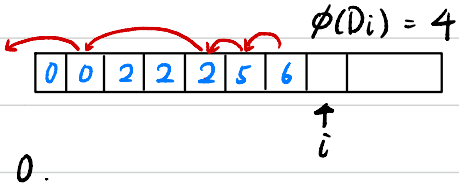
with B08902- 072, 037, 065, 022, ...
stackoverflow, wiki

Problem 5

(1)

1° Define $\phi(D_i)$ to be the # pivots for $dp[i-1]$, which means the total depth $H(i, x)$ can take on $i-1$. (Shown as graph below)

2° Validity check:



- $\phi(D_0) = 0$, the dp table is empty
- $\phi(D_i) \geq 0$, for $i \geq 1$, there is at least one step to 0.

3° Suppose $\phi(D_{i-1}) = k$, then there are two cases:

- if $a[i-1] > a[i]$, then $\phi(D_i) = k+1$

  $\Rightarrow \hat{C}_i = C_i + \phi(D_i) - \phi(D_{i-1}) = 1 + (k+1) - k = 2$

- if $a[i-1] < a[i]$, and it takes p steps to get to the nearest greater element, then $\phi(D_i) = k - p + 1$

  $\Rightarrow \hat{C}_i = C_i + \phi(D_i) - \phi(D_{i-1}) = 1 + p + (k - p + 1) - k = 2$

4° All situations have O(1) amortized cost, so total amortized time complexity: O(n)

(2)

< Algorithm Design >

1° First, I design a structure to keep track of the time, size, type of each group and I also use a deque to stimulate the line.

2° For each instruction, I add new cluster to line and as for the infection, I count the changing people's total existing time as their origin type.

(certain type += existing time × people being changed.)

3° After all instruction dealt, add the remain clusters to the result.

4° (Psuedo code)

```
total people <- cluster size + infection people

determine to process from the front or the back
while infection people > 0:

    if remaining infection people >= the current cluster size:
        infection people <- infection people - the current cluster size
        counter[this type] <- counter[this type] + (current time - time of that cluster) * the current cluster size
        pop out this cluster

    else:
        the current cluster size <- that current cluster size - remaining infection people
        counter[this type] <- counter[this type] + (current time - time of that cluster) * the current cluster size
        remaining infection people <- 0
end while

construct new cluster with total people and current time
push new cluster to the line
```

< Amortized Analysis >

1° Assign per-op amortized cost

Operation	Actual	Amortized
Push	1	3
Pop	1	0
Infection	1	0

2° Validity check :

For every push we deposit 1 or 2 credit. 1 credit is save for the time being pop. The other is spent if this push operation needs to infect part of a cluster (at most 1 group).

∴ There is always enough credit to pay for each operation.

3° All operations have O(1) amortized cost, so total amortized time complexity : O(N)

(3)

1° Without loss of generality, suppose it takes k operations for the tree to rebalance.

2° ∵ $\max(\text{size(node.left)}, \text{size(node.right)}) \le \alpha \cdot \text{size(node)}$, where $0.5 < \alpha < 1$

⇒ let left subtree ≥ right subtree and there's totally n node at the balance state (left = right = $\frac{n}{2}$). Now we insert k times to the left subtree and it force the tree to perform a rebalance.

$\therefore \frac{n}{2} \le \alpha \cdot n \xrightarrow{k \text{ operations}} \frac{n}{2} + k > \alpha(n+k) \Rightarrow (1-\alpha)k > (\alpha - \frac{1}{2})n$

$\Rightarrow k > \frac{\alpha - \frac{1}{2}}{1-\alpha} \cdot n$

3° Consider k operations when the tree is empty and insertion takes O(logn)

⇒ The total cost of k operations = $\underbrace{O(\log n) + O(\log n) + \cdots + O(\log n)}_{k \text{ times}} + \underbrace{O(n)}_{\text{rebalance}}$

∴ $T(n) = k \cdot O(\log n) + O(n)$ ⇒ Amortized per cost = $O(\log n) + O(1) = O(\log n)$. QED

(4)

< Algorithm Correctness >

1° It is trivial that without the break statement, we can obtain the range bitwise AND for all 1 to N. ($B[i] = A[i] \& A[i+1] \& \cdots \& A[N]$)

2° So now let's consider the break scenario. The base case $i = 1$ is trivial. Suppose it works for the first $i-1$ rounds and let $k < i$ which satisfy the break condition.

