CSCI 2200 FOUNDATIONS OF COMPUTER SCIENCE

David Goldschmidt
goldsd3@rpi.edu
Fall 2022

MAXIMUM-SUBSTRING-SUM PROBLEM

Given a sequence of numbers: $a_1 \quad a_2 \quad a_3 \quad a_4 \quad \ldots \quad a_{n-1} \quad a_n$

Determine the substring with the maximum sum $\sum_{k=i}^{j} a_k$

Without loss of generality, assume each $a_k \in \mathbb{Z}$

e.g., what is the maximum substring sum of the following sequence?

1 -1 -1 2 3 4 -1 -1 2 3 -4 1 2 -1 -2 1

HINT: the sum is 12…

MAXIMUM-SUBSTRING-SUM PROBLEM

Without loss of generality, assume each $a_k \in \mathbb{Z}$

Given a sequence of numbers: $a_1 \quad a_2 \quad a_3 \quad a_4 \quad \ldots \quad a_{n-1} \quad a_n$

Determine the substring with the maximum sum $\sum_{k=i}^{j} a_k$

e.g., what is the maximum substring sum of the following sequence?

$$1 \quad -1 \quad -1 \quad \underbrace{2 \quad 3 \quad 4 \quad -1 \quad -1 \quad 2 \quad 3}_{\text{maximum substring sum is 12}} \quad -4 \quad 1 \quad 2 \quad -1 \quad -2 \quad 1$$

One approach is to *exhaustively* calculate sums for all possible substrings…

…which would require <u>three</u> nested loops, i.e., <u>three</u> loop variables *i*, *j*, *k*

Write pseudocode for this "brute force" exhaustive approach…

MAXIMUM-SUBSTRING-SUM PROBLEM

Without loss of generality, assume each $a_k \in \mathbb{Z}$

Given a sequence of numbers: $a_1 \quad a_2 \quad a_3 \quad a_4 \quad \ldots \quad a_{n-1} \quad a_n$

Determine the substring with the maximum sum $\sum_{k=i}^{j} a_k$

Algorithm 1 *exhaustively* calculates sums for all possible substrings:

```
MaxSum ← 0;
for i, j = 1 to n do
   CurSum ← 0;
   for k = i to j do
      CurSum ← CurSum + a_k;
   MaxSum ← max(CurSum, MaxSum);
return MaxSum;
```

We can express the runtime as:

$$T_1(n) = 2 + \sum_{i=1}^{n}\left[2 + \sum_{j=i}^{n}\left(5 + \sum_{k=i}^{j} 2\right)\right]$$

MAXIMUM-SUBSTRING-SUM PROBLEM

Without loss of generality, assume each $a_k \in \mathbb{Z}$

Given a sequence of numbers: $a_1 \quad a_2 \quad a_3 \quad a_4 \quad \dots \quad a_{n-1} \quad a_n$

Determine the substring with the maximum sum $\sum_{k=i}^{j} a_k$

Given multiple algorithms that solve this problem…

Which algorithm is best…?

We base the runtime on the size of the input, n

See Problems 9.69-9.72…

$$T_1(n) = 2 + \sum_{i=1}^{n}\left[2 + \sum_{j=i}^{n}\left(5 + \sum_{k=i}^{j} 2\right)\right].$$

$$T_2(n) = 2 + \sum_{i=1}^{n}\left(3 + \sum_{j=i}^{n} 6\right).$$

$$T_3(n) = \begin{cases} 3 & n = 1; \\ 2T_3(\frac{1}{2}n) + 6n + 9 & n > 1 \text{ and even}; \\ T_3(\frac{1}{2}(n+1)) + T_3(\frac{1}{2}(n-1)) + 6n + 9 & n > 1 \text{ and odd}. \end{cases}$$

