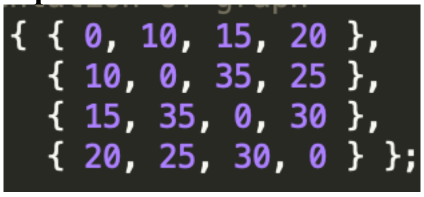
Lab 3
Isa Dzhumabaev

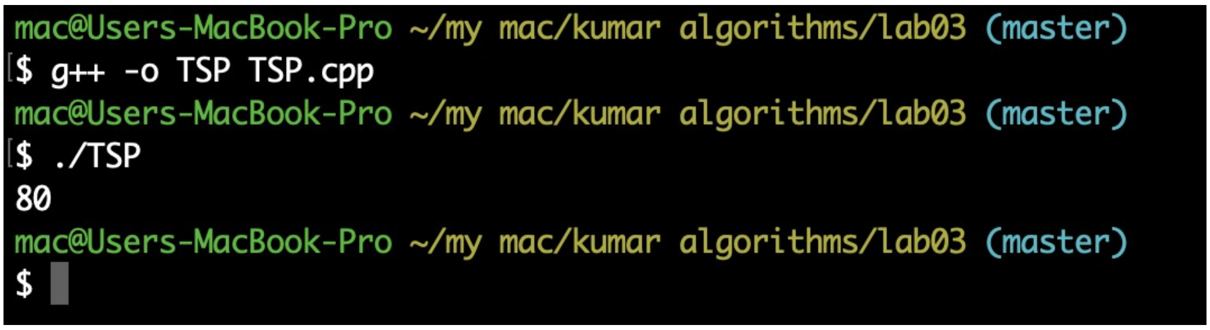
1. Compile/Run the code given in the previous slide

Input:



```
{ { 0, 10, 15, 20 },
  { 10, 0, 35, 25 },
  { 15, 35, 0, 30 },
  { 20, 25, 30, 0 } };
```

Result:

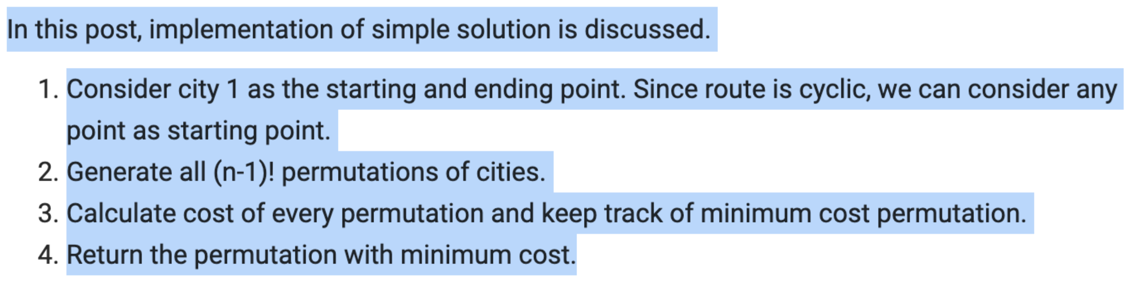


```
mac@Users-MacBook-Pro ~/my mac/kumar algorithms/lab03 (master)
[$ g++ -o TSP TSP.cpp
mac@Users-MacBook-Pro ~/my mac/kumar algorithms/lab03 (master)
[$ ./TSP
80
mac@Users-MacBook-Pro ~/my mac/kumar algorithms/lab03 (master)
$ 
```

2. Explain how does it use the greedy algorithm

I don’t think that this exact implementation uses greedy algorithm. Greedy algorithm is based on making localy optimal decision assuming that this will lead to optimal solution.
I think that this exact implementation actually uses brute-force as it tries to consider all possible permutations, count their cost and take minimal. It is said so it article.

Algorithm idea used in the article:



In this post, implementation of simple solution is discussed.

1. Consider city 1 as the starting and ending point. Since route is cyclic, we can consider any point as starting point.
2. Generate all (n-1)! permutations of cities.
3. Calculate cost of every permutation and keep track of minimum cost permutation.
4. Return the permutation with minimum cost.

