Delhi Technological University

Department of Applied Mathematics

M.Sc Mathematics

Travelling Salesman Problem

Submitted By :
Vidipt Vashist
2K20/MSCMAT/33
Bharat Varshney
2K20/MSCMAT/07

Submitted To :
Prof. Payal

Travelling Salesman Problem

Vidipt Vashist (2K20/MSCMAT/33)
Bharat Varshney (2K20/MSCMAT/07)

January 31, 2021

Contents

1 THE PROBLEM STATED

A traveling salesman wishes to go to a certain number of destinations in order to sell objects. He wants to travel to each destination exactly once and return home taking the shortest total route.
Each voyage can be represented as a graph G = (V,E) where each destination, including his home, is a vertex, and if there is a direct route that connects two distinct destinations then there is an edge between those two vertices. The traveling salesman problem is solved if there exists a shortest route that visits each destination once and permits the salesman to return home.

Traveling Salesman Problem (TSP)

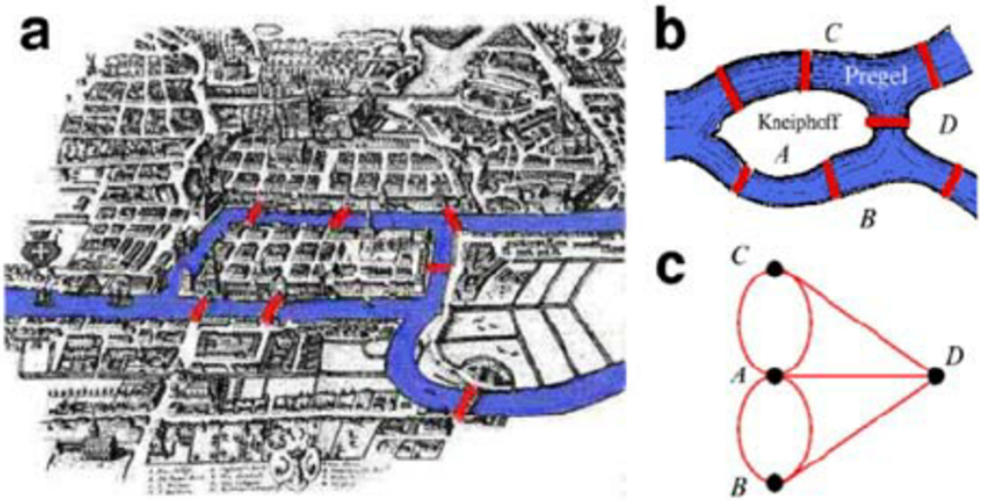
- Given: a collection of cities and the cost of travel between each pair
- Goal: find the cheapest way of visiting all cities and returning to the starting point

2


3

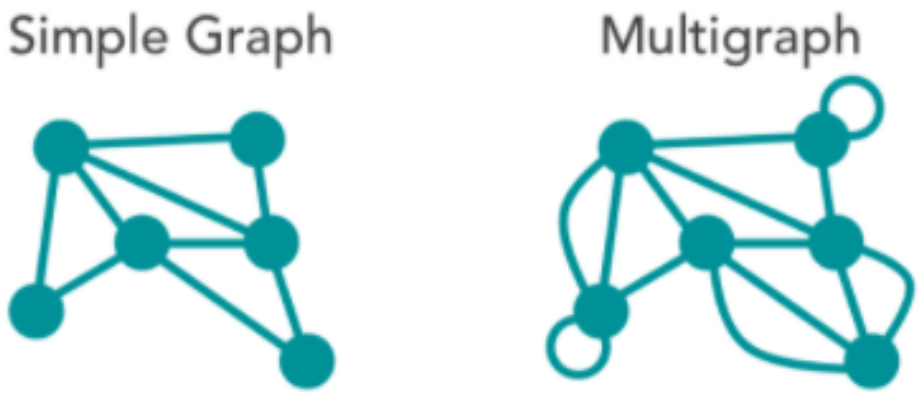
2 SOME BASIC GRAPH THEORY

Before presenting algorithms and conditions for when a locally optimal solution exists, some basic graph theory definitions are needed. For an extensive overview of Graph Theory.



Definition

SIMPLE GRAPH
A simple graph, G = (V,E), is a finite nonempty set V of objects called vertices (singular vertex) to- gether with a possibly empty set E of 2-element subsets of V called edges.



Definition

ADJACENT
Two vertices are adjacent if they are connected by an edge.

Definition

INCIDENT
Vertex u and the edge uv are said to be incident with each other. Similarly, v and uv are incident.

Definition

DEGREE OF A VERTEX
The degree of a vertex v (denoted: deg(v)) is the number of vertices in G that are adjacent to v.

Definition

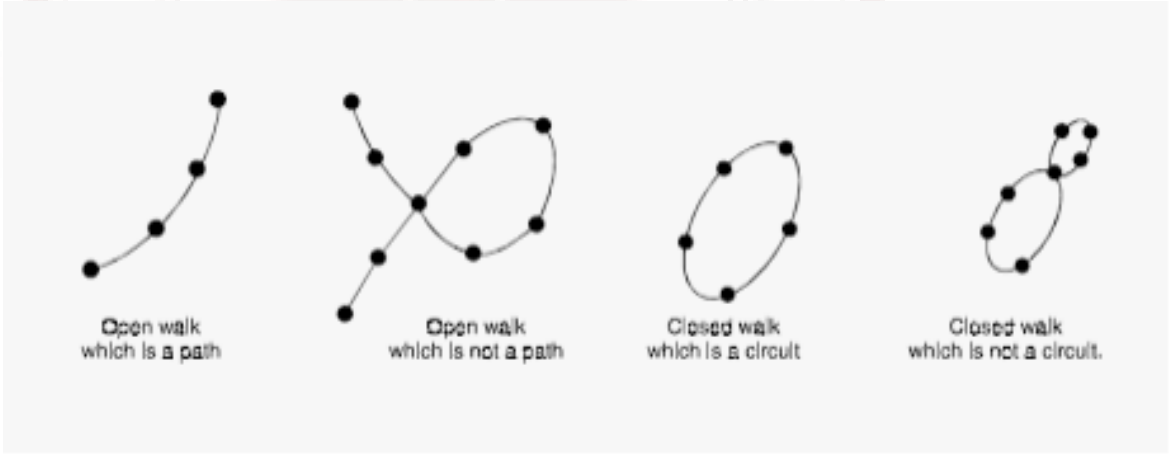
MAX AND MIN DEGREE OF A GRAPH

The maximum degree of a graph, G, is the greatest degree over all of the vertices in G. The minimum degree of a graph is the least degree over all of the vertices in G. The maximum degree is denoted $\triangle$(G), while the minimum degree is denoted ϕG).

