

ECS 122A: Algorithm Design and Analysis

Week 3 Discussion

Ji Wang

Spring 2020

Outline

- ▶ Divide and Conquer: Key Idea
- ▶ Variant of Maximum Subarray Problem: Stock Investment
- ▶ Another Example: Binary Integer Multiplication

Divide and Conquer: Key Idea

1. **Divide** the problem into a number of subproblems that are smaller instances of the **same** problem.
2. **Conquer** by solving the subproblems **recursively**.
3. **Combine** the solutions to the subproblems to produce the solution to the original problem.

Divide and Conquer: Key Idea

1. **Divide** the problem into a number of subproblems that are smaller instances of the **same** problem.
2. **Conquer** by solving the subproblems **recursively**.
3. **Combine** the solutions to the subproblems to produce the solution to the original problem.

In class, we demonstrated three problems that use divide and conquer paradigm:

- ▶ Merge Sort
- ▶ Matrix Multiplication
- ▶ Maximum Subarray

Stock Investment

Problem Statement:

We're doing a simulation in which we look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, \dots, n$ and $p(i)$ is the price per share for the stock on that day. We want to know: When should we have bought and sold in order to have made as much money as possible?

For example, $n = 3$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5$, $p(4) = 3$. Then we should answer “buy on day 2, sell on day 3”.

Stock Investment

Problem Statement:

We're doing a simulation in which we look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, \dots, n$ and $p(i)$ is the price per share for the stock on that day. We want to know: When should we have bought and sold in order to have made as much money as possible?

For example, $n = 3$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5$, $p(4) = 3$. Then we should answer “buy on day 2, sell on day 3”.

Rephrase the problem:

Input: array p of length n

Stock Investment

Problem Statement:

We're doing a simulation in which we look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, \dots, n$ and $p(i)$ is the price per share for the stock on that day. We want to know: When should we have bought and sold in order to have made as much money as possible?

For example, $n = 3$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5$, $p(4) = 3$. Then we should answer “buy on day 2, sell on day 3”.

Rephrase the problem:

Input: array p of length n

Output: $\operatorname{argmax}\{p(j) - p(i)\}$ where $i \leq j$

Stock Investment

Revisit maximum subarray problem:

1. Divide $A[low \cdots high]$ into two subarrays of as equal size as possible by finding the midpoint mid
2. Conquer:
 - a. finding maximum subarrays of $A[low \cdots mid]$ and $A[mid + 1 \cdots high]$
 - b. finding a max-subarray that crosses the midpoint
3. Combine: returning the max of the three

Stock Investment

Revisit maximum subarray problem:

1. Divide $A[low \cdots high]$ into two subarrays of as equal size as possible by finding the midpoint mid
2. Conquer:
 - a. finding maximum subarrays of $A[low \cdots mid]$ and $A[mid + 1 \cdots high]$
 - b. finding a max-subarray that crosses the midpoint
3. Combine: returning the max of the three

Can we apply the same strategy on this problem?

Stock Investment

How does the conquer part work?

1. The optimal solution to $A[low \cdots mid]$
2. The optimal solution to $A[mid + 1 \cdots high]$
3. $\operatorname{argmax}\{p(j) - p(i)\}$ where $low \leq i \leq mid$ and $mid + 1 \leq j \leq high$

Stock Investment

How does the conquer part work?

1. The optimal solution to $A[low \cdots mid]$
2. The optimal solution to $A[mid + 1 \cdots high]$
3. $\operatorname{argmax}\{p(j) - p(i)\}$ where $low \leq i \leq mid$ and $mid + 1 \leq j \leq high$

Time complexity:

$$\begin{aligned}T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(n) + \Theta(1) \\ &= \Theta(n \log n)\end{aligned}$$

1. $2T(\frac{n}{2})$: the first two optimal solutions
2. $\Theta(n)$: the third solution is equivalent to find the min of $A[low \cdots mid]$ and max of $A[mid + 1 \cdots high]$
3. $\Theta(1)$: compare the results of three solutions

Binary Integer Multiplication

Recall how we multiply two integers of equal length

$$\begin{array}{r} 12 \\ \times 13 \\ \hline 36 \end{array}$$

$$\begin{array}{r} 12 \\ \times 12 \\ \hline 156 \end{array}$$

(a)

$$\begin{array}{r} 1100 \\ \times 1101 \\ \hline 1100 \\ 0000 \\ 1100 \\ \hline 10011100 \end{array}$$