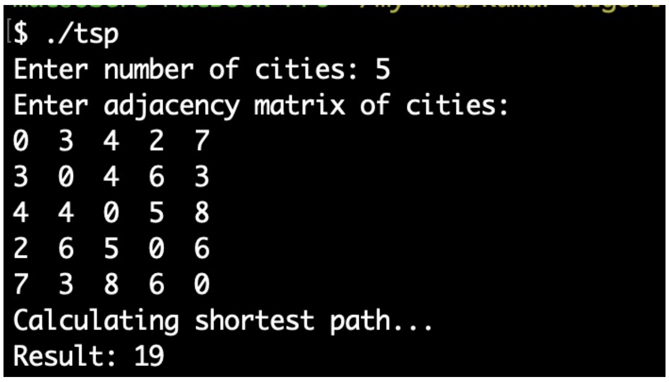
Greedy TSP solution takes minimal cost on every particular vertex and this does not guaranty the best solution.

3. Review the datasets given in the second link

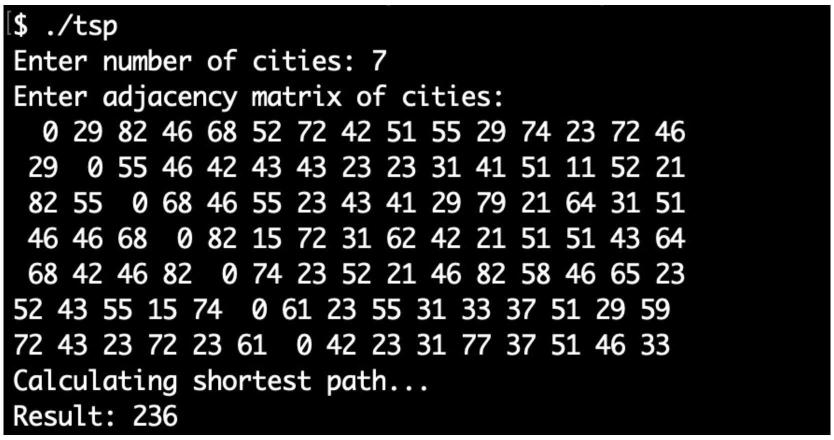
I wanted to test this algorithm with 26, 17, 15 and 5 cities form this link: https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html.

But this algorithm was too slow even on 15 cities so I decided not to use 15, 17 and 26. Instead I just used 5 cities and two reduced versions of input with 15 cities.

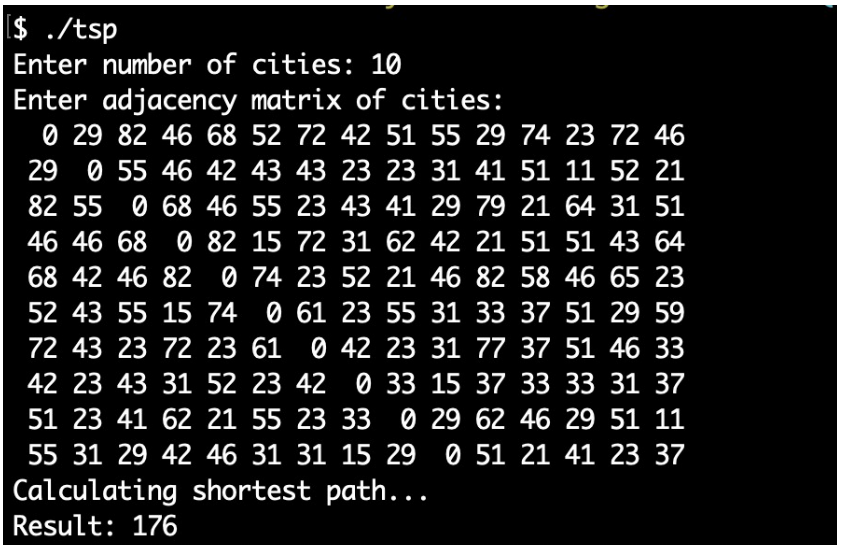
Results of tests:



```
[$ ./tsp
Enter number of cities: 5
Enter adjacency matrix of cities:
0  3  4  2  7
3  0  4  6  3
4  4  0  5  8
2  6  5  0  6
7  3  8  6  0
Calculating shortest path...
Result: 19
```



```
[$ ./tsp
Enter number of cities: 7
Enter adjacency matrix of cities:
  0 29 82 46 68 52 72 42 51 55 29 74 23 72 46
 29  0 55 46 42 43 43 23 23 31 41 51 11 52 21
 82 55  0 68 46 55 23 43 41 29 79 21 64 31 51
 46 46 68  0 82 15 72 31 62 42 21 51 51 43 64
 68 42 46 82  0 74 23 52 21 46 82 58 46 65 23
52 43 55 15 74  0 61 23 55 31 33 37 51 29 59
72 43 23 72 23 61  0 42 23 31 77 37 51 46 33
Calculating shortest path...
Result: 236
```



```
[$ ./tsp
Enter number of cities: 10
Enter adjacency matrix of cities:
  0 29 82 46 68 52 72 42 51 55 29 74 23 72 46
 29  0 55 46 42 43 43 23 23 31 41 51 11 52 21
 82 55  0 68 46 55 23 43 41 29 79 21 64 31 51
 46 46 68  0 82 15 72 31 62 42 21 51 51 43 64
 68 42 46 82  0 74 23 52 21 46 82 58 46 65 23
 52 43 55 15 74  0 61 23 55 31 33 37 51 29 59
 72 43 23 72 23 61  0 42 23 31 77 37 51 46 33
 42 23 43 31 52 23 42  0 33 15 37 33 33 31 37
 51 23 41 62 21 55 23 33  0 29 62 46 29 51 11
 55 31 29 42 46 31 31 15 29  0 51 21 41 23 37
Calculating shortest path...
Result: 176
```