Definition

ORDER AND SIZE OF A GRAPH

The number of vertices in a given graph G, denoted $|V(G)|$, is called the order of G and is denoted Ord(G), or —G—. The number of edges in G, denoted $|E(G)|$ is called the size of G.



Definition

COMPLETE GRAPH

A graph in which every pair of distinct vertices is adjacent is called a complete graph. A complete graph of order n is denoted K_n.

Definition

WALK
For two, not necessarily distinct, vertices u and v in a graph G, a (u → v) walk W in G is a sequence of vertices in G, beginning with u and ending at v such that consecutive vertices in W are adjacent in G.

Definition

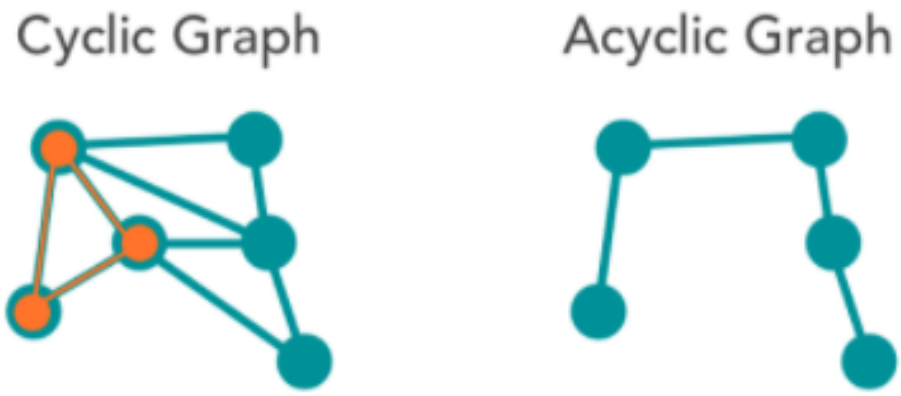
CLOSED WALK
A walk whose initial and terminal vertices are the same is a closed walk

Definition

PATH
A walk in a graph G in which no vertex is repeated is called a path.

Definition

CYCLE
A walk whose initial and terminal vertices are the same and every other vertex is distinct is called a cycle.



Definition

HAMLITONIAN PATH

A path in a given graph G that contains every vertex of G is called a Hamiltonian Path of G.

Definition

HAMLITONIAN CYCLE

A cycle in a given graph G that contains every vertex of G is called a Hamiltonian Cycle of G.

Definition

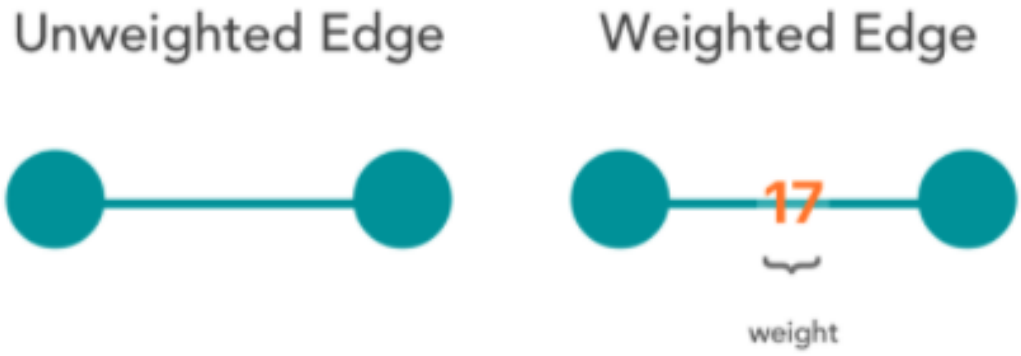
CONNECTED AND DISCONNECTED GRAPHS

A graph G is connected if G contains a (u $\rightarrow$ v) walk for every two vertices u and v of G, otherwise G is disconnected.

Definition

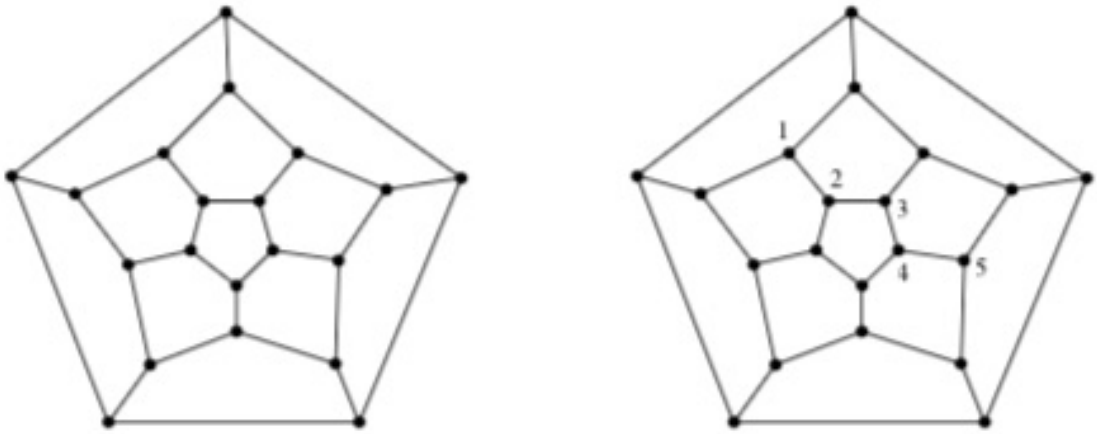
WEIGHTED GRAPH

If each edge in a given graph G is assigned a real number (called the weight of the edge), then G is a weighted graph.



3 THE HISTORY

Although the exact origins of the Traveling Salesman Problem are unclear, the first example of such a prob- lem appeared in the German handbook Der Handlungsreisende - Von einem alten Commis - Voyageur for salesman traveling through Germany and Switzerland in 1832.
This handbook was merely a guide, since it did not contain any mathematical language. People started to realize that the time one could save from creating optimal paths is not to be overlooked, and thus there is an advantage to figuring out how to create such optimal paths.
This idea was turned into a puzzle sometime during the 1800's by W. R. Hamilton and Thomas Kirkman. Hamilton's Icosian Game was a recreational puzzle based on finding a Hamiltonian cycle in the Dodecahe- dron graph as seen below.



