```
    }
  }
```

このプログラムを実行してみると効率のよいことに驚かされるが，その理由は 2 つのループの中の “if (b == c) break;” に隠されている．内側のループでは頂点 a に隣接している点を近い順に探索しているが，この行のために枝 ac が枝 ab よりも長いときには探索が打ち切られる (課題 2).

6 禁断探索法

2-opt 法は十分に強力なヒューリスティクスであるが，古典的なアプローチに属し，我々の直感に矛盾するところはない．これに対して最近では，直感的には間違った探索を許すことでさらによい解を得ようとする試みも盛んである．ここでは，その**モダンヒューリスティクス**とか**メタ戦略**と総称されるアプローチの一つである**禁断探索**を紹介しよう．

局所探索で引き合いに出した山登りでは，夜のうちにできるだけ高い地点に到達することが目的であった．今度は少しルールを変え，「できるだけ高い地点を通過する」ことを目的にしよう．このルールのもとでは，たとえ懐中電灯で照らせる範囲に今よりも高い所がなくなっても休むことはできず，歩き続けなければならない．そして，周りの風景を記憶しておいて，通過したことのある場所へはなるべく戻らないようにするはずである．この風景の記憶保持に用いられるのが**禁断リスト**である．

禁断探索では，近傍 $N(\mathbf{x})$ から禁断リスト Ξ を除いた中から最も標高の高い場所へ移動する．そこで移動先を示す関数を

$$\mathrm{move}(\mathbf{x}) = \begin{cases} \mathbf{x}' \in N(\mathbf{x}) \setminus \Xi & \text{if } c(\mathbf{x}') \geq c(\mathbf{y}) \text{ for all } y \in N(\mathbf{x}) \setminus \Xi \\ \emptyset & \text{if } N(\mathbf{x}) \subset \Xi \end{cases}$$

と定めれば，以下のように疑似アルゴルを記述できる．なお，TL は禁断リスト Ξ の長さを表す.

algorithm tabu_search
begin
  $t := 0$; $\Xi := \emptyset$;
  $\mathbf{x}_0 :=$ any initial feasible solution;
  $TL :=$ a given positive integer;
  while stopping-criterion = *false* **do begin**
    $\mathbf{x}_{t+1} := \mathrm{move}(\mathbf{x}_t)$;
    $\Xi := \Xi \cap \{\mathbf{x}_t\} \setminus \{\mathbf{x}_{t-TL}\}$;
    $t := t + 1$

end;
return the best **x** in Ξ
end;

6.1 禁断リスト

さて，巡回セールス問題を禁断探索で解く場合，近傍は2-opt 法と同様に定義すればよいとして，禁断リストには何を入れるべきだろうか？ もちろん解そのものを入れることもできるが，ここではもっと簡便に頂点を保持することにする．つまり，2-opt 法で枝 ab, cd を除いて ac, bd を加えたとき (言い換えると関数 Flip(a, b, c, d) を呼んだとき)，頂点 a を禁断リストに加えて，しばらくの間は巡回路の改善を行う枝の始点としては使わない．

禁断リストは，頂点集合 V に対応する 1 次元配列 LS[] を用意し，リストの中の頂点に対してはその寿命 (Life Span) を書き込むことで表現する．最初に LS[] には 0 を入れておき，Flip(a, b, c, d) を呼んだときに LS[a] に正の数 TL を入れる．各反復ごとに正の LS をもつ頂点はその値を 1 つ減らし，LS[a] = 0 になったら，頂点 a は禁断リストから外れ，再び巡回路の改善に使うことができるものとする．

6.2 プログラムと考察

禁断探索法は，2-opt 法と同様に任意の巡回路 tour[] およびその長さ length を入力として以下のように記述できる．

```
tabu_search()
{
  int i, j, a, b, c, d, tmp;
  int iter, delta, astar, bstar, cstar, dstar;
  unsigned int LS[n];

  for (iter = 0; iter < ITER_MAX; iter++) {
    delta = -9999;
    for (i = 0; i < n; i++) {
      a = tour[i];
      if (LS[a]) { LS[a]--; continue; }
      b = Next(a);
      for (j = head[a]; j < head[a+1]; j++) {
```

```
            c = adj[j];
            if (LS[c] | b == c) continue;
            d = Next(c);
            if (b == d | a == d) continue;
            tmp = Dis(a, b) + Dis(c, d) - Dis(a, c) - Dis(b, d);
            if (tmp > delta) {
              delta = tmp; astar = a; bstar = b;
              cstar = c; dstar = d;
            }
          }
        }
        lenght -= delta;
        Flip(astar, bstar, cstar, dstar);
        LS[astar] = TL;
      }
    }
```

