Homework #3

(Due: May 1)

Task 1. [100 Points] Randomized Connected Components

A connected component C of an undirected graph G is a maximal subgraph of G such that every vertex in C is reachable from every other vertex in C following a path in G. Figure 1 shows an example.

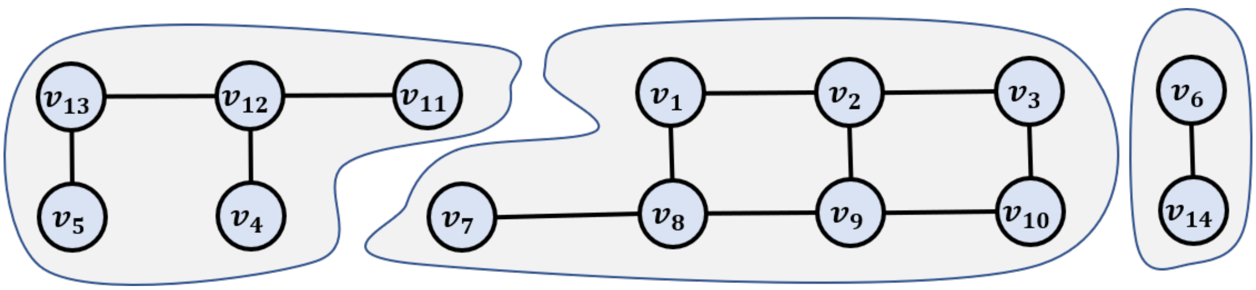

Figure 1: [Task 1] An undirected graph with three connected components.

In this task, we will analyze two randomized algorithms, namely Rand-CC-1 (shown in Figure 2) and Rand-CC-2 (shown in Figure 3), for computing the connected components (CC) of any given undirected graph. For every vertex v of the input graph, both algorithms find the unique id of the connected component containing v. Both algorithms use a function called Rand-Hook (shown in Figure 2) as a subroutine which randomly hooks vertices to their neighbors in such a way that after the function terminates these vertices form a set of disjoint stars. Also, Rand-CC-2 calls Rand-CC-1 as a subroutine.

Pseudocodes (with detailed comments) of Rand-CC-1 and Rand-CC-2 are given in Figure 2 and Figure 3, respectively.

Now answer the following questions.

(*a*) [**10 Points**] Consider the function Rand-Hook($V,\ L,\ N$) in Figure 2, and assume that V has no zero degree vertices. Suppose we start with an empty set Q, and add edges to it as follows. We traverse the vertices in V in some order, and for each $v \in V$ encountered in that order, we add $(v, N[v])$ to Q provided the graph induced by $Q \cup \{(v, N[v])\}$ does not contain a cycle. Prove that $|Q| \geq \frac{|V|}{2}$.

(*b*) [**10 Points**] Consider the set Q constructed in part $1(a)$. We say that an edge $(u, v) \in Q$ is a *hook* provided $C[u] \neq C[v]$ right after step 3 of Rand-Hook. Prove that for all $Q' \subset Q$, if one knows the *hook* status of all edges in Q' that still does not reveal anything about the hook status of any edge in $Q \setminus Q'$.

(*c*) [**10 Points**] Use your results from parts 1(*a*) and 1(*b*) to show that in any invocation of RAND-HOOK(V, L, N), with probability at least $1 - \frac{1}{e^{|V|/32}}$ at least $\frac{|V|}{16}$ vertices of V will change their L values.

(*d*) [**10 Points**] Use your result from part 1(*c*) to prove a high probability bound on the running time of RAND-CC-1.

(*e*) [**20 Points**] Consider RAND-CC-2. The algorithm repeatedly chooses a random sample (of geometrically decreasing sizes) from the edges of the input graph, and uses the sample to identify vertices (or supervertices) that have *potentially high degree* (PHD). Each vertex of the graph starts as a PHD vertex, but loses the status as soon as a chosen sample of edges fails to include an edge connecting that vertex with another PHD vertex. The RAND-HOOK function is called only on the vertices that retain the PHD status. After a sufficient number of such sampling and hooking rounds, the number of edges among the supervertices reduce to a sufficiently small number. At that point connected components among the supervertices are found by calling RAND-CC-1. Now suppose at recursion depth d of RAND-CC-2, n_d denotes the number of vertices/supervertices in V with a PHD status. Then n_0 is the number of vertices in the original input graph. Let $n = n_0$. Prove that for each $d \in [0, d_{max}]$, $n_d \leq \alpha^{2d} \cdot n$ w.h.p. in n, where $\alpha = \sqrt{\frac{15}{16}}$ and $d_{max} = \left\lceil \frac{1}{4} \log_{\frac{1}{\alpha}} n \right\rceil$.