$$T_4(n) = 5 + \sum_{i=1}^{n} 10.$$

MAXIMUM-SUBSTRING-SUM

$$T_1(n) = 2 + \sum_{i=1}^{n}\left[2 + \sum_{j=i}^{n}\left(5 + \sum_{k=i}^{j} 2\right)\right].$$

$$T_2(n) = 2 + \sum_{i=1}^{n}\left(3 + \sum_{j=i}^{n} 6\right).$$

$$T_3(n) = \begin{cases} 3 & n = 1; \\ 2T_3(\frac{1}{2}n) + 6n + 9 & n > 1 \text{ and even}; \\ T_3(\frac{1}{2}(n+1)) + T_3(\frac{1}{2}(n-1)) + 6n + 9 & n > 1 \text{ and odd}. \end{cases}$$

$$T_4(n) = 5 + \sum_{i=1}^{n} 10.$$

Tinker with small values of input size n…

n	1	2	3	4	5	6	7	8	9	10	…	
$T_1(n)$	11	29	58	100	157	231	324	438	575	737	…	✗
$T_2(n)$	11	26	47	74	107	146	191	242	299	362	…	✓
$T_3(n)$	3	27	57	87	123	159	195	231	273	315	…	✓
$T_4(n)$	15	25	35	45	55	65	75	85	95	105	…	✓✓✓

?

We need simpler forms for expressing $T_1(n)$, $T_2(n)$, $T_3(n)$, and $T_4(n)$…

SUMS – CONSTANT RULE

The *index of summation* is *i*, but everything else is a constant…

Given the summations below, we can simplify and get rid of the Σ…

$$S_1 = \sum_{i=1}^{10} 3 = 3+3+3+3+3+3+3+3+3+3 \qquad 3 \times 10$$

$$S_2 = \sum_{i=1}^{10} j = j+j+j+j+j+j+j+j+j+j \qquad j \times 10$$

$$S_3 = \sum_{i=1}^{10} i = 1+2+3+4+5+6+7+8+9+10 \qquad \tfrac{1}{2} \times 10 \times (10+1)$$

Constants can be moved outside of the summation, e.g.,

$$S_1 = \sum_{i=1}^{10} 3 = 3\sum_{i=1}^{10} 1 = 3 \times 10 \qquad S_2 = \sum_{i=1}^{10} j = j\sum_{i=1}^{10} 1 = j \times 10.$$

SUMS – ADDITION RULE

The sum of terms added together is the addition of the individual sums.

$$\sum_i (a(i) + b(i) + c(i) + \cdots) = \sum_i a(i) + \sum_i b(i) + \sum_i c(i) + \cdots$$

e.g.,

$$\begin{aligned} S &= \sum_{i=1}^{5} (i + i^2) \\ &= (1+1^2) + (2+2^2) + (3+3^2) + (4+4^2) + (5+5^2) \\ &= (1+2+3+4+5) + (1^2+2^2+3^2+4^2+5^2) \\ &= \sum_{i=1}^{5} i + \sum_{i=1}^{5} i^2. \end{aligned}$$

SUMS – COMMON SUMS

Prove using induction…!

Common Sums. Prove these sums by induction on n. Please do it!

1. $\sum_{i=k}^{n} 1 = n+1-k$
2. $\sum_{i=1}^{n} f(x) = nf(x)$
3. $\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r} \quad (r \neq 1)$
4. $\sum_{i=1}^{n} i = n(n+1)/2$
5. $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$
6. $\sum_{i=1}^{n} i^3 = n^2(n+1)^2/4$
7. $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$
8. $\sum_{i=0}^{n} \frac{1}{2^i} = 2 - \frac{1}{2^n}$
9. $\sum_{i=1}^{n} \log i = \log n!$

e.g., simplify the following sum: $\sum_{i=1}^{n}(1+2i+2^{i+2})$

SUMS – COMMON SUMS

Common Sums. Prove these sums by induction on n. Please do it!

