Asymptotic Notations.

Consider $f: \mathbb{R}^{+} \xrightarrow{(N)} \mathbb{R}^{+}$, we have five notations.
data ≥ 0, steps ≥ 0

Definition

1. Big-O: $O(g(n)) := \{h(n) \mid \exists c>0, n_0 \in N \text{ s.t. } h(n) \le c g(n), \forall n \ge n_0\}$
2. Big-Ω: $\Omega(g(n)) := \{h(n) \mid \exists c>0, n_0 \in N \text{ s.t. } c g(n) \le h(n), \forall n \ge n_0\}$
3. Big-Θ: $\Theta(g(n)) := O(g(n)) \cap \Omega(g(n))$
$= \{h(n) \mid \exists c_1, c_2 > 0, n_0 \in N \text{ s.t. } c_1 g(n) \le h(n) \le c_2 g(n), \forall n \ge n_0\}$
4. Little-o: $o(g(n)) := \{h(n) \mid \forall c>0, \exists n_0 \in N \text{ s.t. } h(n) \le c \cdot g(n), \forall n \ge n_0\} = \{h(n) \mid \lim_{n\to\infty} \frac{h(n)}{g(n)} = 0\}$
5. Little-ω: $\omega(g(n)) := \{h(n) \mid \forall c>0, \exists n_0 \in N \text{ s.t. } c \cdot g(n) \le h(n), \forall n \ge n_0\} = \{h(n) \mid \lim_{n\to\infty} \frac{g(n)}{h(n)} = 0\}$

Properties

1. Transitivity: $f \in O(g),\ g \in O(h) \Rightarrow f \in O(h)$ (also $\Omega, o, \Theta, \omega$)
2. Reflexivity: $f \in \Theta(f)$
3. Symmetry / Transpose symmetry: $f \in \Theta(g) \leftrightarrow g \in \Theta(f)$ (O / Ω, o / ω)
4. No total ordering
counterexample: $g(n) = \begin{cases} 1, & n = 2k \\ n^2, & n = 2k+1 \end{cases}$

Recurrence Relation

Proof by induction.

$k=1$ 估一個 c
$k=n$ (assuming $\forall k \le n-1$ 都對) 估另一個 c.
猜: 加個低次項.

Repeated substitution

$T(n) \to T(\frac{n}{2}) \to T(\frac{n}{4})$.

Recursion Tree

* 大小樹 / change of variable.
Eg. $T(2^m) = 2T(2^{\frac{m}{2}}) + m$
$\Rightarrow S(m) = 2S(\frac{m}{2}) + m,\ S(m) = m\log_2 m$
$T(n) = T(2^m) = \log_2 n \log(\log n)$

Master theorem:

$T(n) = aT(n/b) + f(n)$, $a \ge 1, b > 1$, $f(n)$ asymptotically positive.
$f(n) > 0$ when n's large en.

Three cases:

(1) $f(n) \in O(n^{\log_b a} \cdot n^{-\varepsilon})$, $\varepsilon > 0 \Rightarrow T(n) \in \Theta(n^{\log_b a})$ f(n) 跑得比 $n^{\log_b a}$ 慢
(2) $f(n) \in \Theta(n^{\log_b a} \cdot (\log n)^k)$, $k \ge 0 \Rightarrow T(n) \in \Theta(f(n)\log n) = \Theta(n^{\log_b a} \cdot (\log n)^{k+1})$.
(3) $f(n) \in \Omega(n^{\log_b a} \cdot n^{\varepsilon})$, $\varepsilon > 0 \Rightarrow T(n) \in \Theta(f(n))$. f(n) 跑得比 $n^{\log_b a}$ 快
$af(n/b) < cf(n)$ for some $c<1$

Divide and Conquer

練 pseudocode.

Eg. $A[i]$ well-ordered. find $A[i] = 0$ $\log n$. $\Rightarrow A[\frac{x+y}{2}] \begin{cases} = 0 \\ > 0, & Alg(A, x, \frac{x+y}{2}) \\ < 0, & Alg(A, \frac{x+y}{2}, y) \end{cases}$
Eg. $B[i]$ well-ordered. find $B[i] = i$ (不要整段吃) $\log n$.
Eg. **Max subseq. sum** $\begin{cases} m_1 = MSS(a_1 \sim a_{\frac{n}{2}}) \\ m_2 = MSS(a_{\frac{n}{2}+1} \sim a_n) \\ m_3 = \max_i (a_i \sim a_{\frac{n}{2}}) \end{cases}$ $m_4 = \max_j (a_{\frac{n}{2}+1} \sim a_j)$
return $\max(m_1, m_2, m_3 + m_4)$
Eg. **Comparison sorting.** compare and swap.

Stable: equal elements don't swap.
In-place: only use $\Theta(1)$ (const.) extra storage places.

