4.5.7 NP 完全問題

既述のように，重み付きグラフの 2 頂点間の最短道は効率良く求めることができる．では，最短ハミルトン閉路についてはどうであろうか？

重み付きグラフ (辺がない頂点間には重みが ∞ の辺があるものと考える) の重みの和最小のハミルトン閉路を求める問題は**巡回セールスマン問題**と呼ばれている．セールスマンの受持地域内にあるすべての市を頂点とし，市と市の間の距離を辺の重みとするグラフを考える．セールスマンがオフィスがある市から出発して区域内のすべての市をちょうど 1 回ずつ訪問して元の市に戻ってくるような順路のうち，総距離が最小となるものを見つける問題である．この問題 (および，その特殊な場合である「与えられたグラフ (すべての辺の重みが 1 である重み付きグラフと見なすことができる) がハミルトン閉路を持つか否か」を判定する問題) は，それを解くための効率の良いアルゴリズムが知られていない．出発した市から同じ市へ戻ってくるすべてのハミルトン閉路 (頂点数を n とすると最大 $(n-1)!$ 個ある) を調べてその中で重みの和が最小のものを選ぶという，すべての場合を尽くしてみる方法しかないであろうと予想されている．

本書の範囲を超えるので厳密な定義はしないが，この種の問題を NP **完全問題**といい，今日，何千何万という問題が NP 完全であることが知られている*.

● **精度を犠牲にして効率を求める**　一方，効率良い解法がないかもしれない NP 完全問題や，NP 完全問題以上にむずかしいことがわかっている問題を効率的に解くためには，解の精度を落として解く**近似アルゴリズム**とか，正しさ

---

*NP 完全問題：NP-complete problem．例えば，与えられたグラフが 3 色で彩色可能かどうかを判定する問題，与えられた 2 つのグラフの一方が他方の部分グラフであるかどうかを判定する問題，グラフ上の 2 点間に一定長以上の道が存在するか否か判定する問題，ある種の条件を満たす時間割を組む問題，等々．NP は nondeterministic polynomial time (予測を許すタイプのアルゴリズムで，実行時間が問題の大きさの多項式以下で解くことができるもの，という意味) の頭文字である．

厳密に言うと，巡回セールスマン問題のように，ある種の基準の下で最大あるいは最小のものを求める問題 (**最適化問題**(optimization problem) と呼ばれる) と，ハミルトン閉路問題のように，ある条件を満たすものが存在するか否かを判定する問題 (**決定問題**(decision problem) と呼ばれる) とは別のものであり，NP 完全問題は決定問題に対して定義されるものである．とはいえ，最適化問題は決定問題として表わすことができるので，本書の範囲では厳密に区別しなくてもよい．

の精度を確率 1 よりも落として解く**確率アルゴリズム**等がある．例えば，重み付き完全グラフ G が与えられたとき，G におけるハミルトン閉路の中で辺に付けられた重みの和が最小のものを求めるための次に述べる方法は最近傍法と呼ばれている近似アルゴリズムである†．

procedure (G)　/* G のハミルトン閉路を求める */
begin
1. 任意に選んだ 1 頂点から出発する．
   この頂点 1 つだけからなる長さ 0 の道を P とする．
   P を第 2 ステップで述べる方法に従って 1 頂点ずつ延長する．
2. x をこの道に付け加えられた最新の頂点とする．
   P の上にない頂点と x とを結ぶ辺の中で重みが最小の辺を xy とする．
   y をこの道に追加する．$V(P)$　$V(G)$ なら 2 へ戻る．
3. 出発点と最後に付け加えられた頂点とを結ぶ辺の中で重み最小のものを選んで終了する．

end

位数 n の完全グラフにおいて，最近傍法で得られたハミルトン閉路の重みの和を d，最小ハミルトン閉路の重みの和を d_0 とすると，$d \leqq d_0\,(\lceil \log_2 n \rceil + 1)/2$ であることが知られている．

一方，n が奇数の合成数ならば，$a^{n-1} \not\equiv 1 \pmod{n}$ が成り立つか，または，$2^i \mid (n-1),\ 1 < \gcd(a^{(n-1)/2^i} - 1, n) < n$ を満たす整数 a, i が存在すること (この性質をAとする) が知られている．これを使った次のアルゴリズムは，出力した答が正しい確率が $\frac{1}{2}$ よりも大きい確率アルゴリズムの例である．

function primality(n)　/* n が素数かどうか判定する */
begin
1. n が偶数なら “n は合成数である”と出力して終了する。
2. $0 < a < n$ なる整数 a をランダムに求める．
   2.1. もし a, n に対し性質Aが成り立つなら，“n は合成数である”と出力して終了する．
   2.2. そうでなかったら，“n は素数でない”と出力して終了する．

end

†近似アルゴリズム：approximation algorithm．確率アルゴリズム：probabilistic algorithm.