1. $\sum_{i=k}^{n} 1 = n+1-k$
2. $\sum_{i=1}^{n} f(x) = nf(x)$
3. $\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r} \quad (r \neq 1)$
4. $\sum_{i=1}^{n} i = n(n+1)/2$
5. $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$
6. $\sum_{i=1}^{n} i^3 = n^2(n+1)^2/4$
7. $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$
8. $\sum_{i=0}^{n} \frac{1}{2^i} = 2 - \frac{1}{2^n}$
9. $\sum_{i=1}^{n} \log i = \log n!$

e.g., simplify the following sum: $\sum_{i=1}^{n}(1+2i+2^{i+2})$

$$
\begin{aligned}
\sum_{i=1}^{n}(1+2i+2^{i+2}) &= \sum_{i=1}^{n}1+\sum_{i=1}^{n}2i+\sum_{i=1}^{n}2^{i+2} && \text{addition rule}\\
&= \sum_{i=1}^{n}1+2\sum_{i=1}^{n}i+4\sum_{i=1}^{n}2^{i} && \text{constant rule}\\
&= n+n(n+1)+4\cdot(2^{n+1}-1-1) && \text{common sums and rearranging}\\
&= n+n(n+1)+2^{n+3}-8
\end{aligned}
$$

SUMS – NESTED SUM RULE

To compute a nested sum, start with the innermost sum and proceed outward.

e.g.,
$$S_1 = \sum_{i=1}^{3}\sum_{j=1}^{3} 1 = \underbrace{\sum_{j=1}^{3} 1}_{(i=1)} + \underbrace{\sum_{j=1}^{3} 1}_{(i=2)} + \underbrace{\sum_{j=1}^{3} 1}_{(i=3)} = 3+3+3 = 9$$

e.g.,
$$S_2 = \sum_{i=1}^{3}\sum_{j=1}^{i} 1 = \underbrace{\sum_{j=1}^{1} 1}_{(i=1)} + \underbrace{\sum_{j=1}^{2} 1}_{(i=2)} + \underbrace{\sum_{j=1}^{3} 1}_{(i=3)} = 1+2+3 = 6$$

(or more generally…)

$$S(n) = \sum_{i=1}^{n}\sum_{j=1}^{i} 1 = \sum_{i=1}^{n}\underbrace{\sum_{j=1}^{i} 1}_{f(i)=i} = \sum_{i=1}^{n} i = \tfrac{1}{2}n(n+1)$$

MAXIMUM-SUBSTRING-SUM PROBLEM

Returning to our runtime problem, compute formulas for $T_1(n)$, $T_2(n)$, and $T_4(n)$…

$$T_1(n) = 2 + \sum_{i=1}^{n}\left[2 + \sum_{j=i}^{n}\left(5 + \sum_{k=i}^{j} 2\right)\right].$$

$$T_2(n) = 2 + \sum_{i=1}^{n}\left(3 + \sum_{j=i}^{n} 6\right).$$

$$T_3(n) = \begin{cases} 3 & n = 1; \\ 2T_3(\tfrac{1}{2}n) + 6n + 9 & n > 1 \text{ and even}; \\ T_3(\tfrac{1}{2}(n+1)) + T_3(\tfrac{1}{2}(n-1)) + 6n + 9 & n > 1 \text{ and odd}. \end{cases}$$

