準備

定義 1 (Subgraph Aggregation)**.** $G = (V, E)$ をネットワークグラフとし、$\mathcal{P} = (P_1, ...P_{|\mathcal{P}|})$ を part の集合とし、各 P_i について H_i を P_i のノード上の G の連結部分グラフとする。必ずしもグラフ $G[P_i]$ から誘導されるとは限らない。各部分グラフ H_i について、$V(H_i)$ 内のすべてのノードが部分グラフ H_i 内の隣接ノードを認識し、それ以外は何も知らないと仮定する。すべてのノード $v \in \bigcup_i P_i$ が O(log n) ビットの整数 x_v を持ち、$\oplus$ を長さ O（log n）の整数に作用する結合関数とする。P_i 内の各ノードは値 $\bigoplus_{v \in P_i} x_v$、すなわち P_i 内のすべての値 x_v の集合 $\oplus$ を知りたいとする。このようなタスクをオペレーター $\oplus$ における Subgraph Aggregation と呼ぶ。

補題 1. 木幅が高々k のグラフ $G = (V, E)$ と二つの頂点 $s, t \subseteq V$ を与えると、k 点素 s-t パスを見つけるか、サイズ k 以下の s-t ノードカットを $\tilde{O}(k^{O(1)}D)$ ラウンドで出力することができる。前者の場合、すべてのノードは、それがパス上にあるかどうかを知っており、そうであれば、そのパス上のその前方と後方を知る。後者の場合、k 点素パスが存在しないという事実と、ノードカットに含まれるかどうかをすべてのノードが知っている。

系 1. 結合演算子 $\oplus$ について、Q_G がグラフ G とその直径 D に依存するパラメータである場合、$\tilde{O}(Q_G)$ ラウンドで Subgraph Aggregation 問題を解くことができる。

全てのグラフ $G : Q_G = O(\sqrt{n} + D)$

種数 g のグラフ $G : Q_G = O(\sqrt{g+1}D)$

木幅 k のグラフ $G : Q_G = \tilde{O}(kD)$

H をマイナーとして含まないグラフ $G : Q_G = \tilde{O}(f(H) \cdot D^2)$,$f$ は H にのみ依存する関数

系 2. 補題 1 のグラフ G を一般のグラフに置き換えると、最大 ℓ 本の s-t パスは $\tilde{O}(\ell^{O(1)}(\sqrt{n} + D))$ ラウンドで見つけることができる。

補題 2 (全域木)**.** ネットワークグラフの連結部分グラフ $H \subseteq G$ が与えられると、G の全域木を $O(\log n)$SA ラウンドで計算することができる。どのノードもスパニングツリーでの隣接ノードを認識している。

補題 3 (根付き木収集)**.** G の木 T を考える。根 $v_r \in V(T)$ が与えられると、$O(\log n)$SA ラウンドで v_r を根とする木 T を計算することができ、$V(T) - v_r$ の各ノードは v_r を根とする木 T の親を知る。さらに、各ノード v_i が整数 x_i と共通結合演算子 $\oplus$ を知っていれば、各ノード v_i に部分木収集 $\bigoplus_{j \in T(v_i)} x_j$ を学習させることができる。ここで、$T(v_i)$ は v_i をルートとする部分木、すなわち、根への経路に v_i を含む T 内のすべてのノードである。

補題 4 (経路収集)**.** G の有向パス $P = \{v_1, ..., v_\ell\}$ を考える。ここで、各ノード v_i はそのパスの先頭ノードと後尾ノードを知っている。$O(\log n)$SA ラウンドにおいて、各ノード v_i は i の値と経路内のそのインデックスを知ることができる。さらに、各ノード v_i が整数 x_i と共通の結合演算子 $\oplus$ を知っていれば、各ノード v_i に先頭集約 $\bigoplus_{j \leq i} xj$ と後尾集約 $\bigoplus_{j \geq i} x_j$ を学習させることができる。

