Second Direction: Heuristics

Characteristics:
- based on common sense, intuition; greedy
- no rigorous mathematical analysis
- time-efficient
- don't guarantee optimal soln
- hopefully produce fairly good solns at least some of the time

Ex: Nearest Neighbor Heuristic for TSP
- Visit the nearest node not yet visited.

"Bad" examples:
1) \[ \begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array} \] 
2) \[ \begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array} \]

• Usually, there is a benchmark of problems on which the algorithm is tested.
  • For TSP, such a benchmark is TSPLIB.
  On problems of TSPLIB, the costs of Nearest Neighbor outputs are on average 1.26 times the costs of optimal tours.

Third Direction: Approximation Algorithms

Characteristics:
- time-efficient (sometimes not as efficient as)
- don't guarantee optimal soln
- guarantee good soln within some factor of the optimum
- rigorous mathematical analysis to prove the approximation guarantee
- often use algorithms for related problems as subroutines

\[ 1 - \text{approximation algorithm:} \]
\[ \text{(for minimization problem)} \]
\[ \text{For any instance of the problem} \]
\[ \frac{Z_{app}}{Z_{opt}} \leq 1 \]
\[ Z_{app} = \text{cost of algorithm output} \]
\[ Z_{opt} = \text{cost of optimum soln} \]

\[ 2 - \text{approximation algorithm for TSP:} \]
\[ \text{(given triangle inequality for distance)} \]

Step 1: Find Min Spanning Tree for the same problem instance (using Prim's algorithm)
Step 2: Use shortcuts to get a tour from the tree.

Ex: 

Step 1: 

Step 2: 

Start from this node