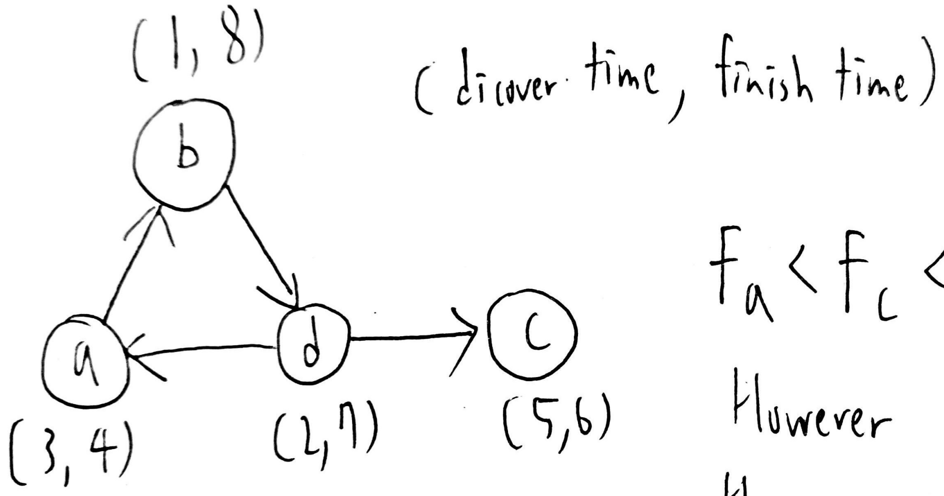
Homework 3

B10902033 林祐辰

References:
Problem 1: B10902032 李沛宸
Problem 2: B10902032 李沛宸
Problem 3: B10902032 李沛宸
Problem 4: B10902032 李沛宸
Problem 5: B10902032 李沛宸
Problem 6: B10902032 李沛宸

Problem 5:
(a) False, counterexample:



$f_a < f_c < f_d < f_b$

However c is not in the same SCC with a, b.

(b) Ans: R13 Minquan W. Rd., Cost: 45$
My method: First I guess that 台北車站 would be the answer, since it looks like the middle of the map. Next I find out that 淡水 has highest fare $50, so I follow the red line to find the answer. Last I find out that there are serveral station around 民權西路 has the same cost to the farthest station which is 45$, so I choose 民權西路 as my answer.

(c) Proof:
1. Because every node in the graph has exactly one out-degree, so starting from any node -> go to the node it's pointing to -> new node -> go to the node it's pointing to .... This process can go forever, since every node has an out-degree.
2. Because the sequence above is infinite, and the nodes in the graph are finite. So we can easily find out that there must be a node that appears at least twice in the sequence. (because the sequence > n)
3. There exist a pointing cycle, since there exist at least two same node in the sequence.

(d) & (e) Algorithm:

1. Prepare the reverse graph G'.
2. Traverse G' by DFS to find every cycle, and save their cycle sequence and cycle size.
3. Run DFS from every vertex in the cycle sequence(don't visit the node on the cycle)
4. When running 3 save the sequence from the vertex on the cycle to leaf (branch sequence), and their distance (branch distance).
5. After getting the information we need we can get the answer for every vertex by the method below.

當 vertex i 在其中一個 cycle sequences 裡:

Seq:存在 i 的 cycle sequence, Size(Seq): Seq 的 cycle size

t: i 在 Seq 中的第幾項

$ans[i] = Seq[((t+Size(Seq)) - K \bmod Size(Seq)) \bmod Size(Seq)]$

當 vertex i 不在任何一個 cycle sequences 裡:

u: 從 cycle 到 i 的出發點, D: 從 u 到 i 的距離

Condition 1: i 走不到 cycle $\Rightarrow K < D$

B_seq: i 的 branch sequence

$ans[i] = B\_seq[D-K]$

Condition2: i 走的到 cycle $\Rightarrow K \geq D$

ans[i] = Same answer for ans[u] when $K = D-K$

Reminder: The sequence is for G' so we need to count backwark to get the answer.

Correctness: The algorithm works by finding the sequence we need and find the answer in O(1).

Time complexity: Reversing the graph cost O(V + E) = O(2*n) = O(n). Traversing while getting information cost O(n) + O(n) = O(n). Total time complexity = O(n).

(f) & (g) Algorithm:

1. Find all strongly connected component.
2. Set the answer of all vertex in SCC infinity, and marked it visited.
3. Run DFS on the vertex that hasn't been visited, if its child answer is infinity its answer is infinity, else its answer = child answer + 1.

Correctness: The algorithm works by changing the graph into a DAG which can simplify the question. By running DFS you can get the answer from its child with easy caculate.

Time complexity: Finding SCC costs O(V + E). Runnig DFS costs O(V + E). Total time complexity = O(V + E).