(*f*) [**20 Points**] We call an edge (u, v) *heavy* provided $PHD[u] = PHD[v] = \text{TRUE}$, otherwise we call it *light*. At recursion depth d of RAND-CC-2, let r_d be the expected number of heavy edges that become light. Prove that for each $d \in [0, d_{max}]$, $r_d \leq \alpha^d \cdot n$.

(*g*) [**10 Points**] Prove that in the call to RAND-CC-1 in line 21 of RAND-CC-2, the expected number of edges in E' is $\mathcal{O}(n)$.

(*h*) [**10 Points**] Compute the expected running time of RAND-CC-2.

Task 2. [80 Points] Partial Sums on 2D Grids

This task asks you to solve the *partial semigroup sums* problem on two dimensional (2D) square and hexagonal grids[1].

In the 2D square grid version of the problem you are asked to preprocess an $n \times n$ grid S filled with entries from a given semigroup $(\Pi, \oplus)$ using as little space as possible so that queries asking for the sum of the entries in any given rectangular area r of S can be answered efficiently. Space complexity is measured in terms of the number of values from Π stored in the data structure, and query complexity is measured in terms of the number of times the semigroup operation $\oplus$ is applied when answering a query. You can assume $n = 2^k$ for some integer $k \geq 0$.

In the 2D hexagonal grid version you are required to preprocess a hexagonal grid H of side length n using as little space as possible so that given any hexagonal region h of H one can return the sum of the entries in h as efficiently as possible. You can assume $n = 2^k - 1$ for some $k \in \mathbb{Z}^+$.

Now answer the following questions.

[1] 1D version of the problem was solved in the class during the guest lecture on "the α technique" by Shih-yu Tsai

Rand-CC-1(V, E, L)

(Input is an unweighted undirected graph with vertex set V and edge set E. For each $v \in V$, $L[v]$ is set to v before the invocation of this function. When this function terminates, for each $v \in V$, $L[v]$ contains the unique id of the connected component containing v. We call an edge (u, v) *live* provided $L[u] \neq L[v]$.)

if $|E| = 0$ ***then return*** *{no edge to contract}*

for each $(u, v) \in E$ ***do*** $N[u] \leftarrow v$, $N[v] \leftarrow u$ *{try to associate the edge (u, v) with u and v}*

Rand-Hook(V, L, N) *{hook among vertices in V based on the edges chosen in the previous step}*

$V' \leftarrow \{ \ v \mid (\ (u, v) \in E \ \vee \ (v, u) \in E \ ) \wedge \ v = L[v] \neq L[u] \ \}$ *{collect the non-zero degree roots after hooking}*

$E' \leftarrow \{ \ (L[u], L[v]) \mid (u, v) \in E \ \wedge \ L[u] \neq L[v] \ \}$ *{E' contains only edges among roots, and no duplicate edges and self loops}*

Rand-CC-1(V', E', L) *{recurse on the smaller instance}*

for each $v \in V$ ***do*** $L[v] \leftarrow L[L[v]]$ *{map the solution back to the current instance}*

Rand-Hook(V, L, N)

(Input is an unweighted undirected graph with vertex set V. For each $v \in V$, $L[v]$ is set to v before the invocation of this function. For each $u \in V$, $N[u]$ is set to a v such that (u, v) is an edge in the graph. This function randomly hooks vertices in V to their neighbors in such a way that after the function terminates these vertices form a set of disjoint stars. For each $v \in V$, $L[v]$ is set to u (possibly $u = v$) provided u is the center of the star containing v.)

for each $u \in V$ ***do*** *{for each vertex in V}*
    $C_u \leftarrow$ Random{ Head, Tail } *{toss a coin}*
    $H_u \leftarrow$ False *{record that this vertex has not yet been hooked}*
for each $u \in V$ ***do*** *{for each vertex u in V}*
    $v \leftarrow N[u]$ *{will try to hook u with $v = N[u]$}*
    if $C_u =$ Tail ***and*** $C_v =$ Head ***then*** *{if u tossed Tail and v tossed Head}*
        $L[u] \leftarrow v$ *{make u point to v}*
        $H_u \leftarrow$ True, $H_v \leftarrow$ True *{record that both u and v are hooked}*
for each $u \in V$ ***do*** *{manipulate the coin tosses to hook more in a second try}*
    if $H_u =$ True ***then*** $C_u \leftarrow$ Head *{if u is already hooked, will try to hook unhooked vertices pointing to u}*
    else if $C_u =$ Tail ***then*** $C_u \leftarrow$ Head ***else*** $C_u \leftarrow$ Tail *{if u is not hooked, flip C_u}*
for each $u \in V$ ***do*** *{try to hook again}*
    $v \leftarrow N[u]$ *{will try to hook u with $v = N[u]$}*
    if $C_u =$ Tail ***and*** $C_v =$ Head ***then*** *{if u has Tail and v has Head}*
        $L[u] \leftarrow L[v]$ *{make u point to whatever v is pointing to}*

