Compare the growth rate of functions

- We have two algorithms A_1 and A_2 for solving the same problem, with runtime functions $T_1(n)$ and $T_2(n)$, respectively. Which algorithm is more efficient?
- We compare the growth rate of $T_1(n)$ and $T_2(n)$.
- If $T_1(n) = \Theta(T_2(n))$, then the efficiency of the two algorithms are about the same (when n is large).
- If $T_1(n) = o(T_2(n))$, then the efficiency of the algorithm A_1 will be better than that of algorithm A_2 (when n is large).
- By using the definitions, we can directly show whether $T_1(n) = O(T_2(n))$, or $T_1(n) = \Omega(T_2(n))$. However, it is not easy to prove the relationship of two functions in this way.

Limit Test

Limit Test is a powerful method for comparing functions.

Limit Test

Let $T_1(n)$ and $T_2(n)$ be two functions. Let $c = \lim_{n\to\infty} \frac{T_1(n)}{T_2(n)}$.

1. If c is a constant > 0, then $T_1(n) = \Theta(T_2(n))$.
2. If $c = 0$, then $T_1(n) = o(T_2(n))$.
3. If $c = \infty$, then $T_1(n) = \omega(T_2(n))$.
4. If c does not exists (or if we do not know how to compute c), the limit test fails.

Proof of (1): $c = \lim_{n\to\infty} \frac{T_1(n)}{T_2(n)}$ means: $\forall \epsilon > 0$, there exists $n_0 \geq 0$ such that for any $n \geq n_0$: $\left|\frac{T_1(n)}{T_2(n)} - c\right| \leq \epsilon$; or equivalently: $c - \epsilon \leq \frac{T_1(n)}{T_2(n)} \leq c + \epsilon$. Let $\epsilon = c/2$ and let $c_1 = c - \epsilon = c/2$ and $c_2 = c + \epsilon = 3c/2$, we have

$$c_1 T_2(n) \leq T_1(n) \leq c_2 T_2(n)$$

for all $n \geq n_0$. Thus $T_1(n) = \Theta(T_2(n))$ by definition.

Example

Example 1

$$T_1(n) = 10n^2 + 15n - 60, \quad T_2(n) = n^2$$

$$\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = \lim_{n\to\infty} \frac{10n^2 + 15n - 60}{n^2} = \lim_{n\to\infty} (10 + \frac{15}{n} - \frac{60}{n^2}) = 10 + 0 - 0 = 10$$

Since 10 is a constant > 0, we have $T_1(n) = \Theta(T_2(n)) = \Theta(n^2)$ by the statement 1 of Limit Test (as expected).

Log function

The log functions are very useful in algorithm analysis.

$$\lg = \log_2 n$$

$$\log n = \log_{10} n$$

$$\ln n = \log_e n$$

($\ln n$ is the log function with the natural base $e = 2.71828\ldots$).

Log base change formula

Log base change formula

For any $1 < a, b$, $\log_b n = \frac{\log_a n}{\log_a b} = \log_b a \cdot \log_a n$.

Proof: Let $k = \log_b n$. By definition: $n = b^k$.
Take $\log_a$ on both sides: $\log_a n = \log_a(b^k) = k \cdot \log_a b$
This implies: $\log_b n = k = \frac{\log_a n}{\log_a b}$.

Let $n = a$ in this formula and note $1 = \log_a a$:

$$\log_b a = \frac{\log_a a}{\log_a b} = \frac{1}{\log_a b}$$

This proves the second part of the formula.