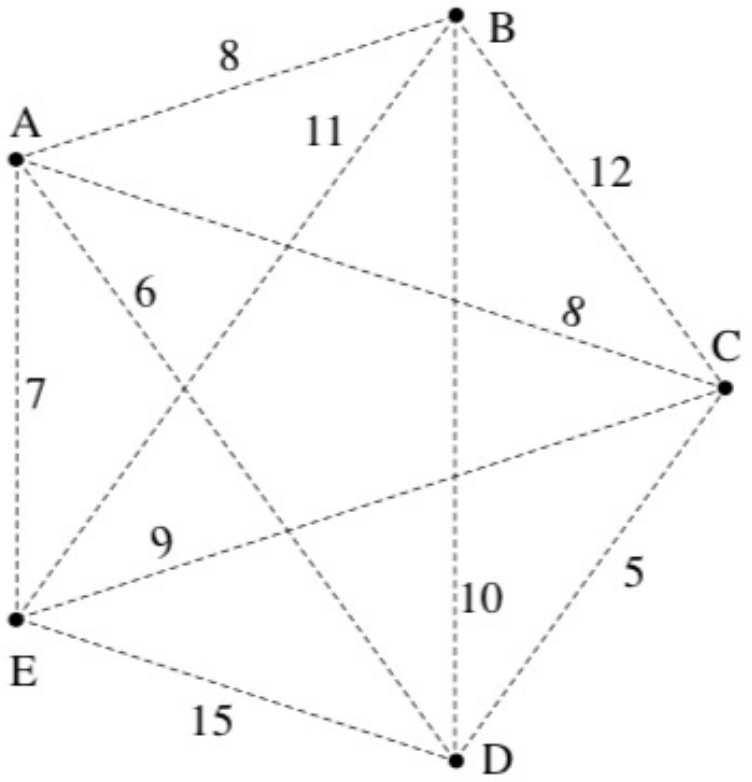
The Traveling Salesman Problem, as we know and love it, was first studied in the 1930's in Vienna and Harvard as explained in [3]. Richard M. Karp showed in 1972 that the Hamiltonian cycle problem was NP- complete, which implies the NP-hardness of TSP (see the next section regarding complexity). This supplied a mathematical explanation for the apparent computational difficulty of finding optimal tours.
The current record for the largest Traveling Salesman Problem including 85,900 cities, was solved in 2006

4 SOME KNOWN ALGORITHMS

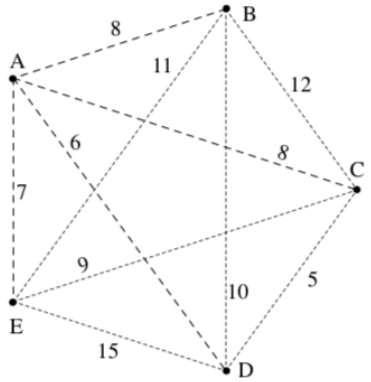
Since a globally optimal solution has not been found, there are many algorithms that give locally optimal solutions.

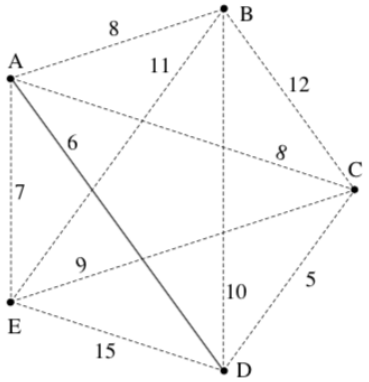
4.1 NEAREST-NEIGHBOR ALGORITHM

The Nearest-Neighbor Algorithm is a type of greedy algorithm which means that at every stage of the al- gorithm we pick the best possible edge to use. In other words, we make a locally optimal decision in every step of the algorithm. Another thing to note about this algorithm is that we need to pick a starting vertex and if we choose different vertices we might get different Hamiltonian Cycles, as illustrated in the following examples. We will think of this starting vertex as the Traveling Salesman's Home.

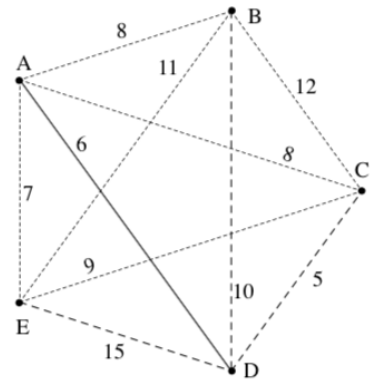


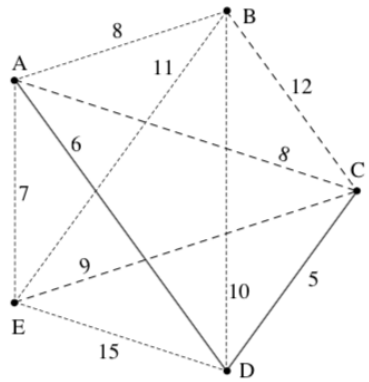
We start with a weighted complete graph of order 5 as our example. First, we see what the Nearest-Neighbor
Algorithm gives us if we start with vertex A. In Stage 1 we can see that we have four edges to choose from.





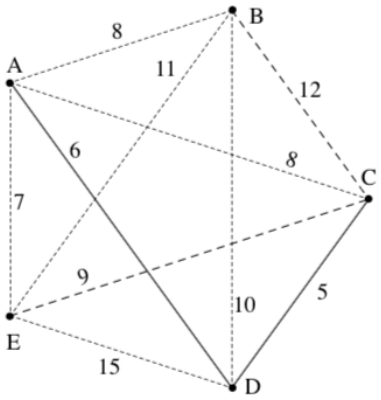
Edge AD has the least weight, 6, so we will use that edge, and ignore the others for now. Then, as we can see in Stage 3, we will look at all of the vertices incident with D and again pick the edge with the least weight. In this case, we have edge DC with a weight of 5.

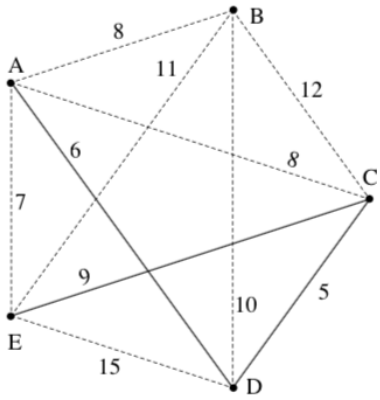




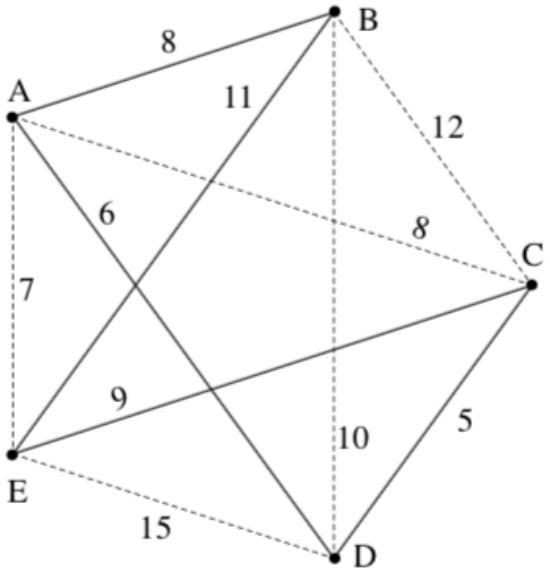
In stage 4 we consider the edges incident with vertex C. Note that the edge with the least weight, eight, is edge CA. However, recall that our salesman needs a Hamiltonian Cycle, which means he does not want to travel to any vertex more than once. We ignore edges to vertices that have already been

travelled to, until our salesman can travel home, and choose the edge with the least weight of 9, edge CE.