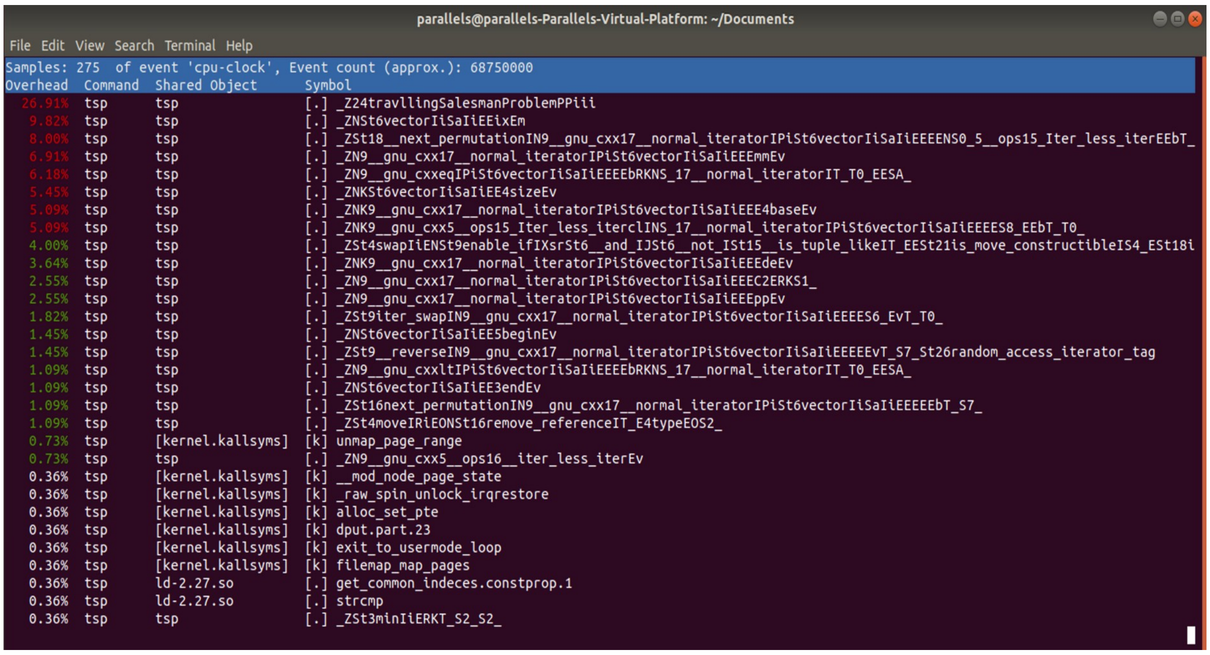
4. Modify/Use both programs to use larger inputs