L'Hospital Rule

L'Hospital Rule

- If $\lim_{n\to\infty} f(n) = 0$ and $\lim_{n\to\infty} g(n) = 0$, then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

- If $\lim_{n\to\infty} f(n) = \infty$ and $\lim_{n\to\infty} g(n) = \infty$, then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

Example

Example 2

$T_1(n) = n^2 + 6$, $T_2(n) = n \lg n$. (Recall: $\lg n = \log_2 n$.)

$$\begin{aligned}
\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} &= \lim_{n\to\infty} \frac{n^2+6}{n \lg n} = \lim_{n\to\infty} \frac{n+\frac{6}{n}}{\lg n} \\
&= \lim_{n\to\infty} \frac{1-\frac{6}{n^2}}{\frac{1}{\ln 2 \cdot n}} \text{ (by L'Hospital Rule)} \\
&= \ln 2 \lim_{n\to\infty}(n - \tfrac{6}{n}) = \ln 2(\infty - 0) = \infty
\end{aligned}$$

By Limit Test, we have $n^2 + 6 = \omega(n \lg n)$.

Example

Example 3

$T_1(n) = (\ln n)^k$, $T_2(n) = n^\epsilon$, where $k > 0$ is any (large) constant and $\epsilon > 0$ is any (small) constant. (Recall: $\ln n = \log_e n$.)

$$
\begin{aligned}
& \lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = \lim_{n\to\infty} \frac{(\ln n)^k}{n^\epsilon} \text{ (use L'Hospital Rule)} \\
= \quad & \lim_{n\to\infty} \frac{k(\ln n)^{k-1} \times (1/n)}{\epsilon n^{(\epsilon-1)}} \\
= \quad & \frac{k}{\epsilon} \lim_{n\to\infty} \frac{(\ln n)^{k-1}}{n^\epsilon} \text{ (use L'Hospital Rule again and simplify)} \\
= \quad & \frac{k(k-1)}{\epsilon^2} \lim_{n\to\infty} \frac{(\ln n)^{k-2}}{n^\epsilon} \text{ (use L'Hospital Rule } k \text{ times)} \\
& \ldots\ldots \\
= \quad & \frac{k(k-1)\cdots 2\cdot 1}{\epsilon^k} \lim_{n\to\infty} \frac{1}{n^\epsilon} = 0
\end{aligned}
$$

So by Limit Test, $(\ln n)^k = o(n^\epsilon)$ for any k and ϵ. For example, take $k = 100$ and $\epsilon = 0.01$, we have $(\ln n)^{100} = o(n^{0.01})$.

Example

Example 4

$T_1(n) = n^k$, $T_2(n) = a^n$, where $k > 0$ is any (large) constant and $a > 1$ is any constant bigger than 1.

$$
\begin{aligned}
& \lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = \lim_{n\to\infty} \frac{n^k}{a^n} \text{ (using L'Hospital Rule)} \\
= \; & \lim_{n\to\infty} \frac{k \cdot n^{k-1}}{\ln a \cdot a^n} = \frac{k}{\ln a} \lim_{n\to\infty} \frac{n^{k-1}}{a^n} \text{(using L'Hospital Rule } k \text{ times)} \\
= \; & \frac{k(k-1)\cdots 2\cdot 1}{(\ln a)^k} \lim_{n\to\infty} \frac{n^0}{a^n} \\
= \; & \frac{k(k-1)\cdots 2\cdot 1}{(\ln a)^k} \lim_{n\to\infty} \frac{1}{a^n} = 0
\end{aligned}
$$

So by Limit Test, $n^k = o(a^n)$ for any $k > 0$ and $a > 1$. For example, take $k = 1000$ and $a = 1.001$, we have $n^{1000} = o((1.001)^n)$.

Example

Example 5

$T_1(n) = \log_a n$, $T_2(n) = \log_b n$, where $a > 1$ and $b > 1$ are any two constants bigger than 1.

By the Log Base Change Formula: $\log_b n = \log_b a \cdot \log_a n$

Thus: $\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = \lim_{n\to\infty} \frac{\log_a n}{\log_b n} = \lim_{n\to\infty} \frac{\log_a n}{\log_b a \cdot \log_a n} = \frac{1}{\log_b a}$

Since $\frac{1}{\log_b a} > 0$ is a constant, we have $\log_a n = \Theta(\log_b n)$ by Limit Test.