$$T_4(n) = 5 + \sum_{i=1}^{n} 10.$$

Simplify $T_2(n)$…

$$
\begin{aligned}
2+\sum_{i=1}^{n}\Big(3+\sum_{j=i}^{n}6\Big) &= 2+\sum_{i=1}^{n}3+\sum_{i=1}^{n}\sum_{j=i}^{n}6 && \text{addition rule}\\
&= 2+3\sum_{i=1}^{n}1+\sum_{i=1}^{n}\sum_{j=i}^{n}6 && \text{constant rule}\\
&= 2+3n+\sum_{i=1}^{n}\sum_{j=i}^{n}6 && \text{common sum}\\
&= 2+3n+\sum_{i=1}^{n}\sum_{j=i}^{n}6 && \text{innermost sum}\\
&= 2+3n+6\sum_{i=1}^{n}\sum_{j=i}^{n}1 && \text{constant rule}\\
&= 2+3n+6\sum_{i=1}^{n}(n+1-i) && \text{common sum}\\
&= 2+3n+6(n+(n-1)+\cdots+1)\\
&= 2+3n+6\times\tfrac{1}{2}n(n+1) && \text{common sum}\\
&= 2+6n+3n^2 && \text{algebra}
\end{aligned}
$$

Simplify this nested sum…

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{i}ij &= \sum_{i=1}^{n}\sum_{j=1}^{i}ij && \text{innermost sum}\\
&= \sum_{i=1}^{n}i\sum_{j=1}^{i}j && \text{constant rule}\\
&= \sum_{i=1}^{n}i\times\tfrac{1}{2}i(i+1) && \text{common sum}\\
&= \tfrac{1}{2}\sum_{i=1}^{n}(i^3+i^2) && \text{constant rule \& algebra}\\
&= \tfrac{1}{2}\sum_{i=1}^{n}i^3+\tfrac{1}{2}\sum_{i=1}^{n}i^2 && \text{addition rule}\\
&= \tfrac{1}{8}n^2(n+1)^2+\tfrac{1}{12}n(n+1)(2n+1) && \text{common sums}\\
&= \tfrac{1}{12}n+\tfrac{3}{8}n^2+\tfrac{5}{12}n^3+\tfrac{1}{8}n^4 && \text{algebra}
\end{aligned}
$$

CATEGORIZING ALGORITHM RUNTIMES

Simplified formulas for our algorithms…

$$T_1(n) = 2 + \tfrac{31}{6}n + \tfrac{7}{2}n^2 + \tfrac{1}{3}n^3$$
$$T_2(n) = 2 + 6n + 3n^2$$
$$3n(\log_2 n + 1) - 9 \le T_3(n) \le 12n(\log_2 n + 3) - 9$$
$$T_4(n) = 5 + 10n$$

What happens when input size n grows?

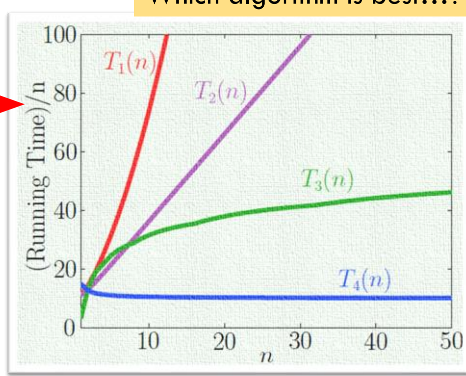
$$T_1(n) = 2 + \sum_{i=1}^{n}\Big[2 + \sum_{j=i}^{n}\Big(5 + \sum_{k=i}^{j} 2\Big)\Big].$$
$$T_2(n) = 2 + \sum_{i=1}^{n}\Big(3 + \sum_{j=i}^{n} 6\Big).$$
$$T_3(n) = \begin{cases} 3 & n = 1; \\ 2T_3(\frac{1}{2}n) + 6n + 9 & n > 1 \text{ and even}; \\ T_3(\frac{1}{2}(n+1)) + T_3(\frac{1}{2}(n-1)) + 6n + 9 & n > 1 \text{ and odd}. \end{cases}$$
$$T_4(n) = 5 + \sum_{i=1}^{n} 10.$$

$T_3(n)$ is very difficult…see Problem 9.71…

CATEGORIZING ALGORITHM RUNTIMES

Which algorithm is best…?