Figure 2: A randomized algorithm for computing the connected components (CC) of a graph.

Rand-CC-2($V,\ E,\ L,\ PHD,\ N,\ U,\ d$)

(Input is an unweighted undirected graph with vertex set V and edge set E. The recursion depth of the function is given by d which is set to 0 when the function is invoked for the first time. Let n be the number of vertices in the graph when $d = 0$, and $m = |E|$. Each $v \in V$ is an integer in $[1, n]$. Pointers L (label), PHD (potentially high degree), N (neighbor) and U (updated) point to arrays $L[1 : n]$, $PHD[1 : n]$, $N[1 : n]$ and $U[1 : n]$, respectively. For each $v \in [1, n]$, $L[v]$ is set to v, and $PHD[v]$ is set to True before the initial invocation of this function. When this function terminates, for each $v \in V$, $L[v]$ contains the unique id of the connected component containing v. We assume that $\alpha = \sqrt{\frac{15}{16}}$ and $d_{max} = \left\lceil \frac{1}{4} \log_{\frac{1}{\alpha}} n \right\rceil$. We call an edge (u, v) *live* provided $L[u] \neq L[v]$. Edge (u, v) is *heavy* provided $PHD[u] = PHD[v]$, otherwise it is *light*.)

if $d \le d_{max}$ ***then*** — $\{$*need to recurse more to sufficiently reduce #vertices with* PHD *status*$\}$
    $m_d \leftarrow \lceil m \cdot \alpha^d \rceil$ — $\{$*size of edge sample which geometrically decreases with* $d\}$
    $\widehat{E} \leftarrow$ a sample of size m_d chosen uniformly at random from E — $\{$*do not always touch all edges in* $E\}$
    for each $v \in V$ ***do*** $U[v] \leftarrow$ False — $\left\{\text{flag } U[v] \text{ keeps track if an edge in } \widehat{E} \text{ hits } v\right\}$
    for each $(u, v) \in \widehat{E}$ ***do*** — $\{$*check each edge in the sample*$\}$
        $u' \leftarrow L[u],\ v' \leftarrow L[v]$ — $\{$*find the root of the tree containing each endpoint*$\}$
        if $u' \in V$ ***and*** $v' \in V$ ***and*** $u' \neq v'$ ***and*** $PHD[u'] = PHD[v'] =$ True ***then*** — $\{$*if* (u', v') *is live and heavy*$\}$
            $N[u'] \leftarrow v',\ N[v'] \leftarrow u'$ — $\{$*try to associate the edge with* u' *and* $v'\}$
            $U[u'] \leftarrow$ True, $U[v'] \leftarrow$ True — $\left\{\widehat{E} \text{ hits } u' \text{ and } v'\right\}$
    for each $v \in V$ ***do*** — $\{$*check each vertex* v *in* $V\}$
        if $U[v] =$ false ***then*** $PHD[v] \leftarrow$ False — $\left\{\text{if } \widehat{E} \text{ does not hit } v \text{ then } v \text{ loses its } PHD \text{ status}\right\}$
    $\widehat{V} \leftarrow \{\ v \mid v \in V\ \wedge\ U[v] = \text{True}\ \}$ — $\left\{\widehat{V} \text{ contains the vertices from } V \text{ which still have } PHD \text{ status}\right\}$
    Rand-Hook($\widehat{V},\ L,\ N$) — $\left\{\text{hook among vertices in } \widehat{V} \text{ (see Figure 2)}\right\}$
    $V' \leftarrow \{\ v \mid v \in V\ \wedge\ v = L[v]\ \}$ — $\{V'$ *contains only the roots after hooking*$\}$
    Rand-CC-2($V',\ E,\ L,\ PHD,\ N,\ U,\ d + 1$) — $\{$*recurse on the smaller instance*$\}$
    for each $v \in V$ ***do*** $L[v] \leftarrow L[L[v]]$ — $\{$*map the solution back to the current instance*$\}$
endif
if $d = 0$ ***then*** — $\{$*done compressing*$\}$
    $V' \leftarrow \{\ v \mid v \in V\ \wedge\ v = L[v]\ \}$ — $\{$*collect only the root vertices*$\}$
    $E' \leftarrow \{\ (L[u], L[v]) \mid (u, v) \in E\ \wedge\ L[u] \neq L[v]\ \}$ — $\{E'$ *contains only edges among roots, and no duplicate edges and self loops*$\}$
    Rand-CC-1($V',\ E',\ L$) — $\{$*use the algorithm from Figure 2 to solve the problem once the number of edges reduces to a sufficiently small number*$\}$
    for each $v \in V$ ***do*** $L[v] \leftarrow L[L[v]]$ — $\{$*map the solution back to the current instance*$\}$
endif

