ECS 122A: Algorithm Design and Analysis

Week 4 Discussion

© Ji Wang

Apr 21, 2021

Outline

- Binary Integer Multiplication (Homework 3 Problem 7)
- Variant of Maximum-Subarray Problem: Stock Investment

Binary Integer Multiplication

Recall how we multiply two integers of equal length $= n$

$$\begin{array}{r} 12 \\ \times\, 13 \\ \hline 36 \\ 12 \\ \hline 156 \end{array}$$

(a)

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

(b)

Binary Integer Multiplication

Recall how we multiply two integers of equal length $= n$

$$
\begin{array}{r}
12 \\
\times\, 13 \\
\hline
36 \\
12 \\
\hline
156
\end{array}
\qquad\qquad
\begin{array}{r}
1100 \\
\times\, 1101 \\
\hline
1100 \\
0000 \\
1100 \\
1100 \\
\hline
10011100
\end{array}
$$

(a) **(b)**

Time complexity: $O(n^2)$

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

Binary Integer Multiplication: Speedup

First, how do we represent a decimal integer digit-wise?

$$1024 = 10^3 \times 1 + 10^2 \times 0 + 10^1 \times 2 + 10^0 \times 4$$

Binary Integer Multiplication: Speedup

First, how do we represent a decimal integer digit-wise?

$$1024 = 10^3 \times 1 + 10^2 \times 0 + 10^1 \times 2 + 10^0 \times 4$$

Or, $1024 = \boxed{10}\ \boxed{24} = 10^2 \times 10 + 24$

Binary Integer Multiplication: Speedup

First, how do we represent a decimal integer digit-wise?

$$1024 = 10^3 \times 1 + 10^2 \times 0 + 10^1 \times 2 + 10^0 \times 4$$

Or, $1024 = \boxed{10}\ \boxed{24} = 10^2 \times 10 + 24$

Now, how about two n-bit binary integers?

$$x = \boxed{a}\ \boxed{b} = 2^{\frac{n}{2}} a + b$$
$$y = \boxed{c}\ \boxed{d} = 2^{\frac{n}{2}} c + d$$

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

$$\begin{aligned} xy &= (2^{\frac{n}{2}} a + b)(2^{\frac{n}{2}} c + d) \\ &= 2^n ac + 2^{\frac{n}{2}} (ad + bc) + bd \end{aligned}$$

Binary Integer Multiplication: Speedup

First, how do we represent a decimal integer digit-wise?

$$1024 = 10^3 \times 1 + 10^2 \times 0 + 10^1 \times 2 + 10^0 \times 4$$

Or, $1024 = \boxed{10}\ \boxed{24} = 10^2 \times 10 + 24$

Now, how about two n-bit binary integers?

$$x = \boxed{a}\ \boxed{b} = 2^{\frac{n}{2}} a + b$$
$$y = \boxed{c}\ \boxed{d} = 2^{\frac{n}{2}} c + d$$

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

$$\begin{aligned} xy &= (2^{\frac{n}{2}} a + b)(2^{\frac{n}{2}} c + d) \\ &= 2^n ac + 2^{\frac{n}{2}}(ad + bc) + bd \end{aligned}$$

Time complexity: [1]

$$\begin{aligned} T(n) &= 4T(\frac{n}{2}) + \Theta(n) + \Theta(1) \\ &= \Theta(n^2) \end{aligned} \quad \rightarrow \textbf{No improvement!}$$

[1] $2^n x$ is done by left shift with a constant time cost

Binary Integer Multiplication

Previously, we wrote:

$$\begin{aligned} xy &= (2^{\frac{n}{2}}a + b)(2^{\frac{n}{2}}c + d) \\ &= 2^n \underline{ac} + 2^{\frac{n}{2}}(ad + bc) + \underline{bd} \end{aligned}$$

If we rewrite

$$ad + bc = \underline{(a+b)(c+d)} - ac - bd,$$

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

Binary Integer Multiplication

Previously, we wrote:

$$\begin{aligned} xy &= (2^{\frac{n}{2}}a + b)(2^{\frac{n}{2}}c + d) \\ &= 2^n \underline{ac} + 2^{\frac{n}{2}}(ad + bc) + \underline{bd} \end{aligned}$$

If we rewrite

$$ad + bc = \underline{(a+b)(c+d)} - ac - bd,$$

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

Time complexity:

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

Binary Integer Multiplication: Pseudocode

MULTIPLY(x, y)

// x, y are positive integers of n-bit, assuming n is even
if $n = 1$
    return xy
else
    $a, b =$ leftmost $n/2$, rightmost $n/2$ bits of x
    $c, d =$ leftmost $n/2$, rightmost $n/2$ bits of y
    $p_1 =$ MULTIPLY(a, b)
    $p_2 =$ MULTIPLY(c, d)
    $p_3 =$ MULTIPLY$(a + b, c + d)$
    return $p_1 \times 2^n + (p_3 - p_1 - p_2) \times 2^{\frac{n}{2}} + p_2$

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 $i = 1, 2, ..., n$ 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? (If there was no way to make money during the n days, we should report this instead.)

For example, $n = 4$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5, p(4) = 3$. 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 $i = 1, 2, ..., n$ 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? (If there was no way to make money during the n days, we should report this instead.)

For example, $n = 4$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5, p(4) = 3$. 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 $i = 1, 2, ..., n$ 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? (If there was no way to make money during the n days, we should report this instead.)

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

Rephrase the problem:

Input: array p of length n

Output: $\texttt{argmax}\{p(j) - p(i)\}$ 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 $p[low \cdots mid]$
2. The optimal solution to $p[mid + 1 \cdots high]$
3. $\texttt{argmax}\{p(j) - p(i)\}$ 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 $p[low \cdots mid]$
2. The optimal solution to $p[mid + 1 \cdots high]$
3. $\texttt{argmax}\{p(j) - p(i)\}$ where $low \leq i \leq mid$ and $mid + 1 \leq j \leq high$ (equivalent to finding the index of `min` of $p[low \cdots mid]$ and that of `max` of $p[mid + 1 \cdots high]$)

Stock Investment

How does the conquer part work?

1. The optimal solution to $p[low \cdots mid]$
2. The optimal solution to $p[mid + 1 \cdots high]$
3. $\texttt{argmax}\{p(j) - p(i)\}$ where $low \leq i \leq mid$ and $mid + 1 \leq j \leq high$ (equivalent to finding the index of `min` of $p[low \cdots mid]$ and that of `max` of $p[mid + 1 \cdots high]$)

How to describe the design of an algorithm in English

- First point out what strategy/method used, e.g. divide-and-conquer, binary search.
- Separate paragraphs if necessary, e.g. branches.
- Bullet-point format is also a good practice.

Stock Investment: Pseudocode

```
STOCK-INVESTMENT(p, low, high)
// Base case: only one element
if low == high
    return low, high, 0
else mid = low + ⌊(high − low)/2⌋
    leftBuy, leftSell, leftGain = STOCK-INVESTMENT(p, low, mid)
    rightBuy, rightSell, rightGain = STOCK-INVESTMENT(p, mid, high)
    // Find the index of min(leftArray)
    crossBuy = MIN-INDEX(p, low, mid)
    // Find the index of max(rightArray)
    crossSell = MAX-INDEX(p, mid + 1, high)
    crossGain = p[crossSell] − p[crossBuy]
    if max(leftGain, rightGain, crossGain) ≤ 0
        return "no gain"
    elseif leftGain ≥ rightGain and leftGain ≥ crossGain
        return leftBuy, leftSell, leftGain
    elseif rightGain ≥ leftGain and rightGain ≥ crossGain
        return rightBuy, rightSell, rightGain
    else return crossBuy, crossSell, crossGain
```

Pseudocode Conventions [2]

1. Pseudocode should be self-explanatory, only a few comments might be needed.
2. Indentation/`end` indicates block structure (curly braces are not good practice).
3. Array index starts at 1 and ends at n, i.e. $A[i]$ is the ith element of the array A.
4. Use subroutine if the main procedure is too long, e.g. `merge`, `MaxXingSubarray`

---

[2] See pp.[20-22] in the textbook for pseudocode conventions

Stock Investment: Time complexity

$$T(n) = 2T(\frac{n}{2}) + \Theta(n) + \Theta(1)$$
$$= \Theta(n \log n)$$

1. $2T(\frac{n}{2})$: the first two optimal solutions
2. $\Theta(n)$: the third solution is equivalent to finding the `min` and `max`, e.g. linear scan
3. $\Theta(1)$: compare the results of three solutions