We want to know how each algorithm scales with input size n, so divide each runtime formula by n…



$$T_1(n) = 2 + \tfrac{31}{6}n + \tfrac{7}{2}n^2 + \tfrac{1}{3}n^3$$
$$T_2(n) = 2 + 6n + 3n^2$$
$$3n(\log_2 n + 1) - 9 \le T_3(n) \le 12n(\log_2 n + 3) - 9$$
$$T_4(n) = 5 + 10n$$

What if we further improve $T_4(n) = 50 + 8n$…?

CATEGORIZING ALGORITHM RUNTIMES

```
               T (n)
 lim            4             lim     5 + 10n
         ------  =             ---------
n->inf          n             n->inf      n

                lim      5     10n
       =               --- + -----   =  10
              n->inf     n      n
```

(in-lecture notes)

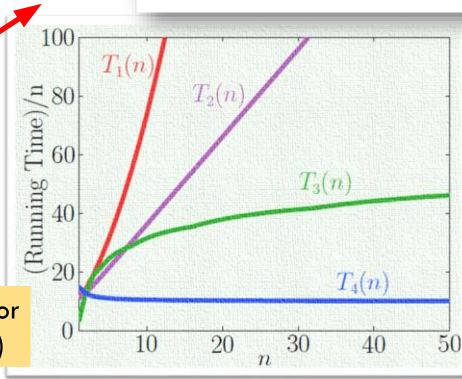
We focus on *scaling up*, i.e., when input size n grows very large — when $n \to \infty$

Algorithm 4 is *linear* in n…

…as $n \to \infty$, $\frac{T_4(n)}{n} \to$ c (a constant)

Therefore, we state that $T_4(n) \in \Theta(n)$

Try doing this for $T_1(n)$ and $T_2(n)$



$\boldsymbol{T_1(n)} = 2 + \frac{31}{6}n + \frac{7}{2}n^2 + \frac{1}{3}n^3$
$\boldsymbol{T_2(n)} = 2 + 6n + 3n^2$
$3n(\log_2 n + 1) - 9 \le \boldsymbol{T_3(n)} \le 12n(\log_2 n + 3) - 9$
$\boldsymbol{T_4(n)} = 5 + 10n$

We categorize algorithms based on *growth rates* of their runtimes, which makes it easier for us to describe them in comparison with one another

ASYMPTOTICALLY LINEAR FUNCTIONS – Θ(n)

Recurrence $T \in \Theta(n)$ if there are positive constants c and C such that…

$$c \cdot n \le T(n) \le C \cdot n$$

As n grows toward ∞, dividing T by n will give us 0, a constant, or ∞

$$\frac{T(n)}{n} \xrightarrow[n\to\infty]{} \begin{cases} \infty & T \in \omega(n), \quad \text{“}T > n\text{”}; \\ \text{constant}>0 & T \in \Theta(n), \quad \text{“}T = n\text{”}; \\ 0 & T \in o(n), \quad \text{“}T < n\text{”}. \end{cases}$$

little-omega-of-*n*
big-theta-of-*n*
little-oh-of-*n*

ASYMPTOTICALLY LINEAR FUNCTIONS – $\Theta(n)$

Example functions that are *asymptotically* linear, i.e., that are in $\Theta(n)$…

$2n + 7$ $30n + 10^{100}$ $2n + 15\sqrt{n}$ $10^9 n + 3$ $2n + \log n$

Functions that are not asymptotically linear, i.e., that are not in $\Theta(n)$…

$10^{-9}n^2$ $n^{1.0001}$ $n^{0.9999}$ $n \log n$ 2^n

How do we know if $T(n) \in \Theta(n)$…?