- Insertion sort: 從 a_1 開始往前換到不能換. worst case: $\Theta(n^2)$.
- Bubble sort. $a_1 a_2, a_2 a_3 \cdots$ 比. all cases: $\Theta(n^2)$
- Stupid sort. $a_1 a_2, a_2 a_3 \cdots$ 比, 然後從頭. worst case: $\Theta(n^3)$.
- Stooge sort. 前 2/3 後 2/3 前 2/3. $T(n) = 3T(\frac{2n}{3})$ Master thm
- Quick sort. (deterministic)
  1. first element x as pivot.
  2. $\boxed{\le x}$ ⊗ $\boxed{\ge x}$
  3. recursively sort
  4. merge

  worst case: $\Theta(n^2)$. n-1 comparisons, $\le \lceil \frac{n}{2} \rceil$ swaps

  (randomized) 1. randomly choose one element as pilot.
  worst: $T(n) = T(n-1) + \Theta(n) \in \Theta(n^2)$
  best: $T(n) = 2T(\frac{n}{2}) + \Theta(n) \in \Theta(n\log n)$

  $T(n) = T(0) + T(n-1) + n \cdots T(n) = T(n-1) + T(0) + n$
  $\Rightarrow T(n) = \frac{1}{n}(T(0) + T(n-1) + n) + \cdots + \frac{1}{n}(T(n-1) + T(0) + n)$
  $\Rightarrow T(n) = \frac{2}{n}\sum_{0}^{n-1} T(i) + n = \frac{2}{n}\sum_{0}^{n-2} T(i) + n + \frac{2}{n}T(n-1)$
  $T(n-1) = \frac{2}{n-1}\sum_{0}^{n-2} T(i) + (n-1)$
  $\Rightarrow (n-1)T(n-1) = 2\sum_{0}^{n-2} T(i) + (n-1)^2$
  $nT(n) = 2\sum_{0}^{n-2} T(i) + n^2 + 2T(n-1)$
  $\Rightarrow nT(n) - (n-1)T(n-1) = 2T(n-1) + 2n - 1.$
  $\Rightarrow \frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2n-1}{n(n+1)}$
  $S(n) = S(n-1) + \frac{3}{n+1} - \frac{1}{n}$
  $S(n) = \Theta(\log n)$. $T(n) = \Theta(n\log n)$.

  $S(n) = S(n-1) + \frac{3}{n+1} - \frac{1}{n}$
  $S(n-1) = S(n-2) + \frac{3}{n} - \frac{1}{n-1}$
  ⋮
  $S(2) = S(1) + \frac{3}{3} - \frac{1}{1}$
  $S(n) = S(1) + \frac{1}{n+1} - \frac{1}{1} +$

- Merge sort.
  Mergesort$(a_1, \dots, a_n)$
  if $n = 1, 2$. sort directly
  else $A = $ Mergesort$(a_1, \dots, a_{\frac{n}{2}})$
  $B = $ Mergesort$(a_{\frac{n}{2}+1}, \dots, a_n)$
  return Merge(A, B)

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

- Decision tree.
  $2^t \ge n!$, $t \ge \log_2 n!$ $(n! \sim \sqrt{2\pi n}(\frac{n}{e})^n)$
  $= \log_2 n + \log_2 (n-1) + \cdots + \log_2 \frac{n}{2} + \cdots + \log_2 1$
  $\ge \frac{n}{2}\log_2 \frac{n}{2} + \cdots + \frac{n}{2}\log_2 \frac{n}{2} + 0 + \cdots + 0$
  $= \frac{n}{2}\log\frac{n}{2}$

- **Ordered statistics.**
  1. randomized. pivot x. $T(n) = \begin{cases} \Theta(1) + n \text{ ?} \\ T(k-1) + n \\ T(n-k) + n \end{cases} \le \max(T(k-1), T(n-k)) + n$

  worst case: $\Theta(n^2)$.
  not so bad: $T(n) \le T(\frac{99n}{100}) + n$. $O(n)$?
  avg: sub, $T(n) \le 4n$

  2. select(i, n) $\Theta(n)$
     1. divide n into $\frac{n}{5}$ groups, each of 5, sort, median.
     2. recursively select $\frac{n}{5}$ medians and use it as pivot. $T(\frac{n}{5})$
     3. D&C $\boxed{\le x}$ ⊗ $\boxed{\ge x}$.
     Claim: $\le T(\frac{3}{4}n)$.

  $T(n) \le T(\frac{n}{5}) + T(\frac{3}{4}n) + \Theta(n) \Rightarrow T(n) \in O(n)$.

Greedy Algorithm.