Source code of modified version to take larger inputs:

```
#include <bits/stdc++.h>
#include <time.h>
using namespace std;

int travllingSalesmanProblem(int** graph, int s, int n)
{
    vector<int> vertex;
    for (int i = 0; i < n; i++)
        if (i != s)
            vertex.push_back(i);

    int min_path = INT_MAX;
    do {

        int current_pathweight = 0;

        int k = s;
        for (int i = 0; i < vertex.size(); i++) {
            current_pathweight += graph[k][vertex[i]];
            k = vertex[i];
        }
        current_pathweight += graph[k][s];

        min_path = min(min_path, current_pathweight);

    } while (next_permutation(vertex.begin(), vertex.end()));

    return min_path;
}

int main()
{
    int n;
    printf("Enter number of cities: ");
    scanf("%d", &n);

    printf("Enter adjacency matrix of cities: \n");
    int** graph;
    graph = new int*[n];
    for (int i = 0; i < n; ++i) {
        graph[i] = new int[n];
        for (int j = 0; j < n; ++j) {
            scanf("%d", &graph[i][j]);
        }
    }
    printf("\nCalculating shortest path...\n");
    int s = 0;
    int result = travllingSalesmanProblem(graph, s, n);
    printf("Result: %d\n", result);
    return 0;
}
```

5. Run operf/opreport on both with larger inputs

Perf was used on VM using Parallels.
We can observe that **travllingSalesmanProblem()** function takes most time as expected.


parallels@parallels-Parallels-Virtual-Platform: ~/Documents
File Edit View Search Terminal Help

```
Samples: 275  of event 'cpu-clock', Event count (approx.): 68750000
Overhead  Command  Shared Object      Symbol
  26.91%  tsp      tsp                [.] _Z24travllingSalesmanProblemPPiii
   9.82%  tsp      tsp                [.] _ZNSt6vectorIiSaIiEEixEm
   8.00%  tsp      tsp                [.] _ZSt18__next_permutationIN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEENS0_5__ops15_Iter_less_iterEEbT_
   6.91%  tsp      tsp                [.] _ZN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEmmEv
   6.18%  tsp      tsp                [.] _ZN9__gnu_cxxeqIPiSt6vectorIiSaIiEEEEbRKNS_17__normal_iteratorIT_T0_EESA_
   5.45%  tsp      tsp                [.] _ZNKSt6vectorIiSaIiEE4sizeEv
   5.09%  tsp      tsp                [.] _ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEE4baseEv
   5.09%  tsp      tsp                [.] _ZNK9__gnu_cxx5__ops15_Iter_less_iterclINS_17__normal_iteratorIPiSt6vectorIiSaIiEEEES8_EEbT_T0_
   4.00%  tsp      tsp                [.] _ZSt4swapIiENSt9enable_ifIXsrSt6__and_IJSt6__not_ISt15__is_tuple_likeIT_EESt21is_move_constructibleIS4_ESt18i
   3.64%  tsp      tsp                [.] _ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv
   2.55%  tsp      tsp                [.] _ZN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEC2ERKS1_
   2.55%  tsp      tsp                [.] _ZN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEppEv
   1.82%  tsp      tsp                [.] _ZSt9iter_swapIN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEES6_EvT_T0_
   1.45%  tsp      tsp                [.] _ZNSt6vectorIiSaIiEE5beginEv
   1.45%  tsp      tsp                [.] _ZSt9__reverseIN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEEEvT_S7_St26random_access_iterator_tag
   1.09%  tsp      tsp                [.] _ZN9__gnu_cxxltIPiSt6vectorIiSaIiEEEEbRKNS_17__normal_iteratorIT_T0_EESA_
   1.09%  tsp      tsp                [.] _ZNSt6vectorIiSaIiEE3endEv
   1.09%  tsp      tsp                [.] _ZSt16next_permutationIN9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEEEbT_S7_
   1.09%  tsp      tsp                [.] _ZSt4moveIRiEONSt16remove_referenceIT_E4typeEOS2_
   0.73%  tsp      [kernel.kallsyms]  [k] unmap_page_range
   0.73%  tsp      tsp                [.] _ZN9__gnu_cxx5__ops16__iter_less_iterEv
   0.36%  tsp      [kernel.kallsyms]  [k] __mod_node_page_state
   0.36%  tsp      [kernel.kallsyms]  [k] _raw_spin_unlock_irqrestore
   0.36%  tsp      [kernel.kallsyms]  [k] alloc_set_pte
   0.36%  tsp      [kernel.kallsyms]  [k] dput.part.23
   0.36%  tsp      [kernel.kallsyms]  [k] exit_to_usermode_loop
   0.36%  tsp      [kernel.kallsyms]  [k] filemap_map_pages
   0.36%  tsp      ld-2.27.so         [.] get_common_indeces.constprop.1
   0.36%  tsp      ld-2.27.so         [.] strcmp
   0.36%  tsp      tsp                [.] _ZSt3minIiERKT_S2_S2_
```