実際には計算途中での最良解を記憶しておき，更新のたびに巡回路を保持する必要があるが，ここでは省略している．また，2 つのパラメータ ITER_MAX と TL の適正な値を決める問題も残っている．ITER_MAX は反復回数の上限を表すが，どの程度の計算時間で終了させたいかに依存して決めればよい．例えば，$O(n^2)$ 時間で終了させようと思えば，ITER_MAX を $O(n)(100n$ くらい) に設定しておく (課題 3)．次に禁断探索の最重要パラメータ TL の値であるが，通常は実験的解析によって決められる．例えば，アメリカ合衆国 48 都市の問題例では適正な TL の値は 20 前後とされている．

7 焼き鈍し法

焼き鈍し法は，モダンヒューリスティクスと言っても禁断探索法などよりもずっと古くから使われており，かつその人気も高い．

再び，山登りの例を使って説明しよう．今度は，懐中電灯はもちろん，寒暖計に乱数表の重装備 (？) で夜の山に出かける．山に入ったら，まず懐中電灯で周りをデタラメに照らし，照らしだされた所と今いる場所の標高差 Δ を測定する．今よりも照らしだされた所の標高が低くなければ (つまり $\Delta \geq 0$ ならば)，局所探索のときと同様，その場所に移動する．もしも $\Delta < 0$ ならば，寒暖計を取り出して現在の気温 T を測定する．そして，乱数表から $[0, 1]$ の一様乱数を得て，その値が $e^{\Delta/T}$ よりも小さかったら，

たとえ $\Delta < 0$ であっても照らしだされた場所に移動する．乱数の方が大きかったら，その場にとどまり，また辺りをデタラメに懐中電灯で照らす．山の気温は明け方が最も冷えて，夜が明けるころまでには T は徐々に下がって 0 度になる．丁度そのとき，局所探索のときのように小高い丘の上にたどり着いていることになる．

焼き鈍し法における移動を制御する関数は次のように定義される:

$$P(\mathbf{x}) = \begin{cases} \mathbf{x}' \in N(\mathbf{x}) & \text{if } c(\mathbf{x}') \geq c(\mathbf{x}) \\ \mathbf{x}' \in N(\mathbf{x}) & \text{with probablity } e^{(c(\mathbf{x}')-c(\mathbf{x}))/T} \text{ if } c(\mathbf{x}') < \mathbf{c}(x) \\ \mathbf{x} & \text{otherwise} \end{cases}$$

気温 T の高いうちは $e^{(c(\mathbf{x}')-c(\mathbf{x}))/T}$ も高く，したがって丘の頂上にいても，もっと高い丘を見つけようとするが，気温 T が下がってくるとその場に止まっている場合の方が多くなる．焼き鈍し法は，いたって人間的である．ところで,「なぜ確率 $e^{(c(\mathbf{x}')-c(\mathbf{x}))/T}$ なのか？」であるが，実は本質的な理由などないようである．

7.1 温度制御と疑似アルゴル

焼き鈍し法で最も重要となるのが気温 T の制御である．本当の山登りでは気温など制御できないが，計算機の中では可能である．これを上手く設定すると厳密な最適解を確率 1 で得ることもできるが，それには $O(n^{n^{2n-1}})$ の時間がかかるのでとても実用には向かない．ここでは**幾何的冷却**と呼ばれる実用的な温度 T の制御法を紹介する．この方法では，同一温度で均衡に達するまで移動を繰り返し，そのあとで温度を定数 (例えば 0.95) 倍することで冷却していく．

```
algorithm simulated_annealing
begin
  x := any initial feasible solution;
  T := T_START;
  while T ≥ T_END do begin
    while equilibrium-criterion = false do
      x := P(x);
    T := T× TFACTOR;
  return x
end;
```

ここで導入したパラメータは，焼き鈍し法に必要な最低限のパラメータ– 初期温度 T_START，最終温度 T_END，冷却比 T_FACTOR – である．

7.2 近傍の探索と均衡の判定

焼き鈍し法の基本は近傍からランダムに要素を選択することではあるが，改善の見込みのない無意味な探索をさけるため，巡回セールスマン問題に対してはある程度の規則性を容認することにする．つまり，置き換えを行う枝の始点 a とそれに隣接する頂点を順番に探索していきながら確率的に移動を行う方法を採用する．山登りの例に例えると，デタラメな方向に懐中電灯を向けるのではなく，決められた方向から時計回りに照らしていくことに相当する．このとき，置き換える枝の始点 a の番号は巡回路 tour[] 上をエンドレスに 0 から $n-1$ を繰り返すことになる．