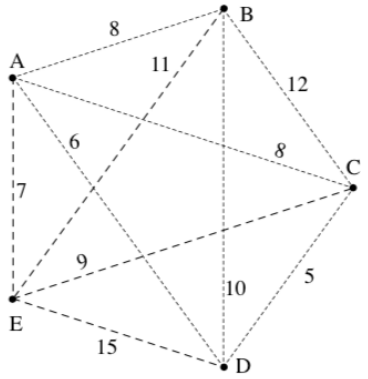


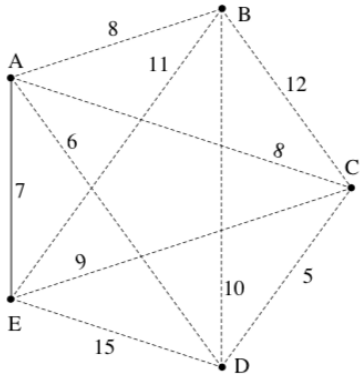


At this point we technically have three edges to choose from: ED, EA, and EB. Again, we do not want to travel to vertices that we have already traveled to, unless we are at the end of our Hamiltonian Cycle, and thus we must choose edge EB, with a weight of 11. Our last step will be to choose the last edge BA that brings our traveling salesman back home. The total trip, when starting at vertex A, has a weight of 39.



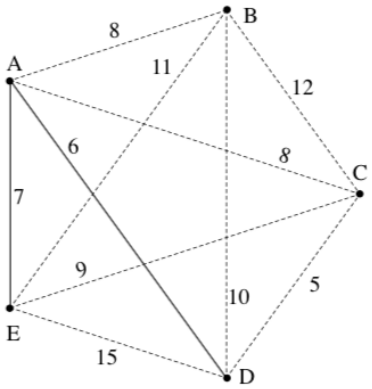
We mentioned before that if we start at a different vertex we might get different weights for the same graph. Instead of starting at vertex A, let the traveling salesman start at vertex E.

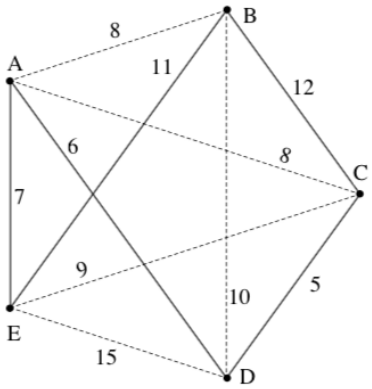




First, we choose edge EA with a weight of 7. Then we look at the vertices from vertex A: AB, AC, and AD and choose edge AD with a weight of 6. We continue in the same way that we did before and obtain the graph in figure 5.12 with a total weight or 35, which is a smaller weight than when we

started with vertex A.





Pseudocode

Initialize: Pick a vertex $u_i \in V(G)$

Form a path $P = \{u_i\}$ and put $T = V(P)$.

Put $r = 1$ and $v_r = u_i$

Iteration:

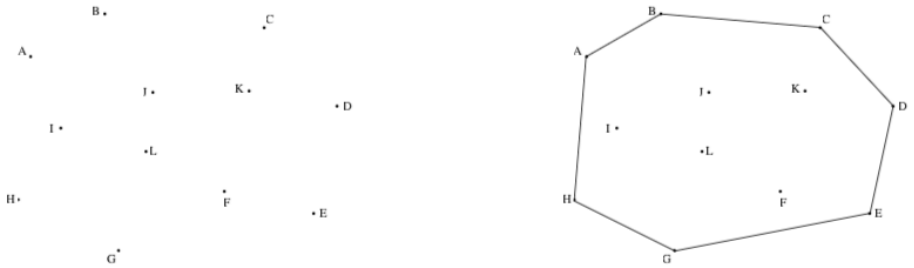
1. Choose $u_j \in V(G) \setminus T$ such that it is the minimum distance from v_r.
   (Ties may be broken arbitrarily.)
2. Put $v_{r+1} = u_j$ and let $P = \{u_i \to u_j\}$
3. Put $r = r + 1$ and repeat until $T = V(G)$

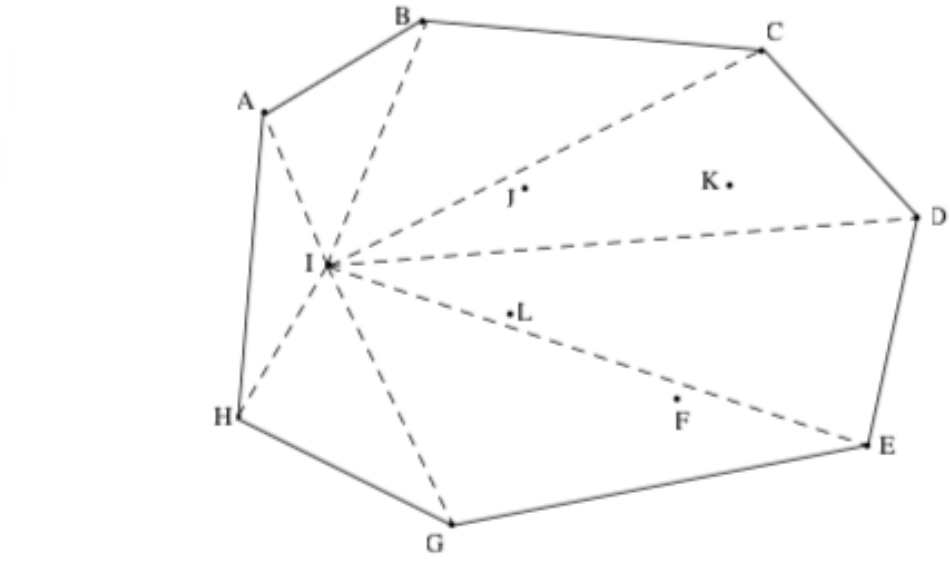
Output: A Hamiltonian Cycle is the path with the last vertex of P made adjacent to the first.

