アルゴリズムの解析

原田 大瑚

2024/03/31

1　問題

問題 π は無限である入力集合 I と質問 $Q(x)$ から成り立っている。入力集合の要素 $s(\in I)$ を入力といい、その大きさを $|s|$ で表せる。問題例は $\Pi(s) = Q(s)$ のように表現する。また、入力が部分集合の場合の問題は部分問題と呼ばれる。また、「yes」あるいは「no」である問題を判定問題といい、判定問題に対して探索問題と最適化問題という付随する問題を定義できる。探索問題とは判定問題の答えが「Yes」である証拠を示す問題であり、最適化問題は判定問題に関連するある値を最適化する問題である。

2　巡回セールス問題

巡回セールス問題は以下のような最適化問題として定義できる。
入力: 完全グラフ K_n、重み関数 w:$E(K_n) \to R_+$
質問: ネットワーク (K_n, w) の重み最小ハミルトン閉路をひとつ示せ。

3　EG の探索問題と最適化問題

(Π_1) 次に示す問題入力: グラフ G
質問:G はオイラーグラフか。
に付随する探索問題と最適化問題を示せ。

オイラーグラフとはオイラー閉トレイルを持つグラフなので、G がオイラー閉トレイルを持つことを示せばよい。オイラー閉トレイルは G のすべての辺を一度だけ通るので、G の最大閉トレイルと言い換えられる。よって解は
(Π_2) 探索問題:
入力: グラフ G
質問:G がオイラーグラフならば、オイラー閉トレイルを 1 つ示せ。
(Π_3) 最適化問題:
入力: グラフ G
質問:G の長さが最大の閉トレイルを 1 つ示せ。

$\Pi 1, \Pi 2, \Pi 3$ をそれぞれハミルトン判定問題 (HG)、ハミルトン閉路問題、最大閉路問題という

例題 1 m 辺から成るグラフ H に対して、問題 $\Pi_1 = (g, Q_1(G))$ を高々$m+1$回解くことにより、問題例 $\Pi_2(H)$ を解けることを示せ。*解:* まず、問題例 Π_1 が「No」であるとき Hはハミルトングラフではないから、問題例 $\Pi_2(H)$ においてハミルトン閉路を示す必要がない。したがって、$\Pi_2(H)$ は解けるということである。Π_1 が「yes」の時、すなわち H がハミルトン閉路を含むときである。この時、H から辺を一つづつ除去した時、除いたグラフがハミルトン閉路を含まない時、除いた辺はハミルトン閉路に必要不可欠な要素といえる。この辺がないと H のハミルトン閉路が成り立たないのでこの辺は除去せず残す。逆に除いたグラフがハミルトン閉路を含んだ時、除いた辺はハミルトン閉路に必要不可欠な要素とは言えない。よってこの辺は除去する。この操作を H のすべての辺に行うと最終的に残ったグラフが Hのハミルトングラフである。この時、$\Pi_2(H)$ は Π_1 を $m+1$ 回解くことにより解けたこととなる。

3.1　アルゴリズムの解析

アルゴリズムの性能は計算量として測定される。アルゴリズムの計算量としては、アルゴリズムを実行するために必要なメモリの大きさを表す領域計算とアルゴリズムを実行するために必要な時間の長さを表す時間計算量とが主に用いられる。以下では、アルゴリズムの計算量について時間計算量に焦点をしぼり説明する。計算機上でアルゴリズムを実行するのに必要な時間を時間計算量として用いることはできるであろうか。計算機上でアルゴリズムを実行するのに必要な時間は計算機の構造や性能に依存するだけではなく、計算機の進歩は目覚ましく、基本的な演算をするのに必要な時間は年々短くなっている。したがって、アルゴリズムの不変的な評価やアルゴリズム間の性能評価をしようとした場合には適さない。したがって、以下のような方法で時間計算量を定めることになる。まず、実際の計算機ではなく、基本的な処理を単位時間で実現できる理想的な計算機を計算機モデルとして定義し、その計算モデルでの実行時間を時間計算量として用いる。

以下では、ランダムアクセスメモリ機械 (random access machine, RAM) を計算モデルとして、RAM 上でアルゴリズムを解析する方法を説明する。RAM は、データを処理する CPU とデータを蓄えるランダムアクセスメモリから成る。ランダムアクセスメモリは単にメモリともいう。メモリは任意の大きさの数をデータとして格納できるメモリセルから成り、その大きさは含まれるメモリセルの数で評価される。CPU は、メモリに含まれるメモリセルの数に関わらず、任意のメモリセルに定数時間でアクセスでき、データの大きさに関わらず、基本的な算術演算や比較などを定数時間で計算できると仮定する。算術演算、比較、およびメモリへのアクセスなどの基本操作の有限系列を手続きという。問題Πのすべての問題例に対して解を求めるための有限時間で停止する手続きがΠを解くアルゴリズムである。アルゴリズムの計算量は一般に、最悪の場合を用いて評価する。問題 $\Pi = (I, Q(x))$ を解くアルゴリズムを A と使用。A に入力された入力 $s(\in I)$ が与えられたときにAが実行する基本操作の最大数、すなわち、問題例 $\Pi(s)$ を解くときに必要な基本操作の最大数を $\tau_A(s)$ で表す。この時、$T_A(n) = max\{\tau_A(s) \mid s \in I, |s| = n\}$ をアルゴリズム A の時間計算量と定義し、A は Π を $T_A(n)$ 時間で解くという。