また，疑似アルゴルの中で明示しなかった同一温度での均衡の判定条件だが，ここでは近傍全体を何回か網羅したら温度を下げる簡便法を採用する．そのためのパラメータとして SIZE を導入し，置き換える枝の始点 a の探索に近傍を SIZE 回巡ったら温度を変える．

7.3 プログラムと考察

区間 $[0,1)$ の乱数を与える関数 random() を用いれば，巡回セールスマン問題を解く焼き鈍し法を以下のように記述できる．

```
simulated_annealing()
{
  int i = 0; j, a, b, c, d, tmp;
  float T;

  for (T = T_START; T ≥ T_END; T *= T_FACTOR) {
    for (count = 0; count++; count < SIZE * n) {
      a = tour[i];
      b = Next(a);
      for (j = head[a]; j < head[a + 1]; j++) {
        c = adj[j];
        if (b == c) continue;
        d = Next(c);
        if (b == d | a == d) continue;
        tmp = Dis(a, b) + Dis(c, d) - Dis(a, c) - Dis(b, d);
        if (tmp ≥ 0 | random() < 1 + tmp/T) {
          length -= tmp;
          Flip(a, b, c, d);
        }
```

```
        }
      if (i == n-1) i = 0;
      else i++;
    }
  }
```

さて，導入された 4 つのパラメータ T_START, T_END, T_FACTOR, SIZE の適正な値をどうするかだが，これらは実験的解析によって定めることになる．パラメータが増えれば増えるほど，こうした実験的解析の手間も増えるので，パラメータの数はむやみに増やさないことが得策である．

課題

1. このテキストの間違いを列挙せよ．
2. 2opt 法のプログラムで，枝 ab, cd を枝 ac, bd に置き換える際，どのような条件が成り立てば解が改善されるか．また，その条件によって “if (b == c) break” が正当化される理由を述べよ．
3. 禁断探索法を n^2 に比例する演算回数で終了させようとするとき，パラメータ ITER_MAX を n に比例する値に設定する理由を述べよ．
4. 厳密算法を 1 つ実装せよ．テキストに紹介した算法でなくてもかまわない．
5. ヒューリスティクスを 2 つ実装せよ．テキストに紹介した算法でなくてもかまわない．
6. 課題 4, 5 で実装したプログラムを用いて大きさの異なる問題をいくつか解き，解の精度，CPU 時間，反復回数などの統計をとってプログラムの性能を評価・考察せよ．
7. この実験に対する感想を述べよ．

参考

2001 年 7 月 13 日 (金)，計算機を更新した際にハードディスクを壊してしまった．ディスクの中の論文とプログラムの修復や,「まつりつくば」で御輿をかついだりしたので，このテキストを準備する時間がなかなかとれず，結局，書き上げるのに 9 月 5 日 (水) までかかった．ディスクの修復の方はほとんど絶望的で，改訂シンプレックス法のコード (Revex3，ホー

ムページからダウンロードできる！ — が，仕様書はない) を修復できたのみ．幸い，このテキストは

> 久保幹雄,「巡回セールスマン問題への招待 I, II, III」，オペレーションズ・リサーチ**39** (1994), No.1: 25–31, No.2: 91–96, No.3: 156–162

という素晴らしい解説書のお陰で，それをほぼ丸写しすることでなんとか書き上げられた．もう商船大の久保君へは足を向けて眠ることはできない．久保君は，このほかにも社会工学系の山本先生と同じタイトルで

> 山本芳嗣，久保幹雄，巡回セールスマン問題への招待，朝倉書店 (1997)

という妙な本を書いている．関西の人には読みやすいかもしれない．

当初，ハードディスクが壊れるまでは，久保君の書き物に頼るのは悔しいので，名著

> Lawer, E.L., J.K., Lenstra, A.H.G.Rinnooy Kan and D.B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Willey & Sons (1985)

でも参考にしてテキストを準備しようかと考えていた．しかし，16 年前の本は，過激に進歩する IT 社会の現代ではカビの生えた古典にすぎない．最先端を知るには，インターネットを駆使して先に挙げた

> www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

や，久保君のホームページ

> www.ipc.tosho-u.ac.jp/ kubo/

を閲覧した方が手っ取り早い．また，東大の松井君のページ

> www.misojiro.t.u-tokyo.ac.jp/ tomomi/opt-code.html

には最適化のソフトウェアとテスト問題へのリンクが網羅してあり，彼はこのページで日本オペレーションズ・リサーチ学会の研究普及賞まで受賞している．ところが，最近は更新をサボっているらしくリンク切れが多い．がっかりさせられることもままあるので，早く更新を再開してほしい．

適当な検索エンジンで「巡回セールスマン問題」を検索すると膨大な量のホームページが検索されるが，中には実にくだらないものもある．取捨選択を誤ると時間ばかりとられることになるので要注意である．