So: **the growth rates of the** $\log$ **functions are the same for any base** > 1.

Example

Example 6

$T_1(n) = a^n$, $T_2(n) = b^n$, where $1 < a < b$ are any two constants.

We have: $\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = \lim_{n\to\infty} \frac{a^n}{b^n} = \lim_{n\to\infty} (\frac{a}{b})^n = 0$.
Thus: $a^n = o(b^n)$ (for any $1 < a < b$) by Limit Test.

The list of common functions:

The following list shows the functions commonly used in algorithm analysis, in the order of increasing growth rate (a, b, c, d, k, ϵ are positive constants, $\epsilon < 1$, $k > 1$, $d > 1$ and $a < b$):

$$c, \log_d n, (\log_d n)^k, n^\epsilon, n, n^k, a^n, b^n, n!, n^n$$

in the sense that if $f(n)$ and $g(n)$ are any two consecutive functions in the list, we have $f(n) = o(g(n))$

Examples

Example 7

$T_1(n) = n!$ and $T_2(n) = a^n$ $(a > 1)$

- $\lim_{n\to\infty} \frac{a^n}{n!} = ?$
- L'Hospital Rule doesn't help: We don't know how to take derivative of $n!$

$$\frac{a^n}{n!} = \underbrace{\frac{a}{1} \cdot \frac{a}{2} \cdots \frac{a}{2\lceil a\rceil}}_{2\lceil a\rceil \text{ terms}} \cdot \underbrace{\frac{a}{2\lceil a\rceil + 1} \cdots \frac{a}{n}}_{(n-2\lceil a\rceil) \text{ terms}}$$

The first part is a constant $c > 0$. In the second part, each term $< 1/2$. So:

$$0 \leq \lim_{n\to\infty} \frac{a^n}{n!} \leq c \cdot \lim_{n\to\infty} (\frac{1}{2})^{(n-2\lceil a\rceil)} = 0$$

By Limit Test: $a^n = o(n!)$.

Stirling Formula

Stirling Formula

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} \le n! \le \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$

or:
$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

When $n = 10$;

- $n! = 3628800$.
- $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n = 3598696$, 99% accurate.

Examples

Example 7 (another solution)

$T_1(n) = n!$ and $T_2(n) = a^n$ $(a > 1)$

$$\lim_{n\to\infty} \frac{n!}{a^n} \geq \lim_{n\to\infty} \sqrt{2\pi n}\left(\frac{n}{ae}\right)^n = \lim_{n\to\infty} \sqrt{2\pi n} \cdot \lim_{n\to\infty}\left(\frac{n}{ae}\right)^n$$

The first limit is ∞. The second limit goes to ∞^∞. So it's also ∞. Thus $\lim_{n\to\infty} \frac{n!}{a^n} = \infty$ and $n! = \omega(a^n)$ by limit test.

Example 8

$T_1(n) = n^n$ and $T_2(n) = n!$

By using similar method as in Example 7, we can show: $n! = o(n^n)$

Summations

Consider the following simple program:

1: **for** $i = 1$ **to** n **do**
2:     the loop body takes $\Theta(i^k)$ time
3: **end for**

What's the runtime of this program? It should be:

$$T(n) = \sum_{i=1}^{n} \Theta(i^k) = c \sum_{i=1}^{n} i^k \quad \text{(for some constant } c\text{)}$$

Thus, it is important to know the sum of the form $\sum_{i=1}^{n} i^k$.

Summation formulas

$$\sum_{i=1}^{n} i^1 = 1 + 2 \cdots + n = \frac{n(n+1)}{2} = \Theta(n^2) \quad (1)$$

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3) \quad (2)$$

$$\sum_{i=1}^{n} i^3 = 1^3 + 2^3 \cdots + n^3 = \frac{n^2(n+1)^2}{4} = \Theta(n^4) \quad (3)$$

In general, for any $k > 0$, the following is true.

$$\sum_{i=1}^{n} i^k = \Theta(n^{k+1}) \quad (4)$$

Summations:

By using these formulas, we can compute the runtime of nested loops.

