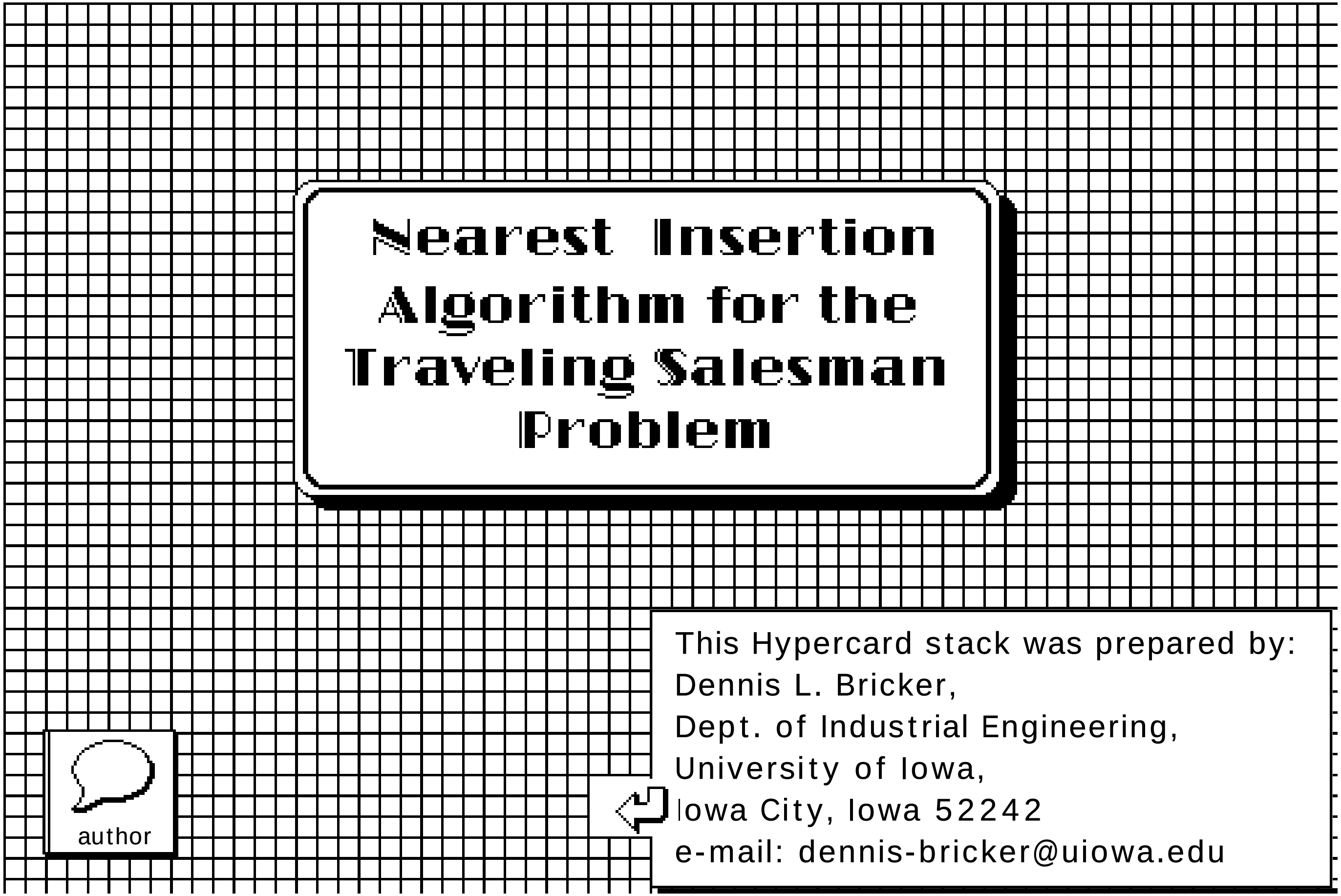

Nearest Insertion Algorithm for the Traveling Salesman Problem
This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dennis-bricker@uiowa.edu
author

The "Nearest Insertion" heuristic algorithm constructs a tour, starting with an arbitrary node.

Each step begins with a subtour, and selects the node which is *nearest* to the set of nodes on the subtour to be added to the subtour.

After selecting the node k to be added, an edge (i,j) is selected and the edges (i,k) and (k,j) then replace the edge (i,j).