Figure 3: Randomized connected components (CC) based on edge sampling.

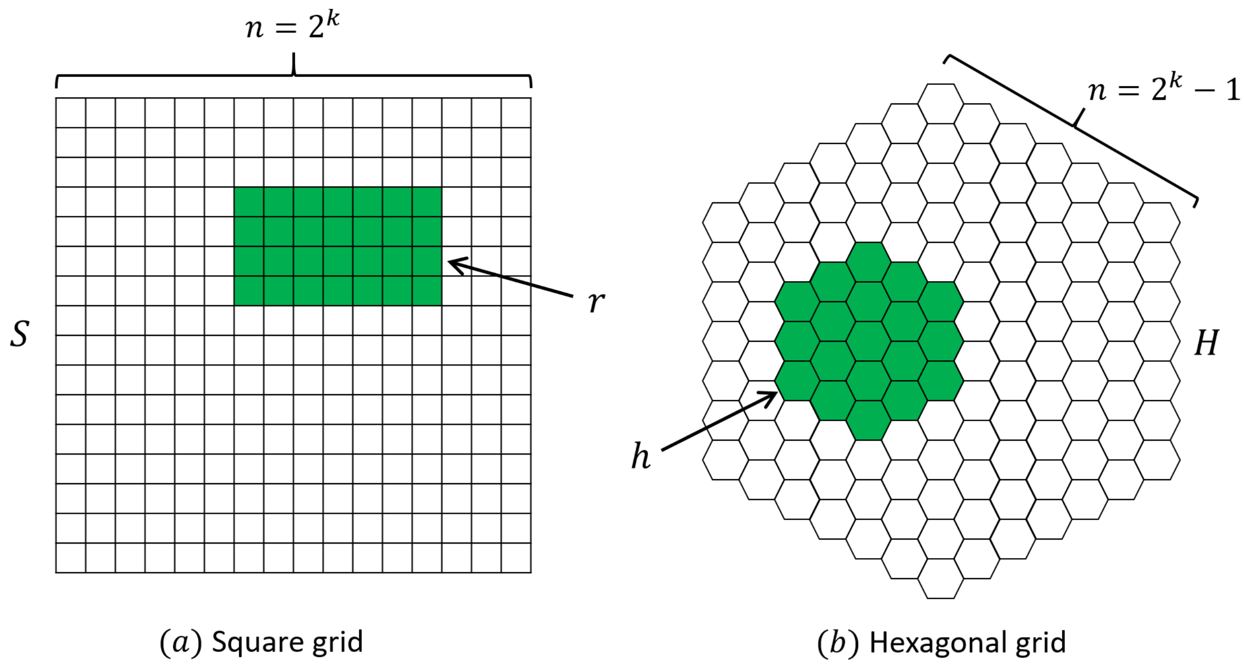

Figure 4: [Task 2] Partial semigroup sum query for (a) a rectangular area r inside a square grid S, and (b) a hexagonal area h inside a hexagonal grid H.

(a) [**15 Points**] Show that the given 2D square grid S can be preprocessed to use $\mathcal{O}\left(n^2 \log n\right)$ space and to answer queries using $\mathcal{O}\left(1\right)$ applications of the semigroup operation.

(b) [**25 Points**] Use the approach shown in the class to extend your result from part $1(a)$ and show that S can preprocessed to use $\mathcal{O}\left(n^2\alpha(n)\right)$ space and answer queries using $\mathcal{O}\left(\alpha^2(n)\right)$ applications of the semigroup operation.

(c) [**15 Points**] Repeat part $1(a)$ for the given hexagonal grid H.

(d) [**25 Points**] Repeat part $1(b)$ for the hexagonal grid H.