平成１８年度　秋期　ソフトウェア開発技術者　午後Ⅱ問題

問１　配送計画問題のアルゴリズムに関する次の記述を読んで，設問 1～3 に答えよ。

〔配送計画問題〕

トラックで，倉庫から店舗に品物を配送する。このとき，所要時間が最小となる配送経路を求めたい。前提条件は次のとおりである。

・一つの倉庫から複数の店舗に品物を配送する。トラックは 1 台である。
・トラックは，倉庫から出発してすべての店舗に品物を届けた後，倉庫に戻る。
・任意の店舗の間，及び倉庫と任意の店舗の間には，それらを結ぶ道路が存在する。
・店舗 A から店舗 B へ行くときの所要時間と，店舗 B から店舗 A へ行くときの所要時間は同じである。倉庫と任意の店舗の間についても同様である。また，所要時間は，時間帯，積載重量などにかかわらず，常に一定である。

〔配送計画問題のモデル化〕

配送計画問題を，グラフの概念を用いて次のようにモデル化する。店舗が三つの場合の地図とそのモデルを，それぞれ図 1，図 2 に示す。

・倉庫と店舗をノード（図 2 の円）で表し，その間の道路を枝（図 2 の線）で表す。
・ノードの数を n とした場合，ノードには，V_0，V_1，V_2，…，V_{n-1} というラベルが付けられている。ノード V_0 は倉庫に，ノード V_1 ～ V_{n-1} は店舗に相当する。
・ノード V_i とノード V_j の間の枝を E_{ij} という。
・枝には，正の数値の重みが付されている。重みは，2 地点間の所要時間に相当する。

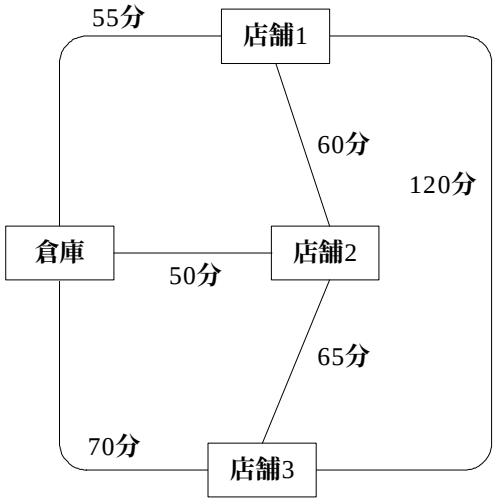


図１　店舗が三つの場合の地図

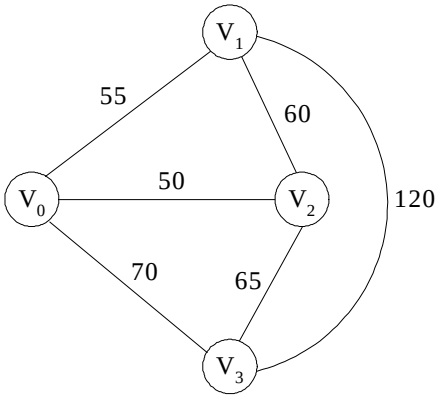


図２　グラフによる図１のモデル

配送計画問題は，グラフの上では“ノード V_0 から出発し，ノード V_1，V_2，…，V_{n-1} のすべてを経由してノード V_0 に戻る経路（以下，巡回路という）のうち，経路上の枝の重みの和が最小のものを求める問題”と考えることができる。

なお，以下では，巡回路の枝の重みの和を，“巡回路の重み”と呼ぶことにする。

配送計画問題に対して，巡回セールスマン問題を解くアルゴリズムを用いることにする。

〔巡回セールスマン問題〕

巡回セールスマン問題とは，与えられたグラフに対して，次の条件①～③を満たし，かつ，枝の重みの和が最小であるような経路を求める問題のことをいう。

① すべてのノードを通る。
② 各ノードを 1 回だけ通る。
③ 出発点に戻る。

巡回セールスマン問題は，最適解を見つけるためには，総当たり的なアルゴリズムによるしかないと考えられている，極めて扱いにくい問題である。そこで，実用的なアプローチとして，近似最適な解を見つけるための様々なアルゴリズムが考案されている。その代表的なものとして，Nearest Neighbor 法がある。