The edge (i,j) is selected so as to minimize the increase in the length of the subtour, i.e.,

$$d_{ik} + d_{kj} - d_{ij}$$

The "Nearest Insertion" heuristic constructs a tour for the TSP as follows:

step 0: Select an initial node $\hat{i}$.

Let N' denote the set of nodes $N - \{\hat{i}\}$

Let $T = \{(\hat{i}, \hat{i})\}$

step 1: Let $\hat{j} = \underset{j \in N'}{\operatorname{argmin}} \left[\min_{i \in T} \{d_{ij}\} \right]$

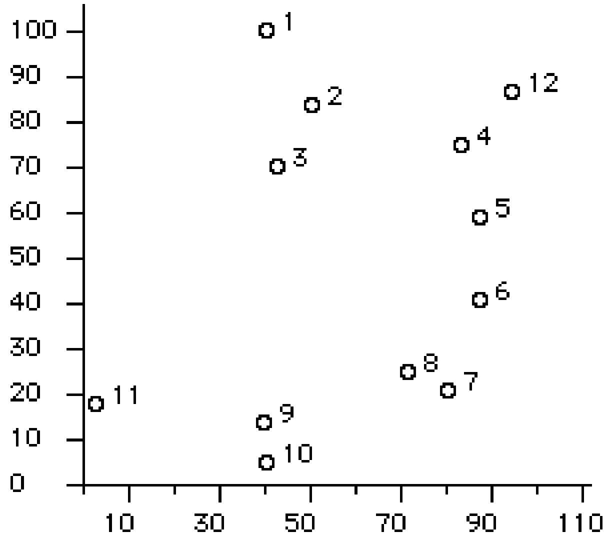
step 2: Let $(i', i'') = \underset{(i_1, i_2) \in T}{\operatorname{argmin}} \{d_{i_1 j} + d_{j i_2} - d_{i_1 i_2}\}$

step 3: Replace arc (i_1, i_2) in the tour T with the pair of arcs $(i_1, \hat{j})$ and $(\hat{j}, i_2)$.
Let $N' = N' - \{\hat{j}\}$ and $\hat{i} = \hat{j}$.

step 4: If $N' = \emptyset$, STOP. Else return to step 1.

Example

Random Symmetric TSP
(seed= 133398)



Distances

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
2	19	0	16	34	45	57	70	63	71	80	82	44
3	30	16	0	41	46	54	62	54	56	65	66	55
4	50	34	41	0	16	34	54	51	75	82	99	16
5	62	45	46	16	0	18	39	38	66	72	94	29
6	75	57	54	34	18	0	21	23	55	59	88	47
7	89	70	62	54	39	21	0	10	42	43	78	67
8	81	63	54	51	38	23	10	0	34	37	69	66
9	86	71	56	75	66	55	42	34	0	9	37	91
10	95	80	65	82	72	59	43	37	9	0	40	98
11	90	82	66	99	94	88	78	69	37	40	0	115
12	56	44	55	16	29	47	67	66	91	98	115	0

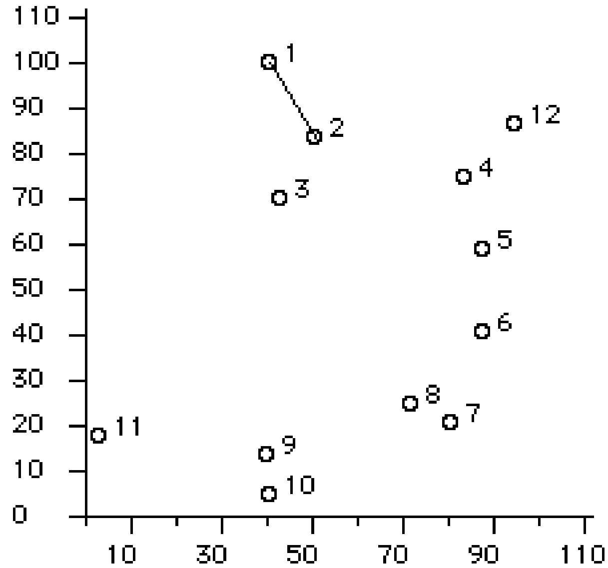
Let's arbitrarily begin the tour with node #1,
i.e., T = {1},
and N' = {2,3,4,5,6,7,8,9,10,11,12}

The nearest node to T={1} is node 2.