Example

for $i = 1$ **to** n **do**
  for $j = i$ **to** n **do**
    for $k = i$ **to** j **do**
      $(\ldots$ loop body takes $\Theta(1)$ time.)
    end for
  end for
end for

Since the inner loop body takes $\Theta(1)$ time, we only need to count the number $D(n)$ of the inner loop iterations. Then $T(n) = D(n) \cdot \Theta(1) = \Theta(D(n))$.

$$D(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=i}^{j} 1 = \sum_{i=1}^{n} \sum_{j=i}^{n} (j - i + 1)$$

Calculate $D(n)$

To calculate the second sum, let $t = j - i + 1$. When $j = i$, $t = 1$. When $j = n$, $t = n - i + 1$. Thus

$$\sum_{j=i}^{n}(j - i + 1) = \sum_{t=1}^{n-i+1} t = 1 + 2 \cdots + (n - i + 1) = \frac{(n - i + 2)(n - i + 1)}{2}$$

Next we calculate: $\sum_{i=1}^{n} \frac{(n-i+2)(n-i+1)}{2}$. Let $s = n - i + 1$. When $i = 1$, $s = n$. When $i = n$, $s = 1$. Thus:

$$\begin{aligned} \sum &= \sum_{s=1}^{n} \frac{(s+1)s}{2} = \frac{1}{2}\{\sum_{s=1}^{n} s^2 + \sum_{s=1}^{n} s\} \\ &= \frac{1}{2}\{\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\} = \Theta(n^3) \end{aligned}$$

More Summations:

The following summation formulas are useful.

$$\sum_{i=0}^{n} a^i = 1 + a + a^2 + \cdots + a^n = \begin{cases} \frac{1-a^{n+1}}{1-a} & = \Theta(1) & \text{if } 0 < a < 1 \\ n+1 & = \Theta(n) & \text{if } a = 1 \\ \frac{a^{n+1}-1}{a-1} & = \Theta(a^n) & \text{if } 1 < a \end{cases} \tag{5}$$

$\sum_{i=0}^{n} a^i$ is called geometric series.

$$H(n) = 1 + 1/2 + 1/3 + \cdots + 1/n = \sum_{i=1}^{n} \frac{1}{i} = \Theta(\ln n) \tag{6}$$

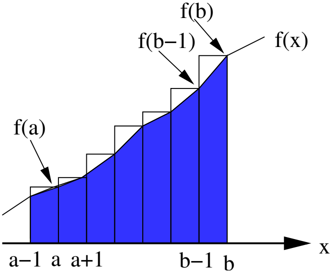
$H(n)$ is called Harmonic series.
How to compute $H(n)$?

Integration Method

Let $f(x)$ be an increasing function. Then for any $a \leq b$:

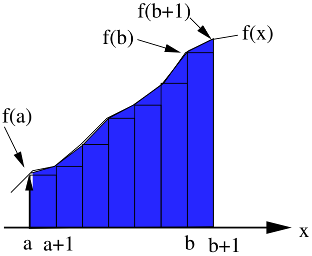
$$\int_{a-1}^{b} f(x)dx \leq \sum_{i=a}^{b} f(i) \leq \int_{a}^{b+1} f(x)dx$$

In the Fig, $\sum =$ the area of the staircase region. The first $\int =$ the area of the shaded region. Since $f(x)$ is increasing, the first $\leq$ holds.



Integration Method

In the Fig, $\sum$ is the area of the staircase region. The second $\int$ is the area of the shaded region. Since $f(x)$ is increasing, the second $\leq$ holds.