〔Nearest Neighbor 法〕

Step1：c = 0 とする。
Step2：ノード V_c とまだ経由していないノードとを結ぶ枝のうち，重みが最も小さい枝を選ぶ。最小の重みの枝が複数存在するときには，最初に見つかった枝を選ぶ。選んだ枝のもう一方の端のノードを V_k とする。c = k として，このステップを繰り返す。
Step3：すべてのノードを経由していれば，V_0 への枝を選ぶ。

Nearest Neighbor 法のプログラム（プログラム 1）を，図 3 に示す。配列 D[i, j] には，枝 E_{ij} の重みが格納されている。前提条件から，D[i, j] = D[j, i] である。また，ノード V_i とノード V_i の間には枝が存在しないことを表すために，D[i, i] には十分大きな数値 X が格納されている。

なお，図 3 中の mod(k, n) は，k を n で除したときの剰余を返す関数である。

プログラム 1 は，巡回路を一つ見つけた時点で終了するので，問題によっては，かなり質の悪い解しか得られないという欠点がある。それに対して，巡回路が得られた後も，その一部分を変形して，より良い巡回路を求める操作を繰り返すアルゴリズムを考える。これによって，処理時間はかかるが Nearest Neighbor 法よりもよい結果が得られることが期待できる。

巡回路の変形方法としては，2-opt と呼ばれる手法が提案されている。

プログラム 1：Nearest Neighbor 法

```
procedure Nearest_Neighbor    //n はノードの数
   for i=0 to n-1             //初期化
      {
      visited[i1 ← false ;
      }
   current ← 0 ;
   for i=0 to n-1
      {
      tour[i] ← current ;
      visited[current] ← true ;
      min ← 十分大きな値 X ;
      for j=0 to n-1
         {
         if (visited[j] = [  ア  ] and D[current, j] < min)
            then
               {
               min ← [  イ  ] ;
               min_j ← [  ウ  ] ;
               }
         }
      [  エ  ] ← min_j ;
      }
   length ← 0 ;
   for i=0 to n-1             //巡回路の重みの算出
      {
      length ← [  オ  ] +D[tour[i], tour[mod(i+1, n)]] ;
      }
```

図３　Nearest Neighbor 法のプログラム

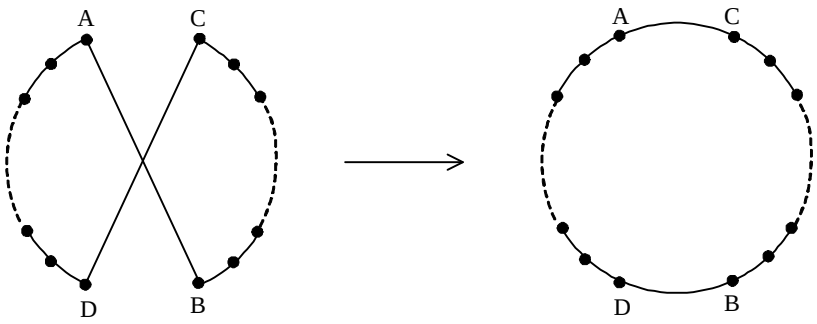


図４　巡回路の変形操作（2-opt）

2-opt とは，図 4 の例に示すように，巡回路中のある二つの枝 AB と CD を除去し，枝 AC と BD を付加して新たな巡回路を得る変形操作のことをいう。なお，枝 AB と CD を除去した後に枝を付加して新たな巡回路を構成する方法は，枝 AC と BD を付加する方法しかないことに注意する必要がある。また，2-opt による変形操作後に得られる巡回路では，その一部分の巡回方向が元の巡回路と逆になることにも注意が必要である。

