Traveling Salesman Problem

Consider the problem of the traveling salesman who has a large number of cities to visit. Since he is paid on commission it is to his advantage to minimize the travel time so he is interested in finding the shortest route that connects all of the cities. For the purposes of this problem the starting city will also be the ending city and the only city that can be visited twice.

For example, if the number of cities is 100 then the salesman has 99 choices for the first place to visit, 98 choices for the second place, 97 for the third, and so on. This means that there are 99! possible routes. In general for N cities then, there are (N-1)! routes. To solve this problem exhaustively on a computer we would have to check all routes to verify that the one we have is the shortest. If N is a small number (say 4) then this is a relatively simple problem. There are just (4-1)! = 6 routes. We can list them as

1\rightarrow2\rightarrow3\rightarrow4\rightarrow1
1\rightarrow2\rightarrow4\rightarrow3\rightarrow1
1\rightarrow3\rightarrow2\rightarrow4\rightarrow1
1\rightarrow3\rightarrow4\rightarrow2\rightarrow1
1\rightarrow4\rightarrow2\rightarrow3\rightarrow1
1\rightarrow4\rightarrow3\rightarrow2\rightarrow1

But factorials grow very quickly – even faster than exponentials. The number 50! is already 3 \times 10^{64} and 100! = 9.3 \times 10^{157} is too big to calculate for many hand held calculators (try it in Derive where you get an exact answer).

So the traveling salesman problem cannot be solved in your brief lifetime by doing an exhaustive search of 99! routes for 100 cities.

One way to get a reasonable answer to this problem is to use an evolutionary algorithm. In this approach you select a random route as the first choice. You then make random changes to the route and discard those changes which make it longer (survival of the fittest). After doing this for many times you will find that your random route becomes pretty good and for many problems it is a suitable (but not optimal) solution.

For this assignment you will run 10,000 random corrections to an initially random route between 100 cities. Your program should do a graphical map of the 100 city locations using Visual Studio 6 and the Carnegie Mellon Graphics package and draw in the initial random route in red. After making 10,000 random corrections to the route and keeping only those which shorten it, your program should draw another map of the shortened route in green on the same screen. You should also display the total distance length for each route.

Your program must use the City class defined below. In this definition, each city is assigned an index. For example if there are 100 cities then the city indices would range from 0 to 99. The city index is used in place of a city name. The City class provides for storage of the next index and the old index. The next index is the next city that the salesman goes to after this one and the old index is the city that he came from. The Boolean variable visited is TRUE if the city was visited so that you can keep track of which cities were visited. The location variables, x and y, are the screen coordinates on the graphical screen.

class City
For this program you can declare an array of 100 cities. Initialize the index of each city to its array index. Let index 0 be the starting (and ending) city. Your program should determine a random route through all of the cities. To do this you will choose a random integer between 0 and 100. This randomly chosen number will be the index of the city that is being visited next. Verify that the index of the chosen city has not been visited before and link that city to the present city. You can do this link by setting the indexNext variable of the present city to the randomly chosen index. You should set the indexOld of the chosen city to the index of the city that you are at now. Continue this process until all 100 cities have been visited. After a route has been chosen you should have a function which will return the total distance between the cities on the route. This FindDistance function can follow the index links, determine successive locations, and use the distance formula to find the distance. (The distance formula is 

\[ d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \]

where one city is at \((x_1, y_1)\) and the other city is at \((x_2, y_2)\).)

You should use the following code segment to initialize the location of the cities.

```cpp
rseed(2);
for (i=0; i<numCity; i++)
{ x1 = 1 + rand() % right;
  y1 = 1 + rand() % bottom;
  c[i].SetLocation(x1, y1);
  c[i].SetIndex(i);
}
srand(yourInitialSeed);
```

In this code fragment, numCity is a constant and is equal to the number of cities (100). The numbers right and bottom are the right most and bottom most coordinate of your screen respectively. The function rseed, initializes the random number generator so that every one in the class will begin with cities at the same coordinates. The variable yourInitialSeed can be an arbitrary positive integer that you use to reinitialize the random number generator so that your solution is not exactly the same as everyone else’s.

A test program has been created on the web site at

http://csserver.evansville.edu/~blandfor

This program uses CarnegieMellonGraphics.h, draws 25 random points on the screen, connects them with a blue route, and colors the first city in red. The name of the program is TravelTest.cpp.