4.2 GEOMETRIC ALGORITHM

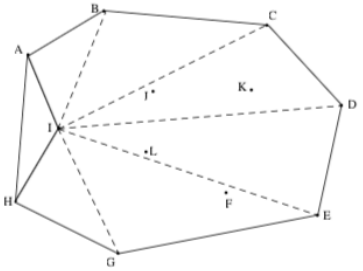
For this example we will start with 12 destinations for our traveling salesman.
In the Geometric algorithm it is assumed that we have a complete graph. Recall that a complete graph means that each vertex is adjacent to every other vertex, but note that we will not draw each edge in the figures in order to make the algorithm more clear.
The first step is to create the convex hull around all of the destinations, then draw the edges that connect these surrounding vertices as seen in Stage 2.

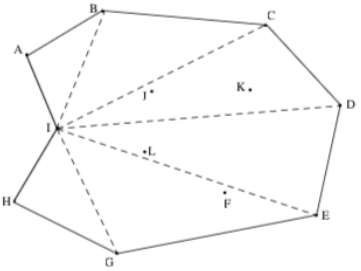
We call the vertices that are not on the border of the convex hull towns. In Figure 5.25 above, we can see that the towns are F, I, J, K, L. We take the first town, vertex I, and draw the edges from that vertex to every other vertex in the convex hull, as seen in Stage 3.

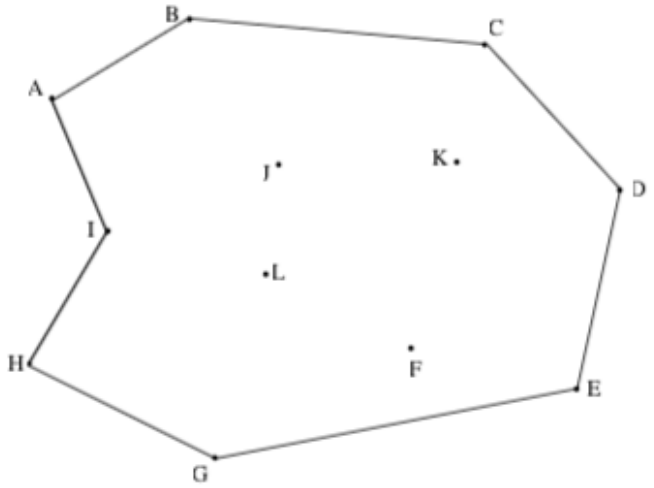




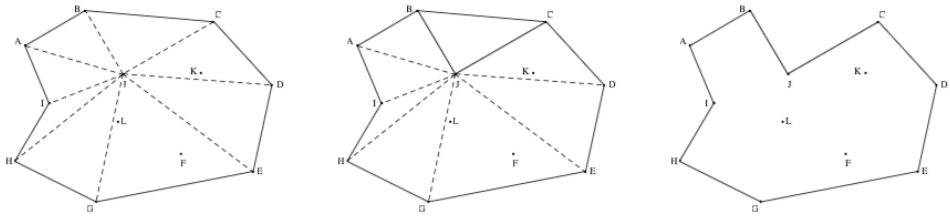
Next, we examine all of the angles centered at vertex I. We find the largest of these angles, highlight the edges that create that angle, then destroy the edge from the convex hull that is no longer needed as seen in Figure 5.28. As we destroy the edge that was on the convex hull we also destroy the temporary edges from vertex I to the other vertices as depicted in Stage 6.



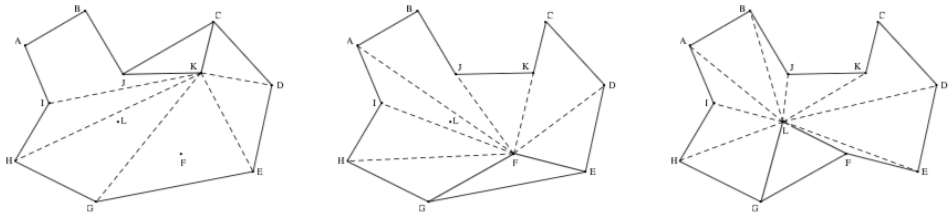


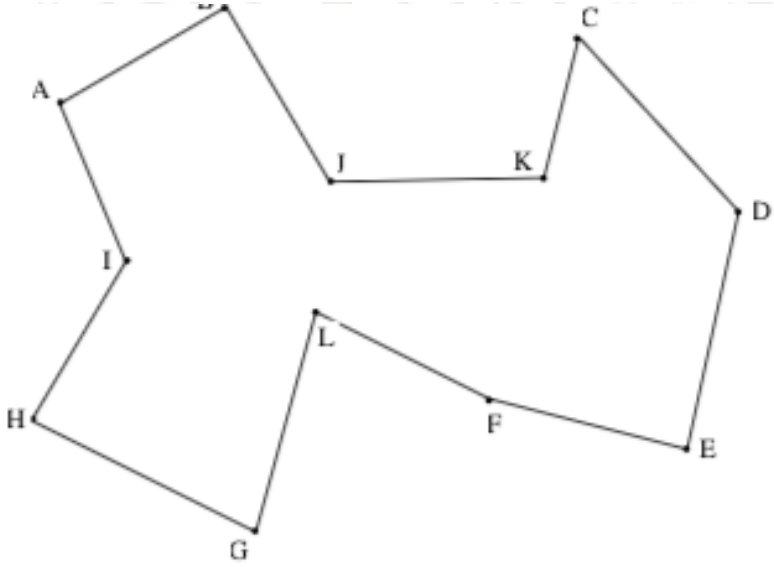


Now the border contains the vertices A, B, C, D, E, G, H, I and the set of towns is F, J, K, L. In the next step we draw all of the edges from town J to the vertices on the border including vertex I. As before, look at the angles created this way, choose largest one, delete the edge on the border, then delete all of the temporary edges incident to J.



We are now left with the graph in Figure 5.32. The algorithm continues in this way until there are no more towns left to put in the tour for our traveling salesman. In our example, we choose to look at vertices K, F, and L, in that order. In the event of a tie between two angles, the programmer is allowed to arbitrarily choose an angle and continue.





Pseudocode

Initialize: Create the convex hull around all of the vertices.

Let A be the set of all vertices that lie on the convex hull.

Let C be the set of edges in the convex hull.

Let T be the set of all vertices that do not lie on the convex hull.