Problem 6:

(a)

Brief description of algo:

1. maintain two vector: a dango vector and a dango sum vector
2. AA: push an element and sum to the vectors

   BB: pop an element and sum from the vectors

   CC: pop and store x elements, using selection algorithm to find the k-largest elements, then push the ones that are smaller back in the vector and also sum the ones that aren't pushed in. (maintain the sum vector at the same time)

   DD: calculate the sum diff in sum vector

Accounting method:

	actual cost	amortized cost	Credit change
AA	1	k + 1	save k credits in the account
BB	1	1	remain the same
CC	x	0	take x dollars from the remain credits
DD	1	1	remain the same

Valid check:

Assume when CC vector size $= n$, credit change $= k \cdot n - x$

, we know that $n - \lceil \frac{x}{k} \rceil \geq 0 \Rightarrow k \cdot n - x \geq 0$

The upper bound of amortized cost $= O(k)$

Total cost $= O(k) \cdot M = O(kM) = o(k \cdot M^{1.1101420})$

(b)

Potential method:

potential function: $\Phi = \frac{3+\sqrt{5}}{2}n - \frac{1+\sqrt{5}}{2}N$

(n: used positions, N: table size)

$\Phi(D_0) = 0$, $\Phi(D_i) \geq 0$ since the functions drops when resizing

example: $n = F(k)+1$ $N = F(k+1)$, $\Phi = \frac{3+\sqrt{5}}{2}(F(k)+1) - \frac{1+\sqrt{5}}{2}F(k+1) \doteqdot \frac{3+\sqrt{5}}{2}$

amortized costs per-op: $\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$

when Insert without resize:

$$\hat{C}_i = 1 + \left(\frac{3+\sqrt{5}}{2}n - \frac{1+\sqrt{5}}{2}N\right) - \left(\frac{3+\sqrt{5}}{2}(n-1) - \frac{1+\sqrt{5}}{2}N\right)$$

$$= \frac{5+\sqrt{5}}{2}$$

when insert with resizing:

operation	table-size
$n-1$	$F(k)$
n	$F(k+1)$

resize when $n-1 = F(k) \Rightarrow n = F(k)+1$

$$\hat{C}_i = \underbrace{1+F(k)}_{\text{actual cost}} + \left(\frac{3+\sqrt{5}}{2}(F(k)+1) - \frac{1+\sqrt{5}}{2}F(k+1)\right) - \left(\frac{3+\sqrt{5}}{2}F(k) - \frac{1+\sqrt{5}}{2}\cdot F(k)\right)$$

$$= 1 + F(k) + \frac{3+\sqrt{5}}{2} + \frac{1+\sqrt{5}}{2}(F(k) - F(k+1))$$

$$= \frac{5+\sqrt{5}}{2} + F(k) - \frac{1+\sqrt{5}}{2}F(k-1) \doteqdot \frac{5+\sqrt{5}}{2}$$

amortized complexity:

$$T(n) = \sum_{i=1}^{n} \hat{C}_i = \frac{5+\sqrt{5}}{2}\cdot n = O(n)$$

(c)

Algorithm:

1. The initialize tree root is $(1, C, 0)$
2. For every operation (l, r, c), there are two situations
   ① a tree node (a, b, x), which $a \le l, r \le b$
   find it and delete it, next add 3 nodes into the tree
   $(a, l-1, x)$ $(r+1, b, x)$ (l, r, c)
   ② a tree node (a, b, x), which $a \le l \le b \le r$
   , and a tree node (u, w, y), which $u = b+1$
   Find them and delete them, next add 3 nodes into the tree
   $(a, l-1, x)$ $(r+1, w, y)$ (l, r, c)

   Note that: when happen adding a node which $l > r$ simply don't add the node to the tree

Brief explain for time-complexity:

For every operation we find and delete atmost two nodes, and insert atmost three nodes, which means that each operation do to the balanced binary search tree would be $O(\log N)$, since the size of the tree won't exceeds $2N$.

(d)

Aggregate method:

Total cost = Sum of every node cost

node cost = times that the node was chose to be x

$$\text{node cost} = \underbrace{V}_{\text{total nodes}} - \underbrace{1}_{\text{self node}} - \underbrace{a}_{\text{parent nodes}} - \underbrace{b}_{\text{child nodes}} \quad , \quad 1 \le a+b \le V-1$$

, because when a node is chose to be x, the possible y that can pair with it is every node except itself, it's parent, and it's children.

So the upper-bound of node cost = V−2

$$\text{Amortized total cost} = \underbrace{0}_{\text{root cost}} + \underbrace{(V-1)\cdot(V-2)}_{\text{non-root cost}} = V^2-3V+2 = O(V^2)$$

$$\text{Amortized cost per-vertex} = \frac{V^2-3V+2}{V} = O(V)$$

$$\text{total time-complexity} = O(V)\cdot V = O(V^2)$$