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56

Nearest Insertion

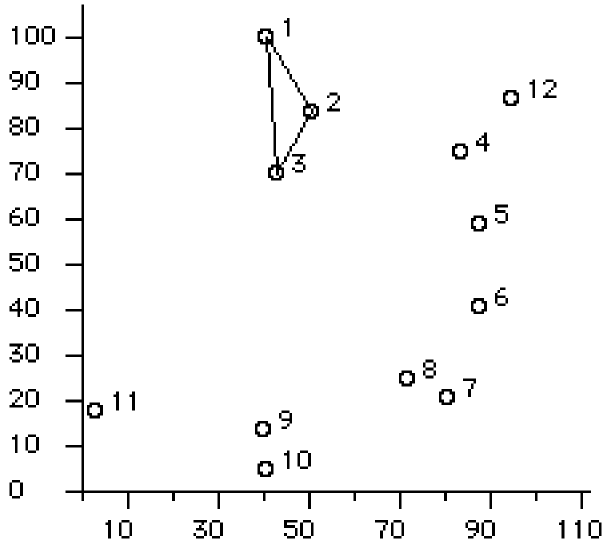
(Starting with node #1)



Insert node 2

The nearest node to the subtour T = {1,2} is node #3.

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
2	19	0	**16**	34	45	57	70	63	71	80	82	44



Insert node 3

The nearest node to T = {1,2,3} is node #4.

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	19	30	50	62	75	89	81	86	95	90	56
2	19	0	16	34	45	57	70	63	71	80	82	44
3	30	16	0	41	46	54	62	54	56	65	66	55

Node #4 can be inserted in the tour in 3 different ways:

1 → 4 → 2

2 → 4 → 3

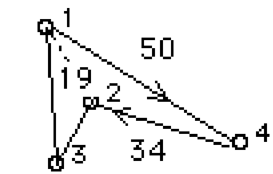
3 → 4 → 1

We insert node #4 in such a way as to minimize the increase in tour length:

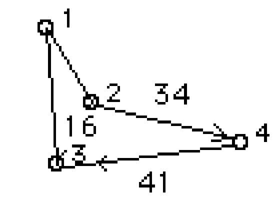
distances

	1	2	3	4
1	0	19	30	50
2	19	0	16	34
3	30	16	0	41
4	50	34	41	0

Increase in tour length

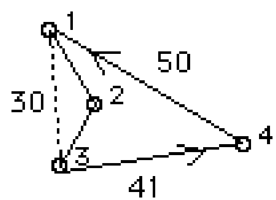


$50 + 34 - 19 = 65$

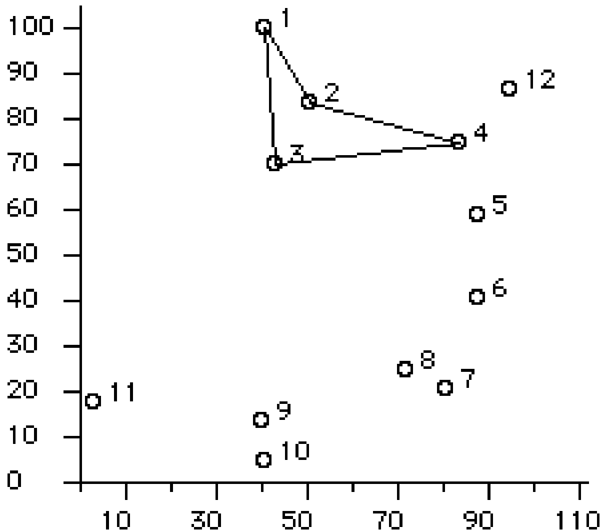


$34 + 41 - 16 = 59$

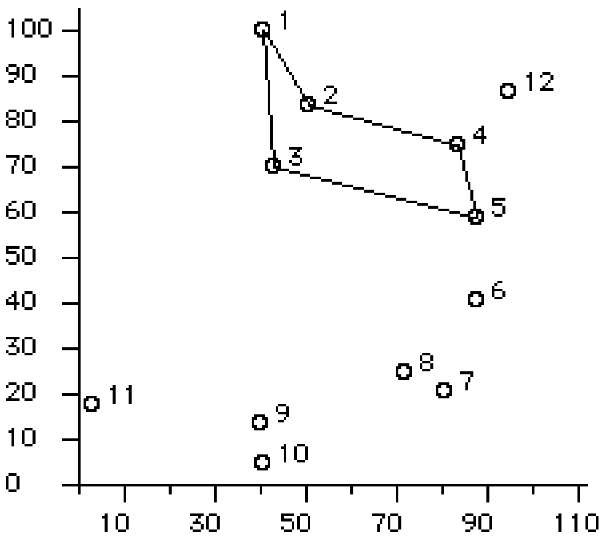
minimum!