3.2 多項式時間アルゴリズム

アルゴリズム A の時間計算量 $T_A(n)$ は、入力の大きさ n の関数であり、A で Π を解くときに必要な最悪の計算時間に対応する。一般に、$T_A(n)$ はそのオーダで評価される。$T_A(n)$ が線形オーダであるとき A を線形時間アルゴリズム、多項式オーダであるとき多項式時間アルゴリズム、指数オーダであるときは指数時間アルゴリズムなどという。一般に、多項式アルゴリズムは入力の大きさに応じて計算時間が増加するがその変化は比較的緩やかである。一方で、指数時間アルゴリズムは比較的小さな入力に対してさえも天文学的な長さの時間を必要になる。

4 グラフの大きさ

グラフの大きさはどのように評価すればよいでしょうか。メモリセルを任意の大きさの数を格納できると仮定し、グラフを格納するために必要なメモリセルの数でその大きさを評価する。グラフを計算機上で格納するのによく用いられるデータ構造の中で、隣接行列と接続行列がある。行列を格納するメモリセルの数は、行列の各要素のデータを格納するためのメモリセルと、行列の行数と列数を格納すためのメモリセルの和として表せられる。すなわち、行列の要素数を x とした時、必要なメモリセルの数は $a_1x + a_2$ となる。これをオーダで表すと $O(x)$ となる。つまり、少なくとも $O(x)$ のメモリセルがあれば行列を格納することができる。また少なくとも $\Omega(x)$ のメモリがないと格納できないことが分かる。このことから n 点と m 辺から成るグラフの隣接行列と接続行列の大きさは $\Theta(n^2)$ と $\Theta(nm)$ であることが分かる。それでは、グラフを計算機上に格納する大きさがより小さいデータ構造あるだろうか。隣接行列では対応する辺があれば 1 で示し、なければ 0 で示す。しかし、どちらか一方のみをデータとして格納すれば、データ構造として十分である。隣接リストは、隣接行列の 1 を表現するデータ構造として、点集合を配列で表現し、各店に隣接する点の集合をポインタを用いた連結リストで表す。

定理 1 隣接行列の大きさは $\Theta(n^2)$ である。

定理 2 接続行列の大きさは $\Theta(nm)$ である。

定理 3 隣接リストの大きさは $\Theta(n + m)$ である。。

証明: 隣接行列と接続行列の大きさは上記の通りである。隣接リストの大きさは以下のようにして評価できる。まず、G の点集合を表現する配列の大きさは $\Theta(n)$ である。点 v に隣接する点の集合を表現するためのリストは $\Theta(deg_G(v))$ であるので、すべての点について隣接する点の集合を表現するための連結リストの大きさは $\Theta(\sum_v \in V(G) deg G(v))$ である。$\sum_{v \in V}(G) deg_G(v) = 2m$ だから、$\Theta(\sum_{v \in V}(G) deg_G(v)) = \Theta(m)$

定理 4 n 点と m 辺から成る任意の連結グラフ G に対して、

$$m = \Omega(n), m = O(n^2)$$

である。また、

$$n = \Omega(m^{\frac{1}{2}}), n = O(m)$$

である。

証明:G は連結なので $m > n-1$ が成り立つ。したがって、

$$\lim_{n\to\infty} \frac{m}{n} \leq \lim_{n\to\infty} \frac{n-1}{n} = \lim_{n\to\infty} 1 - \frac{1}{n} = 1$$

$$\lim_{m\to\infty} \frac{n}{m} \geq \lim_{m\to\infty} \frac{m+1}{n} = \lim_{m\to\infty} 1 + \frac{1}{m} = 1$$

