# Tutorial

November 1, 2022

# PS 5-2

- Stirling's formula: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$
- $\log \binom{2n}{n} = \log \frac{(2n)!}{(n!)^2} = 2n - o(n)$.
- Cannot distinguish $[\ldots, x, y, \ldots]$ and $[\ldots, y, x, \ldots]$.

# Merge $k$ sorted list

- Lower bound: $\log \frac{(kn)!}{(n!)^k}$;

# PS 5-3

- $|\{i \in S \mid a \leq i \leq b\}| = |\{i \in S \mid i \leq b\}| - |\{i \in S \mid i \leq a-1\}|$.
- Group each integer according to its length, and apply radix sort in each group.

# PS 5-4

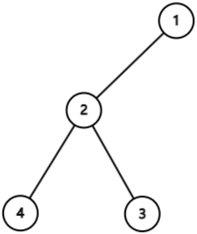
- $T(n) = T(\frac{5n}{7}) + T(\frac{n}{7}) + \Theta(n)$: $T(n) = \Theta(n)$.
- $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + \Theta(n)$: $T(n) = \Theta(n \log n)$.
- Two approaches:
  - Find quartiles as candidates.
  - Remove four distinct elements in one round.

# PS 5-5

- Dynamic programming:
  - Let $f_u$ be the maximum depth of complete subtree rooted at $u$;
  - $f_u = \min\left(f_{u_1}, f_{u_2}\right) + 1$.

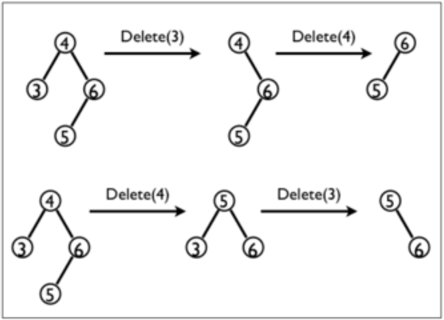
# PS 6-2

- No. Consider search path $1 - 2 - 4$.



# PS 6-2

- No.



# PS 6-3

- Insertion: it suffices to find the predecessor of the inserted node.

```
TREE-INSERT(T, z)
y = NIL
x = T.root
while x ≠ NIL
    y = x
    if z.key < x.key
        x = x.left
    else x = x.right
z.p = y
if y == NIL
    T.root = z        // tree T was empty
elseif z.key < y.key
    y.left = z
else y.right = z
```

(a) original

```
INSERT(T, z)
    y = NIL
    x = T.root
    pred = NIL
    while x != NIL
        y = x
        if z.key < x.key
            x = x.left
        else
            pred = x
            x = x.right
    if y == NIL
        T.root = z
        z.succ = NIL
    else if z.key < y.key
        y.left = z
        z.succ = y
        if pred != NIL
            pred.succ = z
    else
        y.right = z
        z.succ = y.succ
        y.succ = z
```

(b) modified

# PS 6-3

- Deletion: same merit, with an additional parent oracle.

```
PARENT(T, x)
    if x == T.root
        return NIL
    y = TREE-MAXIMUM(x).succ
    if y == NIL
        y = T.root
    else
        if y.left == x
            return y
        y = y.left
    while y.right != x
        y = y.right
    return y
```

# PS 6-5

- Consider a maximum right-going chain $C$.
- Right-Rotate $x$ if the left child of $x$ exists.

# PS 6-6

- $F_h = F_{h-1} + F_{h-2} + 1$;
- two rotation suffices.
- Maintain the height when insertion, and call balance subroutine along the insertion path.
- $O(1)$ rotation: the height of subtree decreases after balancing!

# PS 7-3

- Rejection sampling:
  - Draw $X \in [7]$ repeatedly until $X \neq 7$;
  - Draw $Y \in [7]$ repeatedly until $Y \leq 5$;
  - Output $2Y - (X \leq 3)$;
  - Expected number of samples: $\frac{7}{6} + \frac{7}{5}$.
- general (and better) algorithm:
  - In round $i$, generate $X_i \in [7]$;
  - Let $S = \sum_{j=1}^{i} \frac{X_i - 1}{7^i}$;
  - If $[S, S + \frac{1}{7^i}) \subseteq [\frac{R-1}{10}, \frac{R}{10})$, return $R$.
  - Expected number of samples:
    $2 + \sum_{k=2}^{+\infty} Pr[Y > k] = 2 + \sum_{k=2}^{+\infty} \frac{9}{7^k} = \frac{31}{14}$.