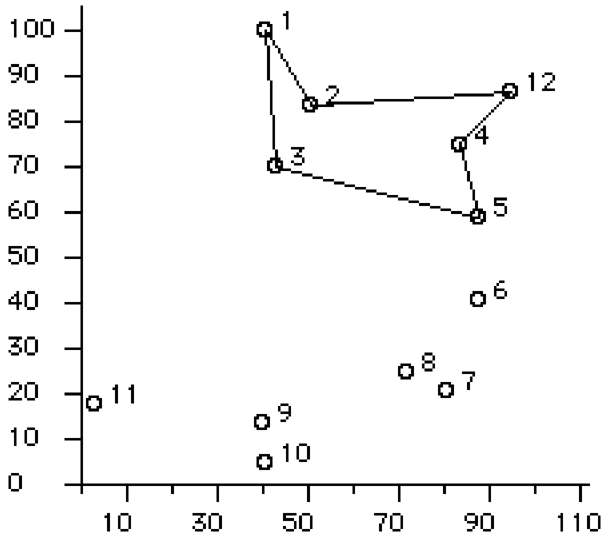
$41 + 50 - 30 = 61$



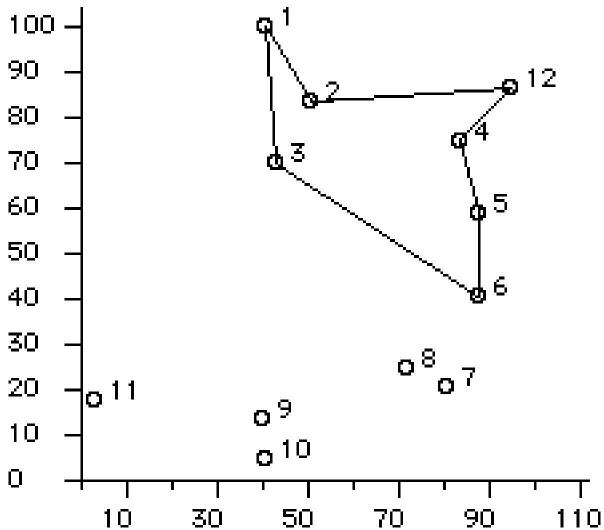
Insert node 4



Insert node 5



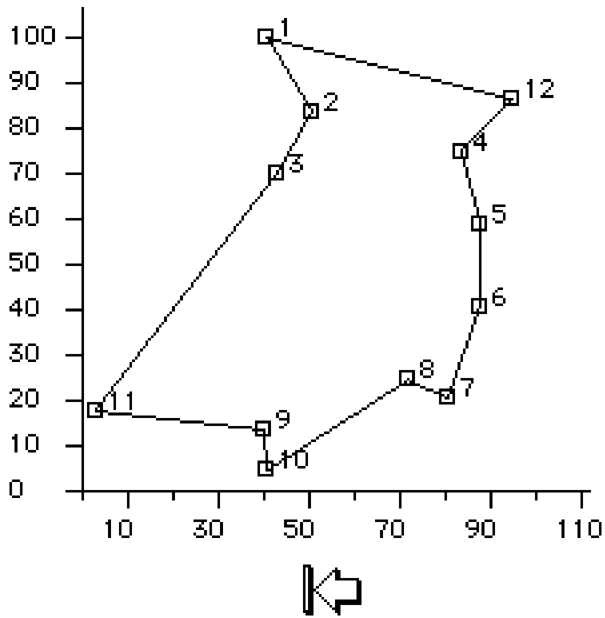
Insert node 12



Insert node 6

... *etc.*

Nearest Insertion Tour: 6 7 8 10 9 11 3 2 1 12 4 5 6, with length 321



Note that the final tour varies according to the initial node in the tour!