であるから、$m = \Omega(n)$、および $n = O(m)$ を得る。また、n 点から成る完全グラフ K_n の辺数は $n(n-1)/2$ であることが知られている。よって

$$m \leq \frac{n(n-1)}{2}$$

である。したがって、

$$\lim_{n\to\infty}\frac{m}{n^2}\leq\lim_{n\to\infty}\frac{1}{2}(\frac{n^2-n}{n^2})=\lim_{n\to\infty}\frac{1}{2}(1-\frac{1}{n})=\frac{1}{2}$$

$$\lim_{n\to\infty}\frac{n}{m^{\frac{1}{2}}}\geq\lim_{n\to\infty}\frac{1+\sqrt[2]{1+8m}}{2\sqrt[2]{m}}=\lim_{n\to\infty}\frac{\sqrt[2]{1+8m}}{2\sqrt[2]{m}}$$
$$=\lim_{n\to\infty}\frac{\sqrt[2]{\frac{1}{m}+8}}{2}=\sqrt[2]{2}$$

であるから、$m=O(n^2)$、および $n=\Omega(m^{\frac{1}{2}})$ を得る。

定理 5 隣接行列の大きさは $\Theta(n^2)$ である。接続行列の大きさは $\Omega(n^2)$ かつ $O(n^3)$ である。隣接リストの大きさは $\Omega(n)$ かつ $O(n^2)$ である。

5 連結オイラーグラフ判定問題 (C-EG)

連結オイラーグラフ判定問題 (C-EG)
入力: 連結グラフ G
質問:G はオイラーグラフか。
一般に、離散構造に関する問題は「すべての場合をつくす」という自明なアルゴリズムを用いることが多いがこのようなアルゴリズムは多項式時間アルゴリズムでないことが多く、「すべての場合をつくす」というアルゴリズムだけでは問題を解けたとは言えない。実際に、連結オイラーグラフ判定問題 (C-EG) のすべての場合をつくすアルゴリズムについて考える。

1. まず、グラフ G の辺集合 $E(G)$ のすべての順列を生成する。
2. 次に、各々の順列がオイラー閉トレイルに対応しているか否かを調べる。

まず、$E(G)$ のすべての順列を生成できたとする。ある順列がオイラー閉トレイルに対応しているかは、すべての辺で一方の端点が順列の前の辺の端点と一致し、もう一方の端点は順列の次の辺の端点と一致することを確かめればよい。これは、ある順列に対してオイラーグラフに対応する順列かを判定するには辺数を m としたとき $O(m)$ 時間だけかかる。しかし、順列の数は $m!$ 個あるので少なくとも $\Omega(m!)$ 時間はかかってしまう。これは多項式時間ではないので、このアルゴリズムは多項式時間アルゴリズムではない。
そこで、次のようなアルゴリズムを活用する。

5.1 連結オイラーグラフ判定アルゴリズム

入力: 連結グラフ G(ただし、$|V(G)|=n, |E(G)|=m$ とする)
出力:「Yes」または「No」
ステップ 0: $X=V(G)$ とする。
ステップ 1: $X=\emptyset$ ならば「Yes」を出力して終了する。
ステップ 2: 点 x を X から任意の 1 つ選び、$X=Xnx$ とする。
ステップ 3: $deg_G(x)$ が奇数ならば「No」を出力して終了する。そうでなければ、ステップ 1 に戻る。

このアルゴリズムは

定理 6 連結グラフ G がオイラーグラフであるための必要十分条件は、G のすべての点が偶点であることである。

に基づいている。つまり、すべての点が偶点ならば、オイラーグラフであると言える。以上から、このアルゴリズムは、すべての点の次数を調べ、オイラーグラフかを判定するアルゴリズムである。
このアルゴリズムの時間計算量について考える。ステップ 0 は点集合を X に代入する操作である。これは X の大きさだけかかるので $O(n)$ 時間で実行できる。ステップ 1 は集合が空か否かを判定するだけなので $O(1)$ 時間で実行でき、ステップ 2 も集合から x を取り除くだけなので $O(1)$ 時間で実行できる。ステップ 3 はグラフが隣接リストで表現されていたとすると、点 x のリストは $O(deg_G(x))$ 時間で実行できることがわかる。
次に、各ステップが実行される回数と全体の時間計算量について考える。ステップ 0 は最初に一度実行されるだけなので、ステップ 0 の時間計算量は $O(n)$ 時間である。また、ステップ 1 は $n+1$ 回実行され、ステップ 2 は最大で n 回繰り返されるので、それぞれの全体の時間計算量は $O(n)$ 時間である。次に、ステップ 3 は n 回繰り返されるので、全体の時間計算量は $O(n)\cdot O(n)=O(n^2)$ 時間と考えることもできるが、

$$O(\sigma_{v\in V(G)}deg_G(v))$$

と評価することができる。よって

$$\sigma_{v\in V(G)}deg_G(v)=O(m)$$

であるから、ステップ 3 の時間計算量は $O(m)$ 時間であることが分かる。したがって、アルゴリズム全体の時間計算量は $O(n)+O(m)=O(n+m)$ である。また、G は連結なので、$n=O(m)$ であることが分かり、アルゴリズム全体の計算量は $O(m)$ になる。