補題 5 (s-t パス)**.** 連結部分グラフ $H \subseteq G$ と 2 つの頂点 $s, t \in V$ が与えられると、$O(\log n)$SA ラウンドで G における有向 s-t パスを計算することができる。すべてのノードは、自分がパス上にあるかどうか、またある場合はそのパス上の先頭ノードと後尾ノードを認識している。

定義 2. G をグラフとし、$s, t \in V$ とする。 $f_1, ..., f_k$ を G における一対の点素 s − t パスの集合とする。この時、すべての $1 \leq i \leq k$ に対して、$w_i \in f_i$、$s \neq wi \neq t$ であれば、(f_1, f_k) に関してタプル $(w_1, ..., w_i)$ をスライスと呼ぶ。X を V の任意の部分集合とし、$s, t =\in X$ とする。$G[V \backslash X]$ に s-t パスがない場合、X は s と t を分離すると言う。s と t を切り離すスライスをカットと呼ぶ。

定義 3. $f_1, ..., f_k$ を s-t 点素パスの集合とする。$U = (u_1, ...u_k)$ および $W = (w_1, ..., w_k)$ を $(f_1, ..., f_k)$ に関するスライスとし、すべての $1 \leq i \leq k$ について、u_i は f_i における w_i の前者または $u_i = w_i$ とする。この時 U は W より S に近いと言い、$U \preceq W$ と書く。もし $1 \leq i \leq k$ に対して、さらに $u_i \neq w_i$ ならば、U は W より厳密に s に近いと言い、$U \prec W$ と書く。同様に、W は U より厳密に t に近いと言う。便宜上、タプル $(s, s, ..., s)$ と $(t, t, ..., t)$ についても同様に上記を定義。したがって、例えば、$(s, s, ...s)$ はどのスライスよりも s に近いと言える。" $\preceq$ " は一般的な順序の合計を定義するものではない。

定義 4. U を任意のカットとする。s を含む $G[V \backslash U]$ の連結成分の頂点セットとして $V_s(U)$ を定義し、t を含む $G[V \backslash U]$ の連結成分の頂点セットとして $V_t(U)$ を定義し、V_r を $G[V \backslash U]$ の残りの連結成分の頂点集合の和集合として定義する。すなわち、 s も t も含まないものである（したがって、$V_r(u)$ は空であり得る)。

定義 5. $(f_1, ..., f_k)$ に関して U をスライスとする。$U \preceq X$ であり、かつ $U \preceq X' \prec X$ を満たすカット X' が存在しないようなカットを X とする。この時 $U^+ := X$ を定義する。
上記のような X が存在しない場合は、$U^+ := (t, t, ..., t)$ とする。
同様に、$Y \preceq U$ かつ $U \preceq Y' \prec Y$ を満たすカット Y' が存在しないようなカットを Y とする。この時、$U^- := Y$ を定義する。
上記のような Y が存在しない場合は $U^- := (s, s, ..., s)$ とする

補題 6. $g(s), g(t)$ を正の整数とする。$1 \leq i \leq k$ かつ f_i 上で u_i の真後ろにある u^+ について $U = (u_1, ..., u_k)$ と $W = (u_1, ..., u_{i-1}, u^+, u_{i+1}, ..., u_k)$ はスライスであるとするこの時、
$U^+ = (t, t..., t)$ または $|V_s(U^+)| + g(s) > |V_r(U^+) \cup V_t(U^+) + g(t)|$ ならば $W^+ = (t, t, ..., t)$ または $|V_s(W^+)| + g(s) > |V_r(W^+)| \cup V_t(W^+)|$

補題 7. $U \preceq W$ のような U, W を $(f_1, ..., f_k)$ に関するスライスとする。この時、$U^+ \preceq W^+$ かつ $U^- \preceq w^-$ である。

定義 6. (A^*, S^*, B^*) を頂点セパレータとする。この時、反復で選択された頂点 s, t に対して、$s, t \in A$ または $s, t \in B$ である場合、アルゴリズム 5 の while ループの反復は (A^*, S^*, B^*) に関して失敗する。そうでなければ、(A^*, S^*, B^*) に関して反復は成功したという。