11　組み合わせ最適化問題／補遺

11.1　優れたアルゴリズムの例：Dijkstra 法

ここでネットワークの最短経路問題に対する優れたアルゴリズムの例を紹介する．

> N 個のノード（頂点）からなる有向グラフを考え，ノード i から j へと向かうアーク（辺）(i,j) の長さが d_{ij} で与えられているとする．ただし，d_{ij} は非負であるとする．このとき，ノード 1 から他のすべての頂点に向かう最短経路を求めよ．ただし，ノード 1 から他のノードへの経路は存在するものとする．

全数探索的なアプローチでは，アークの数が増えるに従って計算の手間は大きく増大する．ダイクストラ (Dijkstra)[1]は以下の手順によりこの問題が解けることを示した．

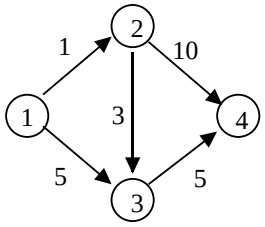

図 1: 最短経路問題のネットワークの例

【アルゴリズム・ダイクストラ】

1. ノード 1 に永久ラベル $u_1^* = 0$ を与える．他のすべてのノード j に一時ラベル $u_j = \infty$ を与える．$i = 1$ とする．すべてのノードに永久ラベルが付くまで，以下の手順 2,3 を繰り返す．
2. ノード i を始点に持つすべてのアーク (i,j) に対し，終点 j が永久ラベルを持っておらず，一時ラベル u_j が $u_i^* + d_{ij}$ よりも大ならば，この一時ラベルを $u_i^* + d_{ij}$ の値に改訂する．
3. 一時ラベル全体の中で最小の値を持つものを 1 個選び，そのノードを i とする．その一時ラベル u_i を永久ラベル u_i^* に変える．

このアルゴリズムの中で，永久ラベル u_j^* は，ノード 1 からノード j までの最短経路の長さを表している．また，最短経路を求めるには，上記の手順 2 で一時ラベルを更新した際に使われた始点を記録しておくことで導出が可能となる．

11.2　計算量の増加と計算時間

問題の大きさ n に対して，それを処理するのに必要な加算回数がちょうど n，$log_2 n$，n^2 となる 3 つのアルゴリズムを考える．すなわち，それぞれの計算量のオーダは順に $O(n), o(log_2 n), O(n^2)$ である．また，加算 1 回につき，1ms かかることにしよう．問題の大きさの増加に対する，それぞれアルゴリズムでの計算量の増え方は，表 1 のようになる（川合著「コンピュータ科学」より抜粋）．

今，あるデータ処理のアルゴリズムを開発したとする．A 君は $O(\log_2 n)$ のアルゴリズムを考えてパソコン上で実行し，$2^{10} = 1,024$ 個のサンプルデータに対して 1 秒で終了した．B 君は $O(n^2)$ のアルゴリズムを考えて，同様に 2^{10} 個のサンプルデータに対してスーパーコンピュータで実行したところ，0.01

[1] E.W.Dijkstra(1930-)．オランダの計算機科学者．構造化プログラミングの提唱者．1972 年度のチューリング賞受賞者．

表 1: 問題の大きさの増加と計算量の増加

n	処理時間	$\log_2$	処理時間	n^2	処理時間
2	0.002 秒	1	0.001 秒	4	0.004 秒
16	0.016 秒	4	0.004 秒	256	0.256 秒
64	0.064 秒	6	0.006 秒	4,096	4.1 秒
256	0.256 秒	8	0.008 秒	65,536	1 分 5 秒
1,024	1 秒	10	0.010 秒	1,048,576	17 分 28 秒
4,096	4.1 秒	12	0.012 秒	16,777,216	4 時間 40 分
16,384	16.4 秒	14	0.014 秒	268,435,456	3 日 2 時間 34 分
65,536	1 分 5 秒	16	0.016 秒	4,294,967,296	49 日 17 時間
262,144	4 分 22 秒	18	0.018 秒	68,719,476,736	2 年 65 日
1,048,576	17 分 28 秒	20	0.020 秒	約 1,100,000,000,000	34 年 310 日

秒で終了したとする．先ほどの表のコンピュータと比較をすると，A 君のコンピュータは 100 倍遅く，B 君のコンピュータは 10^8 倍速いと考えられる．A 君，B 君はともに十分に高速な処理が出来たと思って，本格的に $10^6 = 1,000,000$ 個のデータに対して適用した．すると，A 君のパソコンは約 2 秒で答えを返したのに対して，B 君のスーパーコンピュータはいっこうに答えを返さない．答えを返してくるのは，約 11,000 秒後であり，これは実に 3 時間余り経過してからなのである！

ここでは時間計算量で議論したが，空間計算量に関しても同様の配慮が必要である．近年，主記憶や外部補助記憶の大容量化が進み，空間記憶量に対するプログラミング上の制約が緩くはなったが，大規模な問題を取り扱う場合には十分に注意することが必要である．たとえば，3 次元形状を扱う問題を考える．10cm 立方の領域を 1mm の小立方体の集まりとして表現し，それをそのまま対象のデータとして取り扱うアルゴリズムを考えると $100 \times 100 \times 100 = 10^6$ の小立方体が必要となる．すなわち，立方体の 1 辺の分割数を問題の大きさと考えると，これは $O(n^3)$ の空間計算量となる．100 万という数はコンピュータにとっては決して大きな数ではないが，10cm 立方というのを 10 倍にして 1m 立方にすると 1000 倍の空間計算量を必要とする[2]．

11.3 連続から離散へ：計算幾何学の例

ここでアルゴリズムを工夫することの威力を示す例として，**計算幾何学 (computational geometry)** の分野の代表的な問題である「**串刺し問題**」を紹介する[3]．

> 平面上に n 本の線分が与えられているとする．これらの線分（端点も含む）のすべてを貫く直線が存在するかどうかを判定し，存在する場合にはそのような直線を一つ示せ．

諸君ならばこの問題をどう解くか？すぐに思いつく「直線の方程式 $y = ax + b$ の a, b を色々と変えて調べる」というのではアルゴリズムにならない．それは a, b は連続した実数であり，a, b の全ての組み合わせを調べ尽くすことはできないからである．また，適当に離散化したところで，存在している解を見つけられる保証はない．計算幾何学での成果はこれを $O(n^3)$，さらには $O(n \log n)$ で解くアルゴリズムを与えている．

[2] これに対しては，必要以上には空間を細かく分割することをしない，Octree という手法がある．
[3] 詳しく知りたい人は，杉原厚吉：「計算幾何学と空間推論」，人工知能学会誌, Vol.12, No.2 を見よ．