CS29003 ALGORITHMS LABORATORY
Re-warming Assignment (Not for Submission)
Date: 23–September–2015

The *Traveling Salesperson Problem* (**TSP**) is theoretically, practically, historically, and emotionally a very important problem in Computer Science. We have n cities numbered 0, 1, 2, …, $n - 1$. The distance between City i and City j is denoted by $c(i, j)$. It can be taken as the cost of traveling between City i and City j. Assume that $c(i, j) = c(j, i)$ for all i, j with $i \neq j$. Also take $c(i, i) = 0$ for all i. A salesperson starts from City 0, visits all the cities once and only once, and finally comes back to City 0. A tour of the salesperson is defined by a cyclic permutation of 0, 1, 2, …, $n - 1$. The total cost of such a cycle is the sum of distances of all edges on the cycle. The salesperson wants to minimize the total cost over all possible cycles. There are $(n - 1)!$ cycles for n cities (the count may be divided by two to account for a cycle being traversed in one of the two directions). This quantity grows very rapidly with n.

Part 1 [*Exhaustive Search*]

Write a recursive function *mincostexh*() to generate all (cyclic) permutations of 0, 1, 2, …, $n - 1$. For each such permutation, compute the cost, and take the minimum over all these costs.

In order to reduce the prohibitive running time of exhaustive search, we devise clever strategies so that some recursive calls can be eliminated. Suppose that a partial tour of the salesperson is generated. This tour is based on a choice of the first k cities, call them $C_0, C_1, \ldots, C_{k-1}$ (we always take $C_0 = 0$). This means that $n - k$ cities remain to be added to the tour. Exhaustive search places all of the remaining cities as C_k, and makes a recursive call for each of these choices.

Let w be the best (minimum) cost of a tour found so far. The cost incurred by the partially generated tour $C_0, C_1, \ldots, C_k$ is $u = c(C_0, C_1) + c(C_1, C_2) + \ldots + c(C_{k-1}, C_k)$. We compute a lower bound v on the cost incurred by adding the remaining $n - k$ edges. If $u + v \geq w$, there is no point making the recursive call on $C_0, C_1, \ldots, C_k$, whereas if $u + v < w$, there remains a possibility that exploration with the initial part $C_0, C_1, \ldots, C_k$ may lead to a better (cheaper) solution, so the recursive call is made. This strategy is called *pruning*. The larger the estimate v is, the more effective pruning is to curtail the cost of exhaustive search.

Part 2 [Global Pruning Strategy]

Let m be the minimum of all the costs $c(i, j)$ with $i \neq j$. We can definitely take $v = (n - k)m$. Write a function *mincostgp*() to implement this pruning strategy.

Part 3 [Local Pruning Strategy I]

Consider the partial tour $C_0, C_1, \ldots, C_k$. Name the remaining cities to be visited as $D_{k+1}, D_{k+2}, \ldots, D_{n-1}$. Also, assume that $1 \leq k \leq n - 2$ (C_0 is always 0, and for $k = n - 1$, we terminate recursion). From C_k, the salesperson visits some city D_i for $i = k + 1, k + 2, \ldots, n - 1$ (but not C_0). Let v_k be the minimum of the distances $c(C_k, D_i)$. Since the salesperson has to get out of City C_k, (s)he has to encounter at least this much cost on an outgoing edge from C_k. For $i = k + 1, k + 2, \ldots, n - 1$, the salesperson leaves D_i and either reaches an unvisited D_j or completes the tour by returning to C_0 ((s)he does not go back to C_k), so let v_i denote the distance from D_i to the closest city in $\{D_{k+1}, D_{k+2}, \ldots, D_{n-1}, C_0\} - \{D_i\}$. Leaving D_i uses an edge at least as costly as v_i. Write a function *mincostlp*1() that takes $v = v_k + v_{k+1} + \ldots + v_{n-1}$. These minimum costs $v_k, v_{k+1}, \ldots, v_{n-1}$ are not necessarily compatible with one another, but this v is definitely a lower bound on the remaining cost (and is usually much better than $(n - k)m$ of Part 2).

Part 4 [Local Pruning Strategy II]

An even better local pruning strategy is based upon looking at *two* minimum-cost edges associated with an unvisited city. The reason is that if you enter a city using an edge, you must use a *separate* edge for leaving the city. For any $i = 0, 1, \ldots, n - 1$, let V_i denote the sum of the costs of the incoming and the outgoing edges at the i-th city. But then, $V_0 + V_1 + \ldots + V_{n-1}$ is twice the cost of the entire tour. Improve the estimate v of the remaining cost based upon this observation, and write a function *mincostlp*2() that uses this pruning strategy.

