 2^{n/2} \cdot y_1$$

Question: How many bit operations needed for multiplying X and Y?

Computing number of bit operations

$x = x_2 + 2^{n/2} \cdot x_1$

$y = y_2 + 2^{n/2} \cdot y_1$

$x \cdot y = x_2 y_2 + 2^{n/2}(x_1 y_2 + x_2 y_1) + 2^n \cdot (x_1 y_1)$

$$T(n) = \begin{cases} 1 & n = 1 \\ 4T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

Question: What is the number of bit operations?
Is it worth it?

Improvement

$A = x_1 y_1$

$B = x_2 y_2$

$C = (x_1 + x_2) \cdot (y_1 + y_2)$

$x \cdot y = x_2 y_2 + 2^{n/2}(x_1 y_2 + x_2 y_1) + 2^n \cdot (x_1 y_1) = B + 2^{n/2}(C - A - B) + 2^n \cdot A$

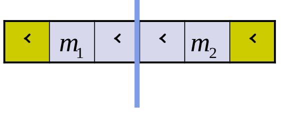
$$T(n) = \begin{cases} 1 & n = 1 \\ 3T\left(\frac{n}{2}\right) + c'n & n > 1 \end{cases}$$

Reminder - QuickSort

- Best Case
- Worst Case

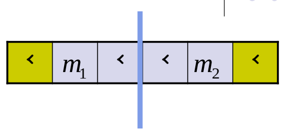
Median

- Problem: Given an unsorted array find its median
- Algorithms:
  1. Sort and return the n/2 element
  2. Divide and Conquer:
  - m_1 and m_2 are medians of A_1 and A_2 respectively

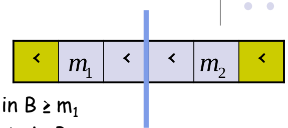


Median

```
Median(A){
  divide A into A1 and A2
  m1 ← Median(A1)
  m2 ← Median(A2)
  if (m1=m2)
     return m1
  if (m1<m2)
     Let B be the blue part of A
     m ← Median(B)
     return(m)
  else //    m1>m2
  …
  }
```



Median Algorithm Correctness



- **Claim 1:** $m_1 \le m \le m_2$
- Proof:
  - at least n/4 elements in B $\ge m_1$
  - $\Rightarrow$ At most n/4 elements in B $< m_1$
  - exactly n/4 elements in B < m
  - $\Rightarrow m_1 \le m$
- **Claim 2:** m is the median of A
  - n/4 elements greater than m in B
  - n/4 elements greater than m_2 in A
  - $\Rightarrow$n/2 elements greater than m in A

Running time

```
Median(A){
  divide A into A1 and A2
  m1 ← Median(A1)
  m2 ← Median(A2)
  if (m1=m2)
     return m1
  if (m1<m2)
     Let B be the blue part of A
     m ← Median(B)
     return(m)
  else //    m1>m2
  …
  }
```

What is the running time?

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T\left(\frac{n}{2}\right) + n & n > 1 \end{cases}$$

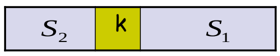
Corollary: Solution 1 is better…

Median

- Algorithms:
  1. Sort and return the n/2 element
  2. Divide and Conquer
  3. Define: FIND(A,t) returns the t-th element in A.
  - The median is FIND(A,n/2)

FIND(A,t)

```
FIND(A,t){
  choose a pivot k randomly
  Let:
```



$S_1 = \{x \in A | x > k\}$

$S_2 = \{x \in A | x < k\}$

```
  if |S1|=t-1
      return k
  else if |S1|>t-1
      FIND(S1,t)
  else // |S1|<t-1
      FIND(S2,t-|S1|-1)
}
```

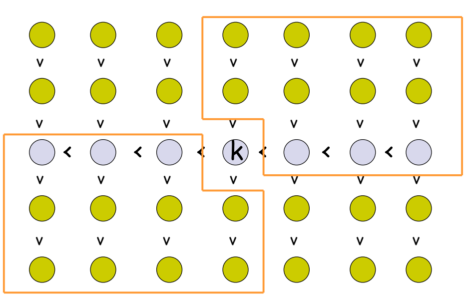
What is the running time in the worst case?

Controlling the pivot

```
FIND(A,t){
if |A|<50 sort A and return the t element
let b be a small odd number
divide A into groups of size b.
Sort each group
B={medians of the groups}
k ← FIND(B,n/2b)
let: S1 = {x ∈ A | x > k}
       S2 = {x ∈ A | x < k}
if |S1|=t-1
   return k
else if |S1|>t-1
   FIND(S1,t)
else // |S1|<t-1
   FIND(S2,t-|S1|-1)
}
```

What is the minimum number of comparisons needed for sorting 5 elements?

Running Time?



Running Time?

- n/b groups
- n/2b groups with medians < k
- Each such group has b/2 elements < median of group ⇒ < k
- ⇒ At least b/2·n/2b=n/4 elements < k
- ⇒ At most 3n/4 elements > k

$$T(n) = \begin{cases} n\log n & n < 50 \\ cn + T\left(\frac{n}{b}\right) + T\left(\frac{3}{4}n\right) & n \geq 50 \end{cases}$$

Running Time?

- Choose b=5

$$T(n)=\begin{cases} n\log n & n<50 \\ cn+T\left(\frac{n}{5}\right)+T\left(\frac{3}{4}n\right) & n\geq 50 \end{cases}$$

- **Claim:** FIND runs in linear time
  - **Proof:** There exists a constant d such that T(n)≤d·n.
  - Divide to n<50 and n≥50