1. Pick the vertex, $v_i \in T$ closest to the convex hull and create all of the edges incident with v_i and the vertices in A.
2. Measure all of the angles created this way. Let $v_i u_j$ and $v_i u_{j+1}$ be the edges that form the largest angle centered at v_i. Keep these edges, and destroy the rest of the edges that are not in C along with the edge $u_j u_{j+1}$.
3. Now $v_i \in A$.
4. Continue until $T = \emptyset$.

Output: A Hamiltonian Cycle with edges in C.

5 APPLICATIONS

5.1 FINDING DISTANCE BETWEEN COLLEGES

Application

Although one may not be a traveling salesman, there are other ways that one can use this information. For example, a student wants to drive to twenty colleges around the United States in the shortest way possible. Since the student is driving, he or she wants to minimize the distances between each of the schools. below shows the twenty schools that the student wants to visit, and below shows the distance between each of the colleges in alphabetical order.

The student in this situation could start anywhere, so let's look at Figure below and find the shortest distance between two colleges and pick one of those to start the journey. The distance between Yale University and Brown University is 103 miles by car, thus we will start at Brown University and head towards Yale University.

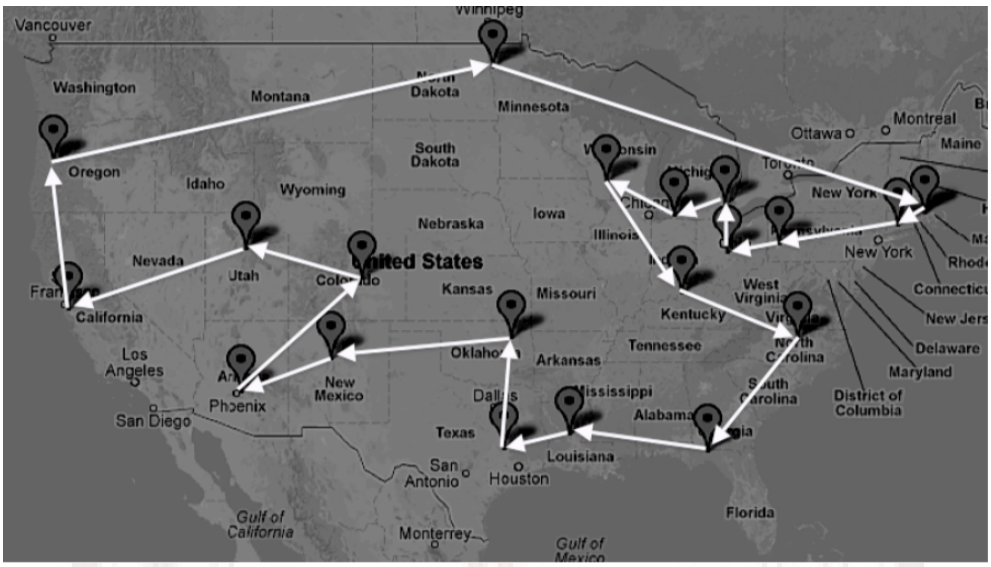
Colleges	Arizona State	Brigham Young	Brown	Colorado	Duke	Florida State	Louisiana	Louisville	Michigan	New Mexico	North Dakota	Notre Dame	Ohio	Oklahoma	Oregon	Pitt	Stanford	Texas A&M	Wisconsin	Yale
Arizona State	0	648	2625	549	2185	1898	1458	1752	1963	427	1743	1817	1899	1060	1148	2084	732	1095	1725	2524
Brigham Young	648	0	2363	481	2129	2030	1641	1594	1638	557	1214	1492	1710	1126	825	1861	811	1195	1375	2262
Brown	2625	2362	0	1965	669	1274	1541	920	744	2172	1623	875	720	1595	3085	543	3113	1734	1111	103
Colorado	549	481	1965	0	1667	1605	1194	1132	1242	431	963	1096	1280	664	1249	1464	1276	799	979	1866
Duke	2185	2129	669	1667	0	621	906	541	643	1733	1504	733	459	1187	2880	479	2791	1169	932	566
Florida State	1898	2030	1274	1605	621	0	443	662	978	1482	1669	925	839	1007	2855	929	2541	843	1107	1172
Louisiana	1458	1641	1541	1194	906	443	0	754	1106	1074	1447	976	968	638	2477	1148	2132	435	1027	1442
Louisville	1752	1594	920	1132	541	662	754	0	347	1293	1015	261	209	724	2319	389	2346	841	443	818
Michigan	1963	1638	744	1242	643	978	1106	347	0	1511	961	170	183	970	2361	287	2389	1124	388	688
New Mexico	427	557	2172	431	1733	1482	1074	1293	1511	0	1318	1363	1447	585	1378	1631	1063	641	1275	2071
North Dakota	1743	1214	1623	963	1504	1669	1447	1015	961	1318	0	813	1078	918	1571	1182	1882	1147	582	1583
Notre Dame	1817	1492	875	1096	733	925	976	261	170	1363	813	0	271	826	2215	373	2242	979	241	774
Ohio	1899	1710	720	1280	459	839	968	209	183	1447	1078	271	0	882	2453	192	2408	1049	504	621
Oklahoma	1060	1126	1595	664	1187	1007	638	724	970	585	918	826	882	0	1891	1066	1667	251	798	1495
Oregon	1148	825	3085	1249	2880	2855	2477	2319	2361	1378	1571	2215	2453	1891	0	2583	559	2042	2108	2984
Pitt	2084	1861	543	1464	479	929	1148	389	287	1631	1182	373	192	1066	2583	0	2612	1220	611	442
Stanford	732	811	3113	1276	2791	2541	2132	2346	2389	1063	1882	2242	2408	1667	559	2612	0	1701	2125	3012
Texas A&M	1095	1195	1734	799	1169	843	435	841	1124	641	1147	979	1049	251	2042	1220	1701	0	977	1634
Wisconsin	1725	1375	1111	979	932	1107	1027	443	388	1275	582	241	504	798	2108	611	2125	977	0	1011
Yale	2524	2262	103	1866	566	1172	1442	818	688	2071	1583	774	621	1495	2984	442	3012	1634	1011	0

5.1.1 BY NEAREST-NEIGHBOR ALGORITHM

We follow the Nearest-Neighbor algorithm in the first case. After traveling from Brown University to Yale University, looking at the column under Yale University in Figure above, the next smallest distance is to the University of Pittsburgh with a total driving distance of 442 miles. We add this destination to the route then look at the next closest college on our list remembering not to travel to a college that we have already visited.