2-opt の考え方を用いた，次のようなアルゴリズムを考える。

〔2-opt による改良アルゴリズム〕

Step1：Nearest Neighbor 法を用いて，巡回路を一つ求める。この巡回路を T とする。

Step2：T に対して，あるノード（“開始ノード”という）から始めて，2 本の枝の組合せに対して順に 2-opt による巡回路の変形操作を試みる。T よりも重みの小さい巡回路が見つかったときには，その巡回路を T として戻る。

Step2 のアルゴリズムのプログラム（プログラム 2）を，図 5 に示す。プログラム 2 を最初に実行するとき，配列 tour には，プログラム 1 で求めた巡回路が格納されている。また，length には，その巡回路の重みが格納されている。start は，開始ノードである。また，floor(k, n) は，k を n で除したときの商の整数部分を返す関数である。

Step2 で改良された巡回路が見つかった後は，それを T とし，開始ノードを変えて Step2 を実行することを規定回数繰り返す。そして，これらの操作を行った後の T を最終結果として出力する。

プログラム 2：Step2 のアルゴリズム

```
procedure step_two (start)
   for i=start to start+n-1
      {
      for k=i+2 to i+n-2
         {
         diff ← (D[tour[mod(i, n)], tour[mod(i+1, n)]]
                  + D[tour[mod(k, n)], tour[mod(k+1, n)]])
                - (D[tour[mod( カ , n)], tour[mod( キ , n)]]
                  +D[tour[mod( ク , n)], tour[mod( ケ , n)]]) ;
         if (diff > 0)          //より重みの小さい巡回路が見いだされた。
            then
               {
               length ← length - コ ;
               for j=0 to floor(k-i, 2)-1      //巡回路の一部分を反転する。
                  {
                     temp ← tour[mod(i+1+j, n)] ;
                     tour[mod(i+1+j, n)] ← tour[ サ ] ;
                     tour[ サ ] ← temp ;
                  }
               return ;
               }
         }
      }
```

図５　Step2 のアルゴリズムのプログラム

設問１　次の記述中の［　　］に入れる適切な字句を解答群の中から選び，記号で答えよ。

本問で対象としているような巡回セールスマン問題においては，ノードの数が n であるとき，ノードの巡回順序が逆転している巡回路を区別しないこととすれば巡回路は［　　］通り存在する。このことから，n が大きくなった場合には，総当たり的なやり方で最適解を求めようとすると天文学的な時間を要することが分かる。

解答群

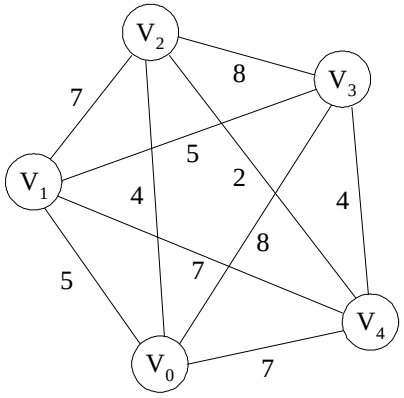
ア　(n－1)!／2　　イ　n!／2　　ウ　(n＋1)!／2

エ　(n－1)!　　オ　n!　　カ　(n+1)!

設問２　Nearest Neighbor 法について，(1)～(3)に答えよ。

(1) プログラム 1 中の［　ア　］～［　オ　］に入れる適切な字句を答えよ。

(2) 次のグラフに対して，プログラム 1 を実行した場合の巡回路を太線で示せ。



(3) プログラム 1 では，問題によってはかなり質の悪い解しか得られない。このことについて述べた次の記述中の［　シ　］，［　ス　］に入れる適切な数値（正の整数）を答えよ。

図 2 のグラフにおける最適解の巡回路の重みは，［　シ　］である。しかし，プログラム 1 では，巡回路の重みが最適解の［　ス　］%増の解しか得られない。

設問３　2-opt による改良アルゴリズムについて，(1)，(2)に答えよ。

(1) プログラム 2 中の［　カ　］～［　サ　］に入れる適切な字句を答えよ。

(2) 図 2 のグラフに対して，配列 tour に，0，2，1，3 を格納し，start の値を 1 として，プログラム 2 を 1 回実行して求められる巡回路を太線で示せ。ただし，配列 tour の添字は 0 から始まる。