Eg. Meeting Scheduling earliest ending
Eg. Machine waiting problem: $\min(T_1 + T_2 + \cdots + T_n)$
$l_1 \le \cdots \le l_n$
OPT: $nl_1 + (n-1)l_2 + \cdots + 2l_{n-1} + l_n$ ...
With weights: $\min(w_1T_1 + w_2T_2 + \cdots + w_nT_n)$ rank t_i/w_i ?
Eg. Restaurant seat problem.
Greedy: pick $\max\{G_1 + \cdots + G_k, G_{k+1}\}$ OPT ≤ 2 Greedy*
OPT: ... Greedy ... Greedy

Dynamic Programming

m個東西, 可取可不取.

Eg. **Trip planning** "$c_i \ge 0$"

$A[i] = c_i \quad \forall i \le 20$
$B[i] = 0$) $\Theta(1)$
for $i = 21$ to n ?
$A[i] = \min(A[i-20], A[i-19], \dots, A[i-1]) + c_i$) $\Theta(1)$ → $\Theta(n)$
$B[i] = j$, $A[j]$ solves for i.
return$[n]$
return $n, B[n], B[B[n]], \dots$

Eg. **Resturant seat problem.**

$S[j,k] = \max_{H} \sum_{i \in H} a_i d_i$ s.t. $\begin{cases} \sum_H a_i \le k \\ H \subseteq \{G_1, \dots, G_j\} \end{cases}$

$T[j,k] :=$ set that solves $S[j,k]$'s max prob.

$S[0,k] = 0, \forall k$
$S[j,k] = -\infty, \forall k \le 0$
for $j = 1$ to m
for $k = 1$ to n) $m \times n$ → $\Theta(mn)$
$S[j,k] = \max\begin{cases} S[j-1,k] \\ S[j,k-a_j] + a_jd_j \end{cases}$ $\Theta(1)$
$T[j,k] = \begin{cases} T[j-1,k] \\ T[j-1,k-a_j] \cup \{G_j\} \end{cases}$ $\Theta(1)$
return $S[m,n], T[m,n]$

$T[j,k] = \begin{cases} 0 \\ 1 \end{cases}$
Check $T[m,n]$.
if $T[m,n] = 0 \Rightarrow G_m$ bye, check $T[m-1,n]$ for G_{m-1}
$= 1 \Rightarrow G_m$ accepted, check $T[m-1, n-a_m]$ for G_{m-1}) $\Theta(m)$.

Eg. **Sequence alignment.** mismatch: c_1, gap: c_2 $\Rightarrow$ min cost.

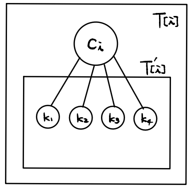
$C[m,n]$ (first m of X and n of Y)

$X[m]$ $Y[n]$
1. $X[m] - Y[n]$: $C[m,n] = C[m-1,n-1] + \text{cost}(X[m] - Y[n])$.
2. $X[m] - \text{gap}$: $C[m-1,n]$ $\begin{matrix} X[m] \\ | \\ \text{gap} \end{matrix}$ $\Rightarrow C[m,n] = C[m-1,n] + c_2$
3. $Y[n] - \text{gap}$: $C[m,n] = C[m,n-1] + c_2$

$C[i,0] = 0 = C[0,j] \quad \forall i, j$
for $i = 1$ to m.
for $j = 1$ to n.
$C[i,j] = \min\begin{cases} C[i-1,j-1] + \text{cost}(X[i] - Y[j]) \\ C[i-1,j] + c_2 \\ C[i,j-1] + c_2 \end{cases}$ $\Theta(mn)$
$B[i,j] = \begin{cases} 1 \\ 2 \\ 3 \end{cases}$
return $C[m,n]$.

check $B[m,n]$:
$B[m,n] = 1$ $X[m] - Y[n]$, $B[m-1,n-1]$ for $X[m-1], Y[n-1]$
$B[m,n] = 2$ $X[m] - \text{gap}$. $B[m-1,n]$ for $X[m-1], Y[n]$) $\Theta(m+n)$
$B[m,n] = 3$ $\text{gap} - Y[n]$, $B[m,n-1]$ for $X[m], Y[n-1]$

Eg. **Sequence alignment.**



$A[i] := \max \sum_{T[i]} c_i$ $\qquad A'[i] = A[k_1] + \cdots + A[k_4]$
$A'[i] := \max \sum_{T'[i]} c_i$ $\qquad A[i] = \max\begin{cases} A'[i] \\ c_i + A'[k_1] + \cdots + A'[k_4] \end{cases}$

$\Rightarrow$ Let $T[i]$ be the tree with i, $T'[i] = T[i] - \{i\}$
$A[i]$ be opt value in $T[i]$, $A'[i]$ in $T'[i]$
$A'[i] =$ 〃, $A = \max\begin{cases} c_i + A'[k_1] + \cdots + A'[k_m] \\ A'[i] \end{cases}$
for $i = n$ down to 1: find A', A