ASYMPTOTICALLY LINEAR FUNCTIONS – $\Theta(n)$

Example functions that are *asymptotically* linear, i.e., that are in $\Theta(n)$…

$2n + 7$ $30n + 10^{100}$ $2n + 15\sqrt{n}$ $10^9 n + 3$ $2n + \log n$

Functions that are not asymptotically linear, i.e., that are not in $\Theta(n)$…

$10^{-9}n^2$ $n^{1.0001}$ $n^{0.9999}$ $n \log n$ 2^n

also see if you can determine constants c and C for those that are asymptotically linear

How do we know if $T(n) \in \Theta(n)$…? Divide by n and then take the limit to ∞

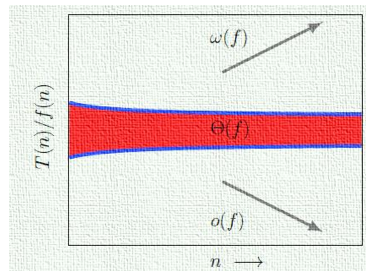
We can generalize this to any function $f(n)$ — not just the linear $f(n) = n$

GENERAL ASYMPTOTIC FUNCTIONS

We can generalize this to any function $f(n)$ — not just the linear $f(n) = n$

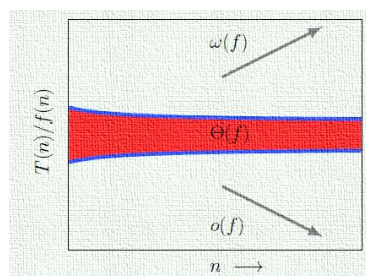
As n grows toward ∞, dividing T by $f(n)$ will again give us 0, a constant, or ∞

$$\frac{T(n)}{f(n)} \xrightarrow[n\to\infty]{} \begin{cases} \infty & T \in \omega(f),\ \text{“}T > f\text{”}; \\ \text{constant}>0 & T \in \Theta(f),\ \text{“}T = f\text{”}; \\ 0 & T \in o(f),\ \text{“}T < f\text{”}. \end{cases}$$



GENERAL ASYMPTOTIC FUNCTIONS

$$\frac{T(n)}{f(n)} \xrightarrow[n\to\infty]{} \begin{cases} \infty & T \in \omega(f),\ \text{“}T > f\text{”}; \\ \text{constant}>0 & T \in \Theta(f),\ \text{“}T = f\text{”}; \\ 0 & T \in o(f),\ \text{“}T < f\text{”}. \end{cases}$$



$T \in o(f)$	$T \in O(f)$	$T \in \Theta(f)$	$T \in \Omega(f)$	$T \in \omega(f)$
“$T < f$”	“$T \le f$”	“$T = f$”	“$T \ge f$”	“$T > f$”
	$T(n) \le Cf(n)$	$cf(n) \le T(n) \le Cf(n)$	$cf(n) \le T(n)$	

FREQUENTLY OCCURRING GROWTH RATES

Runtimes that are reasonable…

log	linear	loglinear	quadratic	cubic
$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^3)$

best → worst

Runtimes that are unreasonable…

superpolynomial	exponential	factorial	forget it…
$\Theta(n^{\log n})$	$\Theta(2^n)$	$\Theta(n!)$	$\Theta(n^n)$

→ worst

TRICKS TO DETERMINING GROWTH RATE

For polynomials, focus on the highest order term to determine the growth rate…

$2n^2$	$n^2 + n\sqrt{n}$	$n^2 + \log^{256} n$	$n^2 + n^{1.99} \log^{256} n$
$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$

Divide by n^2 and take the limit to ∞ to verify…

For summations, the growth rate is the number of *nestings* plus the order of the *summand*…

$\sum_{i=1}^{n} i$	$\sum_{i=1}^{n} \sum_{j=1}^{i} 1$	$\sum_{i=1}^{n} \sum_{j=1}^{i} ij$
$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^4)$

Remove the summations by determining an equivalent $f(n)$, then divide by n^2 and take the limit to ∞ to verify…

(or n^4)

WHAT NEXT…?

Grade inquiries for Exam 1 and Homeworks 1 and 2 due by 11:59PM on October 21

Problem Set 5 will be posted by Monday, October 24…
- …and is due in your October 26 recitations

Practice! Tinker! Practice! Tinker! Practice! Tinker! Practice! Tinker! Practice!