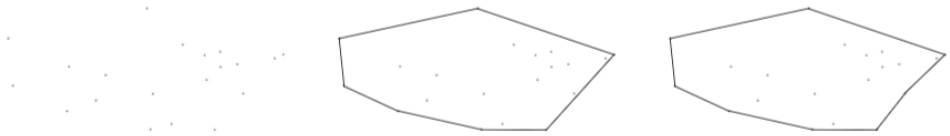
Order	College	Distance
1	Brown University	-
2	Yale University	103
3	University of Pittsburgh	442
4	Ohio State University	192
5	University of Michigan	183
6	University of Notre Dame	170
7	University of Wisconsin	241
8	University of Louisville	443
9	Duke University	541
10	Florida State University	621
11	Louisiana State University	443
12	Texas A&M University	435
13	University of Oklahoma	251
14	University of New Mexico	585
15	Arizona State University	427
16	University of Colorado	549
17	Brigham Young University	481
18	Stanford University	811
19	University of Oregon	559
20	University of North Dakota	1571
	Brown University	1623
	TOTAL DISTANCE	10671

below Shows the route that the student would take on the map (if the roads were a straight line, of course). If the student wants to map their route using graph theory in order to show his or her parents, he or she might draw this map.

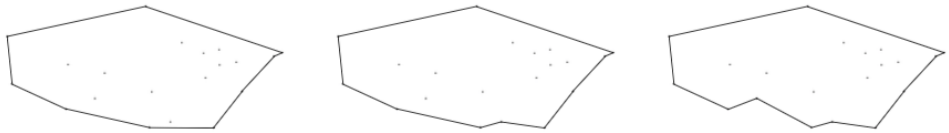


5.1.2 BY GEOMETRIC ALGORITHM

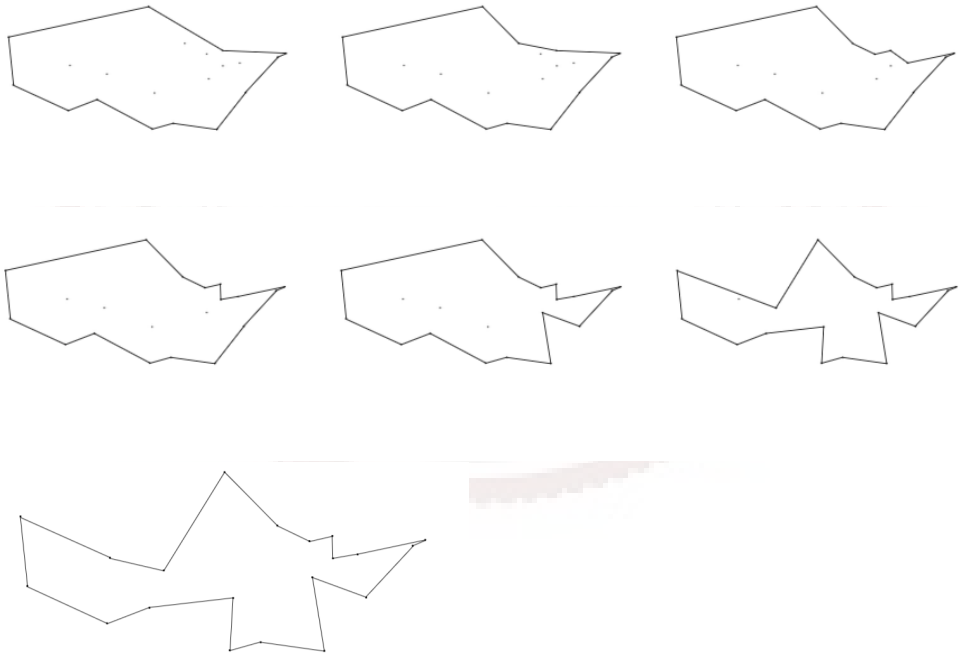
We have just seen that if the student chooses to use the Nearest Neighbor algorithm they will get the same route. If the student uses the Geometric algorithm, however, the journey is different and actually shorter.



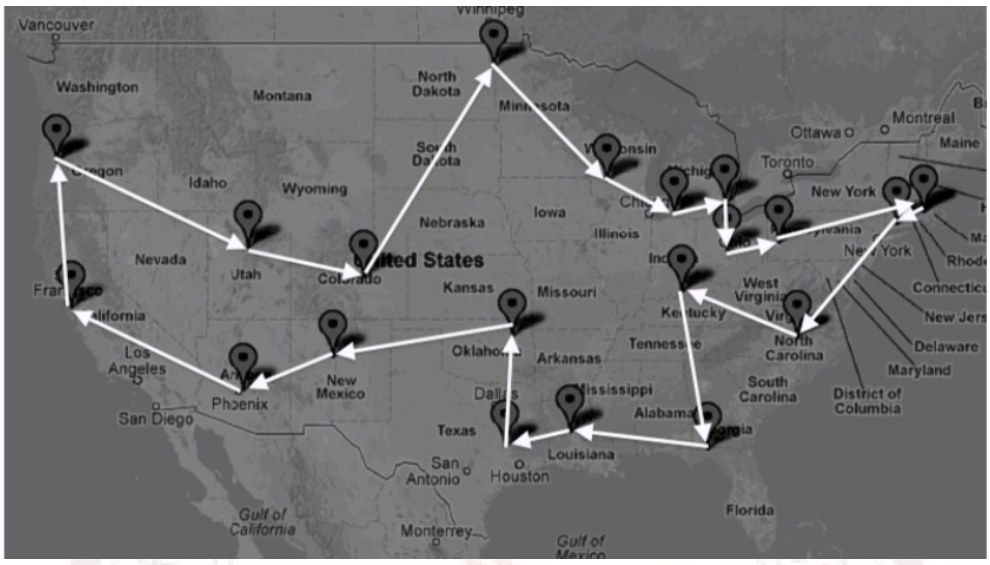
We can see the progression of the algorithm in the proceeding figures below. In each step, the college closest to the edge set at the beginning of each step is the town that is selected next in the algorithm.



At the start of the algorithm it seems as if the total distance will be greater than the distance traveled using the Nearest Neighbor algorithm However, look at the list of the distances as a whole. While the Nearest Neighbor seem to have taken the student across the United States in an effective manner , the journey back home was quite awful. The Geometric Algorithm takes the journey, as a whole, into consideration. This is further justified as we look at the chart of distances for the algorithm. Notice that not one of the distances exceeds 1000 miles. This will make both the student and his or her parents very happy.