Sorting in Linear Time

1. Counting sort : Counting $1 \times m_1$, $2 \times m_2$, $\vdots$, $k \times m_k$ $\Theta(n)$ and $<1 : 0$, $<2 : 0 + m_1$, $\vdots$, $<k : \sum^{k-1} m_i$ $\Theta(k)$ $T(n) = \Theta(n+k)$

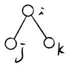
   Stable

2. Radix sort : Apply counting sort digit by digit, lowerest digit first.

   $\Rightarrow$ Assume n numbers, d digits, base m, $\Theta(d(m+n))$

   $\Rightarrow$ Convert $a_1, \cdots, a_n$ into base n, $a_i \in \{0, 1, \cdots, n^d - 1\}$, $\Theta(d(n+n))$

3. Heap sort : Binary Max Heap



   node i is j/k's parent $\Rightarrow$ $\text{Parent}(i) = \lfloor \frac{i}{2} \rfloor$

   j/k are i's left/right child Left child $= 2i$, Right child $= 2i+1$

   $\text{value}(i) \le \text{value}(\text{parent}(i))$

   Max-Heapify : compare $A[i], A[2i], A[2i+1]$, if max $= A[i]$, done

   $A[2i]$, swap then $MH(2i)$

   $A[2i+1]$, swap then $MH(2i+1)$

   $\Theta(n \log n)$

   Build-Heap : for $i = n$ down to 1 : $MH(A, i)$ $\Theta(n \log n)$

   Heap Sort : Build-Heap
   for $i = n$ down to 2:
   Swap $A[i], A[1]$
   heapsize --
   $MH(A, 1)$

Problems

$$(\log n)^{\log n}$$

$\Downarrow$

$* \ (\log n)! = (\log n)(\log n - 1) \cdots (\frac{1}{2}\log n)(\frac{1}{2}\log n - 1) \cdots 1$

$\Downarrow$

$\frac{1}{2}\log n \quad \frac{1}{2}\log n \cdots \frac{1}{2}\log n \quad 1 \cdots 1$

$$(\log n)^{\log n} = (2^{\log\log n})^{\log n} = 2^{\log n \log\log n} = n^{\log\log n} \ge n^c$$

$$\le 2^n$$

$*$ FindMIN(Array [ ]) : $T(n)$

```
low = index of the first element
high = index of the last element
while (Array[low] > Array[high]):
    mid = ⌊(low + high)/2⌋
    if (Array[low] < Array[mid]): low = mid + 1      T(n/2)
    else: high = mid
return Array[low]
```

$*$ Stirling formula : $\log_2 n! = n\log_2 n - n\log_2 e + O(\log_2 n)$

$*$ $T(n) = \sum_{i}^{\lfloor \frac{n}{2} \rfloor} T(n - 2i) \Rightarrow T(n) = T(n-2) + T(n-4) + \cdots + T(1) + T(0)$

$-) \ T(n-2) = \quad T(n-4) + \cdots + T(1) + T(0)$

$T(n) = 2T(n-2) = 2^{\lfloor \frac{n}{2} \rfloor} T(0) T(1) \in \Theta(2^{\frac{n}{2}})$

$*$ Find with Duplicate Element :

```
FindMIN(Array[ ]):
    Find low and high of index
    while (low < high):
        mid = ⌊(low + high)/2⌋
        if A[mid] = A[high]: high --
        elif A[mid] > A[high]: low = mid + 1
        else : high = mid
    reture A[high]
```

$*$ Truck Problem : $N \le 2N^*$

$\because N^* W = \sum_i^n \omega_i$, $\therefore N^* \ge \frac{1}{W} \sum \omega_i$.

Let L_j be the set of item for truck j, K_j be the weight of set.

$\forall j > 1$, $K_j + K_{j-1} > W$, and by greedy, $\sum_{i=1}^{N} K_j = \sum^n \omega_i \Rightarrow N = 2m, \sum_j^N K_j = \sum_j^m (K_{2j} + K_{2j-1}) > mW$

$N = 2m+1, \sum_j^N K_j = \sum_j^m (K_{2j} + K_{2j-1}) + K_{2m+1} > mW$

$\Rightarrow \sum_j^N K_j > \frac{1}{2} W(N-1)$ in general, $\frac{1}{2} W(N-1) < \sum_j^N K_j = \sum_i^n \omega_i \le WN^*$, $\therefore N - 1 \le 2N^*$

$*$ $T(n) = -4T(n-4) + 10 = (-4)^{\lfloor \frac{n}{4} \rfloor} \cdot C + 10(1 + (-4) + \cdots)$

$= (-4)^{\lfloor \frac{n}{4} \rfloor} \cdot C + 10 \frac{1 + (-4)^{\lfloor \frac{n}{4} \rfloor}}{1+4} \in \Theta(4^{\lfloor \frac{n}{4} \rfloor})$