Similarly:

Let $f(x)$ be a decreasing function. Then for any $a \leq b$:

$$\int_{a-1}^{b} f(x)dx \geq \sum_{i=a}^{b} f(i) \geq \int_{a}^{b+1} f(x)dx$$

Example

For any $k > 0$, $f(x) = x^k$ is an increasing function. Let $a = 1$ and $b = n$.

$$\int_0^n x^k dx \leq \sum_{i=1}^{n} i^k \leq \int_1^{n+1} x^k dx$$

$\int_0^n x^k dx = \frac{1}{k+1} x^{k+1} \Big|_{x=0}^{x=n} = \frac{n^{k+1}}{k+1}$; $\quad \int_1^{n+1} x^k dx = \frac{1}{k+1} x^{k+1} \Big|_{x=1}^{x=n+1} = \frac{(n+1)^{k+1}-1}{k+1}$

Thus:

$$\frac{n^{k+1}}{k+1} \leq \sum_{i=1}^{n} i^k \leq \frac{(n+1)^{k+1} - 1}{k+1}$$

By limit test, both lower and upper bounds $= \Theta(n^{k+1})$. Thus $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$.

Example

$f(x) = \frac{1}{x}$ is a decreasing function. Let $a = 1$ and $b = n$.

$$\int_0^n \frac{dx}{x} \geq \sum_{i=1}^{n} \frac{1}{i} \geq \int_1^{n+1} \frac{dx}{x}$$

$\int_0^n \frac{dx}{x} = \ln x \,|_0^n = \ln n - (-\infty) = \infty$. This doesn't work! Try $a = 2$ and $b = n$:

$$\int_1^n \frac{dx}{x} \geq \sum_{i=2}^{n} \frac{1}{i} \geq \int_2^{n+1} \frac{dx}{x}$$

This gives: $(\ln n - \ln 1) \geq \sum_{i=2}^{n} \frac{1}{i} \geq (\ln(n+1) - \ln 2)$.

Note $\ln 1 = 0$. Add 1 to the above: $1 + \ln n \geq \sum_{i=1}^{n} \frac{1}{i} \geq (\ln(n+1) - \ln 2 + 1)$.

By limit test, both lower and upper bounds $= \Theta(\ln n)$. So
$H(n) = \sum_{i=1}^{n} \frac{1}{i} = \Theta(\ln n)$

Note: $\lim_{n\to\infty}(\ln n - \sum_{i=1}^{n} \frac{1}{i}) = c$, where $c = 0.577...$ is Euler constant.

Solving Linear Recursive Equations

Fibonacci number

$$\text{Fib}_0 = 0,\ \text{Fib}_1 = 1, \ldots,\ \text{Fib}_{n+2} = \text{Fib}_{n+1} + \text{Fib}_n$$

How to compute Fib_n directly from n?

$$\text{Fib}_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

- $\frac{1\pm\sqrt{5}}{2}$ are the two roots of the equation: $x^2 = x + 1$.
- Since $\left|\frac{1-\sqrt{5}}{2}\right| < 1$, the second term $\to 0$. So $\text{Fib}_n \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n$.
  ($\alpha = \frac{1+\sqrt{5}}{2} = 1.618...$ is called the golden ratio.)
- For $n = 8$, $\text{Fib}_8 = 21$ where $\frac{1}{\sqrt{5}}\alpha^n = 21.0095$.
- How to find such formula?

Linear recursive sequences

Linear recursive sequences

A sequence $\{f_0, f_1, \ldots, f_n \ldots\}$ is called a linear recursive sequence of order k if it is defined as follows:

- $f_0, f_1, \ldots, f_{k-1}$ are given.
- For all $n \geq 0$, $f_{n+k} = c_{k-1}f_{n+k-1} + c_{k-2}f_{n+k-2} + \cdots + c_1 f_{n+1} + c_0 f_n$ where $c_{k-1}, c_{k-2}, \ldots c_0$ are fixed constants.

Example 1: $\{\mathrm{Fib}_n\}$ is a linear recursive sequence of order 2 where $c_1 = 1$ and $c_0 = 1$.

Example 2: $f_0 = 1, f_1 = 2, f_2 = 4$ and for all $n \geq 0$, $f_{n+3} = 3f_{n+1} - 2f_n$
Then $\{f_n\}$ is a linear recursive sequence of order 3 where $c_2 = 0$, $c_1 = 3$ and $c_0 = -2$.