Order	College	Distance
1	Brown University	-
2	Yale University	103
3	Duke University	566
4	University of Louisville	541
5	Florida State University	662
6	Louisiana State University	443
7	Texas A&M University	435
8	University of Oklahoma	251
9	University of New Mexico	585
10	Arizona State University	427
11	Stanford University	732
12	University of Oregon	559
13	Brigham Young University	825
14	University of Colorado	481
15	University of North Dakota	963
16	University of Wisconsin	582
17	University of Notre Dame	241
18	University of Michigan	170
19	Ohio State University	183
20	University of Pittsburgh	192
	Brown University	543
	TOTAL DISTANCE	9484



5.1.3 CONCLUSION

Conclusion

We have seen in college visit example that the Geometric Algorithm for the Traveling Salesman Problem produced better results than the Nearest-Neighbor This may be true in general, but more research is necessary.

5.2 C PROGRAM TO SOLVE TSP

Application

Compute the solutions of all subproblems starting with the smallest. Whenever computing a solution requires solutions for smaller problems using the above recursive equations, look up these solutions which are already computed. To compute a minimum distance tour, use the final equation to generate the 1st node, and repeat for the other nodes. For this problem, we cannot know which subproblems we need to solve, so we solve them all.

5.2.1 CODE

```
#include <stdio.h>
int matrix[25][25], visited_cities[10], limit, cost = 0;

int tsp(int c)
{
int count, nearest_city = 999;
int minimum = 999, temp;
for(count = 0; count < limit; count++)
{
if((matrix[c][count] != 0) && (visited_cities[count] == 0))
{
if(matrix[c][count] < minimum)
{
minimum = matrix[count][0] + matrix[c][count];
}
temp = matrix[c][count];
nearest_city = count;
}
}
if(minimum != 999)
{
cost = cost + temp;
}
return nearest_city;
}

void minimum_cost(int city)
```

```
{
int nearest_city;
visited_cities[city] = 1;
printf("%d ", city + 1);
nearest_city = tsp(city);
if(nearest_city == 999)
{
nearest_city = 0;
printf("%d", nearest_city + 1);
cost = cost + matrix[city][nearest_city];
return;
}
minimum_cost(nearest_city);
}

int main()
{
int i, j;
printf("Enter Total Number of Cities:\t");
scanf("%d", &limit);
printf("\nEnter Cost Matrix\n");
for(i = 0; i < limit; i++)
{
printf("\nEnter %d Elements in Row[%d]\n", limit, i + 1);
for(j = 0; j < limit; j++)
{
scanf("%d", &matrix[i][j]);
}
visited_cities[i] = 0;
}
printf("\nEntered Cost Matrix\n");
for(i = 0; i < limit; i++)
{
printf("\n");
for(j = 0; j < limit; j++)
{
printf("%d ", matrix[i][j]);
}
}
printf("\n\nPath:\t");
minimum_cost(0);
printf("\n\nMinimum Cost: \t");
printf("%d\n", cost);
return 0;
}
```

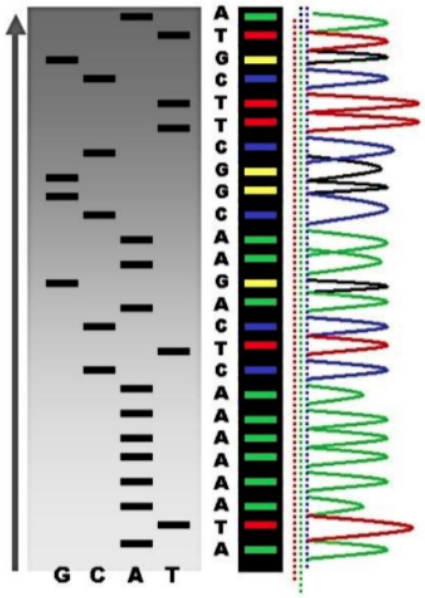
5.2.2 OUTPUT WITH EXAMPLE

```
vidipts-MacBook-Pro:~ vidiptvashist$ /Users/vidiptvashist/Desktop/CodeBlocks.app/Content
op/c_program/sudoku/bin/Debug/sudoku
Enter Total Number of Cities:   4

Enter Cost Matrix

Enter 4 Elements in Row[1]
4
5
7
3

Enter 4 Elements in Row[2]
2
5
6
8

Enter 4 Elements in Row[3]
6
5
4
3

Enter 4 Elements in Row[4]
1
4
6
7

Entered Cost Matrix

4 5 7 3
2 5 6 8
6 5 4 3
1 4 6 7

Path:   1 4 3 2 1

Minimum Cost:   16

Process returned 0 (0x0)   execution time : 17.462 s
Press ENTER to continue.
```

5.3 DNA SEQUENCING

Application

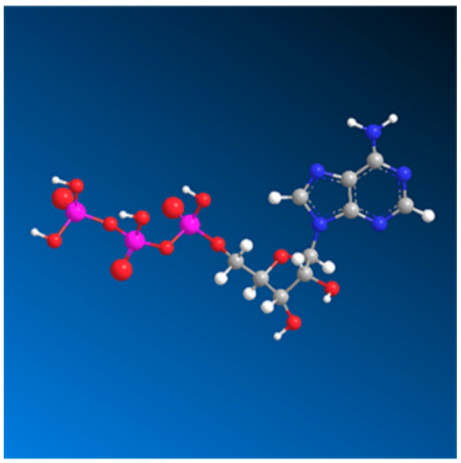
1. Discover the exact sequence of nucleotides in a short DNA segment
2. Biological experiment gives spectrum with errors
3. Goal: Computationally reconstruct the exact sequence from that spectrum



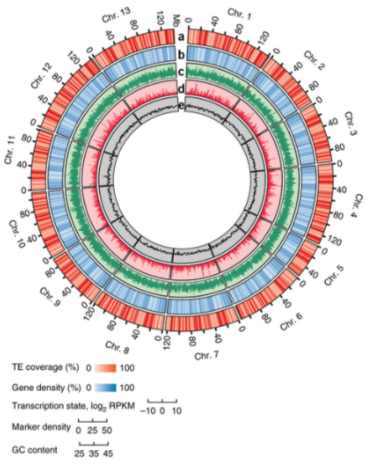
5.3.1 SOLUTION USING TSP

Solution wrt TSP

1. Vertices are l-mer structures
2. Minimize cost (overlapping of l-mers: from AATCG to
3. CGGTG overlapping is AATCGGTG)
4. Maximize profit(number of l-mers)



29



30

6 CONCLUSION

Conclusion

1. We have seen for 1st application college visit example that the Geometric Algorithm for the Traveling Salesman Problem produced better results than the Nearest-Neighbor
2. And we can use C programming to help find solution of given TSP problem.
3. Also TSP solving method are use extensively in DNA sequencing problem.