$B[k] = A[k] \& A[k+1] \& \cdots \& A[i-1] \Rightarrow B[k] \& A[i] = B[k]$

So for any r, where $1 \le r < k$, $B[r] = A[r] \& \cdots \& A[k-1] \& B[k]$

then we operate AND on B[r], we will get B[r]

$\Rightarrow B[r] \& A[i] = A[r] \& \cdots \& A[k-1] \& B[k] \& A[i]$

$= A[r] \& \cdots A[k-1] \& B[k] = B[r]$

so break condition is correct. Thus the algorithm is corrected.

< Time Complexity >

1° Assign per-op amortized costs

Operation	Actual	Amortized
B[i] ← A[i]	1	65
B[j] ← B[j] & A[i]	1	0

2° Validity check:

We can save 64 dollar for each B[i] ← A[i] assignment, and every B[i] has at most 64 chances to be flipped and AND operation (all 1 → all 0). So we can spend the saved

money to pay for B[j] ← B[j]&A[i]

∴ There is always enough credit to pay for each operation.

3° So, the total amortized time is O(N). QED

Problem 6 :

(1) Proof : $V - E + F = 2$ by induction on E

1° Base case : $E = 1 \Rightarrow V = 2, F = 1 \quad \therefore V - E + F = 2$ holds.

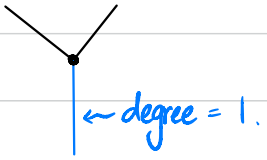
2° Assume $V - E + F = 2$ for $E = k > 1$

3° Consider $E = k + 1$, there are two cases

- There is a vertex whose degree $= 1$

$\therefore$ remove it $\Rightarrow (V-1) - (E-1) + F = 2$

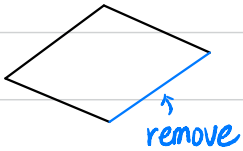
$\Rightarrow V - E + F = 2$



- There is no vertex whose degree $= 1$

$\therefore$ remove it $\Rightarrow V - (E-1) + (F-1) = 2$

$\Rightarrow V - E + F = 2$



4° By Induction 1°, 2°, 3° we know $V - E + F = 2$ holds.

Proof : $E \leq 3 \cdot V - 6$, if $E \geq 2$. consider different

1° Base case : $F = 1$

$\Rightarrow E = V - 1 \quad , E \geq 2 \, , V \geq 3$

$\Rightarrow$ Then $(3V-6) - E = (3V-6) - (V-1) = 2V - 5 \geq 1$

2° For $F > 1$, every region is bounded by at least 3 edges &

every edge is shared by at most 2 region

$\Rightarrow$ every region use at least $3/2$ edges $\Rightarrow F \leq E/(3/2) = 2E/3$.

$\because V-E+F=2 \Rightarrow V-E+2E/3 \geq 2 \Rightarrow E \leq 3V-6$.

3° From 1°, 2° we know, $E \leq 3 \cdot V-6$, if $E \geq 2 \Rightarrow$ Thus, $E = O(V)$

(2)

1° By apply Dijkstra's algorithm on adjacency list with priority queue, we can finish the task within $O((E+V)\log V) = O(V\log V)$ $(\because E=O(V))$

2° First, initial distance of source to 0 and others to INF. Also, add all vertex to a priority queue Q.

3° While Q is no empty, pop out the minimum element u and check all its neighbor vertexs that whether walking from u to them makes lower cost.