(b)

Binary Integer Multiplication

Recall how we multiply two integers of equal length

	1100
	$\times 1101$
	<hr/>
	1100
	0000
	1100
	<hr/>
	10011100
	(b)
12	
$\times 13$	
<hr/>	
36	
12	
<hr/>	
156	
(a)	

Time complexity: $O(n^2)$

1. bit multiplication: n^2
2. bit addition: $O(n)$

Binary Integer Multiplication

Can we speedup? Think about Divide and Conquer

Suppose x and y are two n -bit integers, split each of them into left and right halves:

$$x = \boxed{x_L} \boxed{x_R} = 2^{\frac{n}{2}} x_L + x_R$$
$$y = \boxed{y_L} \boxed{y_R} = 2^{\frac{n}{2}} y_L + y_R$$

For instance, $10110110 = \boxed{1011} \boxed{0110} = 2^4 \times 1011 + 0110$

$$xy = (2^{\frac{n}{2}} x_L + x_R)(2^{\frac{n}{2}} y_L + y_R) = 2^n x_L x_L + 2^{\frac{n}{2}} (x_L y_R + x_R y_L) + x_R y_R$$

Binary Integer Multiplication

Can we speedup? Think about Divide and Conquer

Suppose x and y are two n -bit integers, split each of them into left and right halves:

$$x = \boxed{x_L} \boxed{x_R} = 2^{\frac{n}{2}} x_L + x_R$$
$$y = \boxed{y_L} \boxed{y_R} = 2^{\frac{n}{2}} y_L + y_R$$

For instance, $10110110 = \boxed{1011} \boxed{0110} = 2^4 \times 1011 + 0110$

$$xy = (2^{\frac{n}{2}} x_L + x_R)(2^{\frac{n}{2}} y_L + y_R) = 2^n x_L x_L + 2^{\frac{n}{2}} (x_L y_R + x_R y_L) + x_R y_R$$

Time complexity:

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) + \Theta(1)$$
$$= \Theta(n^2) \quad \leftarrow \text{No improvement!}$$

Binary Integer Multiplication

Previously, we wrote:

$$\begin{aligned}xy &= (2^{\frac{n}{2}}x_L + x_R)(2^{\frac{n}{2}}y_L + y_R) \\ &= 2^n \underline{x_L x_L} + 2^{\frac{n}{2}}(x_L y_R + x_R y_L) + \underline{x_R y_R}\end{aligned}$$

If we rewrite

$$x_L y_R + x_R y_L = \underline{(x_L + x_R)(y_L + y_R)} - x_L y_L - x_R y_R,$$

then the entire xy will only involve three multiplications, namely the underlining parts.

Binary Integer Multiplication

Previously, we wrote:

$$\begin{aligned}xy &= (2^{\frac{n}{2}}x_L + x_R)(2^{\frac{n}{2}}y_L + y_R) \\&= 2^n \underline{x_L x_L} + 2^{\frac{n}{2}}(x_L y_R + x_R y_L) + \underline{x_R y_R}\end{aligned}$$

If we rewrite

$$x_L y_R + x_R y_L = \underline{(x_L + x_R)(y_L + y_R)} - x_L y_L - x_R y_R,$$

then the entire xy will only involve three multiplications, namely the underlining parts.

Time complexity:

$$\begin{aligned}T(n) &= 3T\left(\frac{n}{2}\right) + \Theta(n) + \Theta(1) \\&= \Theta(n^{\log_2 3})\end{aligned}$$

Binary Integer Multiplication: Pseudocode

MULTIPLY(x, y)

```
1  //  $x, y$  are positive integers of  $n$ -bit
2  if  $n = 1$ 
3      return  $xy$ 
4  else
5       $x_L, x_R =$  leftmost  $\lceil n/2 \rceil$ , rightmost  $\lfloor n/2 \rfloor$  bits of  $x$ 
6       $y_L, y_R =$  leftmost  $\lceil n/2 \rceil$ , rightmost  $\lfloor n/2 \rfloor$  bits of  $y$ 
7       $p_1 = \text{MULTIPLY}(x_L, y_L)$ 
8       $p_2 = \text{MULTIPLY}(x_R, y_R)$ 
9       $p_3 = \text{MULTIPLY}(x_L + x_R, y_L + y_R)$ 
10     return  $p_1 \times 2^n + (p_3 - p_1 - p_2) \times 2^{\frac{n}{2}} + p_2$ 
```