# PS 7-4

- Each coin will be flipped to heads in $O(\log n)$ rounds w.h.p. (For example, with probability at least $1 - n^{-2}$).
- Each coin will be flipped to heads in $O(1)$ rounds in expectation.

# Open Addressing hash table

- Given an open addressing hash table of size $m$. we assume the uniform hashing condition.
- Consider inserting $n \leq m/2$ elements, $X_i$ be the length of probe sequence.
- Prove that $E[\max_{1 \leq i \leq n} X_i] = O(\log n)$.

# PS 7-5

- Note that $2^p \mod m = 1$.
- Therefore, $\sum_{i=0}^{n} x_i 2^p \equiv \sum_{i=0}^{n} x_i \mod m$.

# PS 7-6

- $\epsilon > E_{(x,y)\in\binom{U}{2}}[P[h(x) = h(y)]] = \frac{\sum_{z\in B}\binom{c_z}{2}}{\binom{U}{2}}$
- $c_z$ is the number of $x \in U$ satisfying $h(x) = z$.
- By Cauchy-Schwarz inequality, $\sum_{z\in B}\binom{c_z}{2} \geq \frac{|U|^2-|U||B|}{2|B|}$.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \le L \le R \le n$ satisfying $\sum_{i=L}^{R} a_i \ge t$.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.
- Let $S_k = \sum_{i=1}^{k} A_i$(which can be computed in $O(n)$) and $S_0 = 0$.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.
- Let $S_k = \sum_{i=1}^{k} A_i$(which can be computed in $O(n)$) and $S_0 = 0$.
- Calculate the number of pairs $(i, j)$ such that $0 \leq i < j \leq n$ and $S_j > S_i + t$.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.
- Let $S_k = \sum_{i=1}^{k} A_i$(which can be computed in $O(n)$) and $S_0 = 0$.
- Calculate the number of pairs $(i, j)$ such that $0 \leq i < j \leq n$ and $S_j > S_i + t$.
- That's something we can solve using D&C technique.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.
- Let $S_k = \sum_{i=1}^{k} A_i$(which can be computed in $O(n)$) and $S_0 = 0$.
- Calculate the number of pairs $(i, j)$ such that $0 \leq i < j \leq n$ and $S_j > S_i + t$.
- That's something we can solve using D&C technique.
- To be more precise, we may modify merge sort process (combine step actually) to maintain the answer for subarray $S_l, S_{l+1}, \ldots, S_r$.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.
- Let $S_k = \sum_{i=1}^{k} A_i$(which can be computed in $O(n)$) and $S_0 = 0$.
- Calculate the number of pairs $(i, j)$ such that $0 \leq i < j \leq n$ and $S_j > S_i + t$.
- That's something we can solve using D&C technique.
- To be more precise, we may modify merge sort process (combine step actually) to maintain the answer for subarray $S_l, S_{l+1}, \ldots, S_r$.
- $O(n \log n)$ implementation: binary search indexes for each element in left array.

# Large Sum Subarray

- Given an array $A$ of length $n$ and parameter $t$, find the number of integer pairs $1 \leq L \leq R \leq n$ satisfying $\sum_{i=L}^{R} a_i \geq t$.
- It is indeed a modification of inversion counting.
  - Calculate the number of inversions of a given array $a$ of length $n$ in $O(n \log n)$.
  - Inversion: pair $(i, j)$ such that $i < j$ and $a_i > a_j$.
- Let $S_k = \sum_{i=1}^{k} A_i$(which can be computed in $O(n)$) and $S_0 = 0$.
- Calculate the number of pairs $(i, j)$ such that $0 \leq i < j \leq n$ and $S_j > S_i + t$.
- That's something we can solve using D&C technique.
- To be more precise, we may modify merge sort process (combine step actually) to maintain the answer for subarray $S_l, S_{l+1}, \ldots, S_r$.
- $O(n \log n)$ implementation: binary search indexes for each element in left array.
- $O(n)$ implementation: two pointer.