4° After processing all vertex, we will get the minimum distance to get to all vertex.

(3)

1° Construct a graph whose vertices which is strictly inside is greater than $\frac{2}{3}V$, then for every pair of (u,v), the shortest distance between (u,v) on its cycle is greater than that of a subgraph consist of its cycle and the vertices inside it. (Graph is Saiko Psycho Cycle).

2° So we can split the subgraph into two part by replacing the longer path on the cycle. And we can choose the lager (more vertices) part of the splitted graph.

3° Suppose it is split into A, B and $A \geq B$

$\Rightarrow A+B > \frac{2}{3}V \Rightarrow A > \frac{1}{3}V \therefore$ vertices strictly outside $< \frac{2}{3}V$

Also, the number of vertices is lesser so it's also $< 4k$.

However, the number of inside - outside is smaller for the new graph.

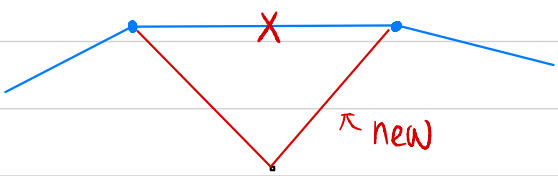
So, the origin graph is not a Saiko Psycho Cycle but new one is.

4° Thus, contradiction ⇒ distance is equal. QED.

(4) 1° Suppose a graph with Saiko Psycho Graph whose strictly inside vertices are greater than $\frac{2}{3}V$ but the number of vertices $< 4\lceil\sqrt{V+1}\rceil$

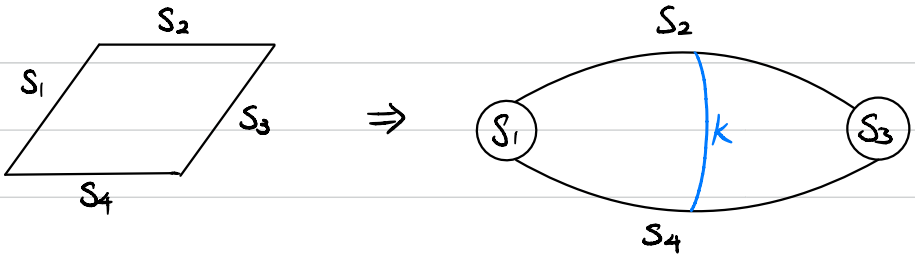
2° Consider faces within the cycle, the other vertex must lies in the graph and not a part of the cycle. Or else it contradict claim 1. () So, we can find another vertex lies inside cycle. Thus, be connecting to it and replace one vertex, we can make # inside - outside smaller.

3° Thus, contradiction ⇒ vertices on edge $= 4\lceil\sqrt{V+1}\rceil$, QED



Inside

(5) 1° Suppose a graph with $\frac{2}{3}V$ strictly inside vertices and $4k$ vertices on the cycle. So we split every k vertices as a group and obtain 4 group S_1, S_2, S_3, S_4



2° Consider S_1, S_2, S_3, S_4 as four groups, to walk from $S_2 \to S_4$ takes a path of distance k (Claim 1, walk on S_1). So, there is a path inside the graph from $S_2 \to S_4$ whose distance is k, meaning there is k points on that path. And to walk from $S_1 \to S_3$, we will cross through that path. So we know the vertex cut of $S_1 \to S_3$ is at least k.

3° From Menger's Theorem and 2° we know, there are also k vertex disjoint path between S_1 and S_3. Also the distance between $S_1 \to S_3$ is k. This suggest that there are at least $k \times k$ vertices on the graph which is $(\sqrt{V+1})^2 = V+1 > V$. So it contradict the hypothesis that there's V vertices on the graph. Thus, there is no way for a Saiko Psycho Cycle to have $\frac{2}{3}V$ strictly inside vertices. QED.