進捗管理・報告(2020/10/22)

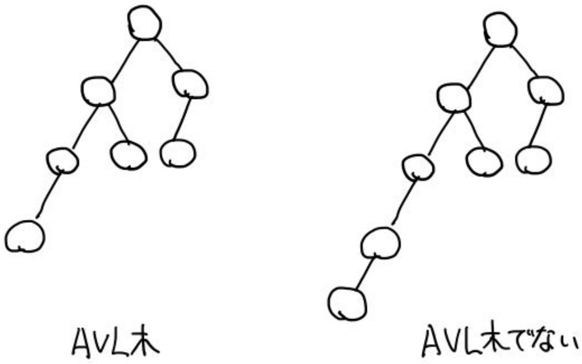
1. 現在取り組んでいること
可逆コンピューティングの研究

2. 進捗状況
・可逆コンピューティングについてと、この分野で行われてきたことの調査

3. 前回からの進捗
・**反復深化深さ優先探索**のAVL木における計算量を見積もった
ーAVL木とは、以下の性質を満たす木のこと
・2分木
・全ての節点について、その子の木の高さの差が1, 0, -1のいずれかである

ーAVL木を用いることで木の高さが平衡になり、探索を効率的に行うことができる

例：



ーAVL木の高さは O(lg n) である
T(h)を高さhの時のAVL木の最小の節点数であるとすると、
T(h) >= T(h-1)+T(h-2)+1 (* T(0)=0, T(1)=1)
が満たされる。
これを解くと、T(h) = ⌊φ^h / sqrt(5) + 1 / 2⌋(* φ = (1+sqrt(5)) / 2)
ここで T(h) <= n であるので、
⌊φ^h / sqrt(5) + 1 / 2⌋<= n
φ^h <= sqrt(5) * (n + 1 / 2)
h <= (lg(sqrt(5)) + lg(n + 1 / 2)) / lg(φ) = O(lg(n))
(以上の証明はこちらから引用しました)

ー したがって、前回の完全二分木の式がほぼそのまま利用できる
(床関数(lg n) の箇所を lg n にするだけでいいはず...)

$$\sum_{i=1}^{\lfloor \lg n \rfloor+1} \left(\sum_{j=1}^{i} 2^{j-1} \right)$$

$$= \sum_{i=1}^{\lfloor \lg n \rfloor+1} (2^i - 1)$$

$$= \sum_{i=1}^{\lfloor \lg n \rfloor+1} 2^i - (\lfloor \lg n \rfloor + 1)$$

$$= \left(\sum_{i=1}^{\lfloor \lg n \rfloor+1} 2 \cdot 2^{i-1} \right) - (\lfloor \lg n \rfloor + 1)$$

$$= (2^{\lfloor \lg n \rfloor + 2} - 2) - (\lfloor \lg n \rfloor + 1)$$

$$= 2^{\lfloor \lg n \rfloor + 2} - \lfloor \lg n \rfloor - 3$$

※ ここで、$\lfloor \lg n \rfloor + 2 \leq \lg n + 2 = \lg 4n$.

したがって、

$$2^{\lfloor \lg n \rfloor + 2} - \lfloor \lg n \rfloor - 3 \leq 2^{\lg 4n} - \lfloor \lg n \rfloor - 3$$
$$= 4n - \lfloor \lg n \rfloor - 3$$
$$= O(n)$$

以上より、AVL木における可逆反復深化深さ優先探索の計算量は O(n) である。

・木構造における反復深化深さ優先探索まとめ
ー 時間計算量：(Bennett法なしで)非可逆な方法に線形時間
ー 空間計算量：O(1)
ー ゴミ出力量：0

・グラフにおける可逆深さ優先探索の計算量まとめ
ー 閉路の存在するグラフ → T: O(V+E), S: O(V), G: O(V) or 0*(Bennett法)
ー 閉路の存在しないグラフ → T: O(V+E), S: O(1), G: 0

グラフにおける深さ優先探索では、訪問印を作成せざるを得ないため、空間計算量は O(V) になる。この訪問印は逆計算の際に必要になるため、削除するには Bennett法を用いて 2 回走査を行う。

木構造の深さ優先探索に用いてきた右手法の考え方は、無向グラフであれば活用できると考えられ、その場合、辿ってきた道のりを保存する必要は無いため、卒論のアルゴリズムと比較してわずかな量の空間使用量を削減できる。(可逆アルゴリズムに限ったことではないのですが…)

4. 今後の課題

・深さ優先探索以外の他のグラフアルゴリズムについて