Solving linear recursive sequences

- The characteristic equation of the linear recursive seq is:

$$x^k = c_{k-1}x^{k-1} + c_{k-2}x^{k-2} + \cdots + c_1x^1 + c_0$$

- Solve this equation for x. Let $\alpha_1, \ldots, \alpha_k$ be the roots.
- Assuming all roots are distinct. Then the solution of f_n has the form

$$f_n = a_1(\alpha_1)^n + a_2(\alpha_2)^n + \cdots + a_k(\alpha_k)^n$$

  for some constants $a_1, a_2, \ldots, a_k$.
- Plug in the initial values $f_0, f_1, \ldots, f_{k-1}$, we get k equations. Solve them to find $a_1, a_2, \ldots, a_k$.

Example

Fibonacci number

$$F_0 = 0, F_1 = 1 \text{ and } F_{n+2} = F_{n+1} + F_n$$

- The characteristic equation is: $x^2 = x + 1$.
- The roots are: $\alpha_1 = \frac{1+\sqrt{5}}{2}$ and $\alpha_2 = \frac{1-\sqrt{5}}{2}$
- The solution has the form: $F_n = a_1(\alpha_1)^n + a_2(\alpha_2)^n$
- Plug in initial value $F_0 = 0$ and $F_1 = 1$:

$$\begin{aligned} 0 = F_0 = a_1\alpha_1^0 + a_2\alpha_2^0 &= a_1 + a_2 \\ 1 = F_1 = a_1\alpha_1^1 + a_2\alpha_2^1 &= a_1\frac{1+\sqrt{5}}{2} + a_2\frac{1-\sqrt{5}}{2} \end{aligned}$$

- Solving this equation system, we get: $a_1 = \frac{1}{\sqrt{5}}$ and $a_2 = -\frac{1}{\sqrt{5}}$.
- Thus:

$$\mathsf{F}_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Solving linear recursive sequences

- The roots $\alpha_1, \ldots, \alpha_k$ of the characteristic equ are not distinct.
- Say $\alpha_1 = \alpha_2 = \ldots = \alpha_t$ repeats t times.
- Then in the solution formula, the portion

$$\cdots a_1(\alpha_1)^n + a_2(\alpha_2)^n + \cdots + a_t(\alpha_t)^n \cdots$$

  is replaced by:

$$\cdots a_1(\alpha_1)^n + a_2 n^1 (\alpha_1)^n + \cdots + a_t n^{t-1} (\alpha_1)^n \cdots$$

- Other steps are the same.

Example

Example

$F_0 = 1, F_1 = 2, F_2 = 4$ and for all $n \geq 0$, $F_{n+3} = 3F_{n+1} - 2F_n$

- The characteristic equation is: $x^3 = 3x - 2$ or $f(x) = x^3 - 3x + 2 = 0$.
- To solve $f(x) = 0$, try $x = 1$. We find $x = 1$ is a root. So $(x-1)$ is a factor of $f(x)$. Thus $f(x) = (x-1)(x^2 + x - 2) = (x-1)(x-1)(x+2)$.
- So the roots are $\alpha_1 = \alpha_2 = 1$ and $\alpha_3 = -2$.
- The solution has the form: $F_n = a_1 \cdot 1^n + a_2 \cdot n \cdot 1^n + a_3 \cdot (-2)^n$. Plug in initial values:

$$\begin{aligned} 1 = F_0 &= a_1 + 0 + a_3 \\ 2 = F_1 &= a_1 + a_2 - 2a_3 \\ 4 = F_2 &= a_1 + 2a_2 + 4a_3 \end{aligned}$$

- Solving this: $a_1 = 8/9$, $a_2 = 4/3$ and $a_3 = 1/9$
- Thus: $F_n = \frac{8}{9} + \frac{4}{3}n + \frac{1}{9}(-2)^n$