Write a *main*() function that first populates the symmetric distance array *c*[][] by user inputs/random numbers. For simplicity, assume that each distance is a positive integer. Call the four functions of Parts 1 – 4 to print the optimal cost. Do not call the exhaustive-search function of Part 1 if n is large (like > 10). For each function, you should also compute the total number of calls made and the total number of permutations generated (leaves in the computation tree).

Do not use any global/static variable/array.

Sample Output

We supply two sample runs. The first one is verbose. It prints a path as soon as its cost is smaller than the current minimum-cost path. The second output omits these details.

```
n = 6

+++ The cost matrix:
     0  43  14  46  99  28
    43   0  51  42  59  15
    14  51   0  74   7  51
    46  42  74   0   1  31
    99  59   7   1   0  96
    28  15  51  31  96   0

+++ Exhaustive search:
    New tour found:  0  1  2  3  4  5 Cost:  293
    New tour found:  0  1  2  4  3  5 Cost:  161
    New tour found:  0  1  5  3  4  2 Cost:  111
    New tour found:  0  2  4  3  1  5 Cost:  107
    Minimum cost computed     = 107
    Number of leaves explored = 60
    Number of calls made      = 120

+++ Search with global pruning strategy:
    New tour found:  0  1  2  3  4  5 Cost:  293
    New tour found:  0  1  2  4  3  5 Cost:  161
    New tour found:  0  1  5  3  4  2 Cost:  111
    New tour found:  0  2  4  3  1  5 Cost:  107
    Minimum cost computed     = 107
    Number of leaves explored = 9
    Number of calls made      = 44

+++ Search with local pruning strategy 1:
    New tour found:  0  1  2  3  4  5 Cost:  293
    New tour found:  0  1  2  4  3  5 Cost:  161
    New tour found:  0  1  5  3  4  2 Cost:  111
    New tour found:  0  2  4  3  1  5 Cost:  107
    Minimum cost computed     = 107
    Number of leaves explored = 4
    Number of calls made      = 19

+++ Search with local pruning strategy 2:
    New tour found:  0  1  2  3  4  5 Cost:  293
    New tour found:  0  1  2  4  3  5 Cost:  161
    New tour found:  0  1  5  3  4  2 Cost:  111
    New tour found:  0  2  4  3  1  5 Cost:  107
    Minimum cost computed     = 107
    Number of leaves explored = 4
    Number of calls made      = 16
```

```
n = 20

+++ The cost matrix:
     0  83  53  77  17  69  82  64   6  62  31  91  46  81  92  56  47  42  96  92
    83   0  49  70  62  56  90  57  87  89  31  42  35   8  25  87  82  41  54  65
    53  49   0   5  59  27  34  48  72  13  41  27  57  80  21  50  30  88  10  85
    77  70   5   0  79  65  70  66  93  12   1  98  34  86  81  72  40  44  75  99
    17  62  59  79   0  70   7  47  40  19  85  64  74  66  83  22  95  71  31  78
    69  56  27  65  70   0  48  93  49  15  87  58  15  83  92   1  62  64  41   6
    82  90  34  70   7  48   0  37  38  73  43  84  12  62  70  75  36  36  56  57
    64  57  48  66  47  93  37   0  29  28  85   7  75  79  55  87  64  14   3  47
     6  87  72  93  40  49  38  29   0   4   1  10  65  39  13   3  76  86  45  59
    62  89  13  12  19  15  73  28   4   0  95   7  29  70  40  62  27  94  90  52
    31  31  41   1  85  87  43  85   1  95   0  78  96  25  55  50  13  17  61  13
    91  42  27  98  64  58  84   7  10   7  78   0  63  64  11  70  29  50  81  31
    46  35  57  34  74  15  12  75  65  29  96  63   0  24  67  76  80  62  80   9
    81   8  80  86  66  83  62  79  39  70  25  64  24   0  30  19  70  54  11  60
    92  25  21  81  83  92  70  55  13  40  55  11  67  30   0   4  86  55  29  41
    56  87  50  72  22   1  75  87   3  62  50  70  76  19   4   0   5  39  57  65
    47  82  30  40  95  62  36  64  76  27  13  29  80  70  86   5   0  51  19  29
    42  41  88  44  71  64  36  14  86  94  17  50  62  54  55  39  51   0  61  86
    96  54  10  75  31  41  56   3  45  90  61  81  80  11  29  57  19  61   0  55
    92  65  85  99  78   6  57  47  59  52  13  31   9  60  41  65  29  86  55   0

+++ Exhaustive search: Not done

+++ Search with global pruning strategy:
    Minimum cost computed     = 198
    Number of leaves explored = 214519
    Number of calls made      = 244971134

+++ Search with local pruning strategy 1:
    Minimum cost computed     = 198
    Number of leaves explored = 111
    Number of calls made      = 1176192

+++ Search with local pruning strategy 2:
    Minimum cost computed     = 198
    Number of leaves explored = 111
    Number of calls made      = 38226
```