Dynamic Programming

How to design DP Algos

- Define "Subproblems"

→ Relate the subproblems using a recurrence relation.

→ Design an algorithm: 0) Base Cases.
1) Order of solving subproblems

EXAMPLE: COMPUTING OPTIMAL STRATEGIES

→ GAME with two different MOVES:

MOVE A → with prob. $\frac{1}{2}$: +1 point
MOVE A → with prob $\frac{1}{2}$: −1 point

MOVE B → with prob $\frac{1}{2}$: +10 points
MOVE B → with prob $\frac{1}{2}$: −10 points.

NEED to PLAY K Moves

→ WIN if endup with > 100 points.

Find Strategy that maximises the Probability of WINNING.

WIN
- if #points in perfect square

What is a strategy?

Strategy:

$$S: \left(\{ \text{points you have} \}, \{ \text{moves are left} \} \right)$$

$\longrightarrow$ MOVE A

or

MOVE B.

1) $T\{a \text{ points}, b \text{ moves left}\}$ = Probability of the optimal strategy winning the game.

2) $T[a,b] = \max$ { MOVE A $\frac{1}{2} T[a+1, b-1] + \frac{1}{2} T[a-1, b-1]$; MOVE B $\frac{1}{2} T[a+10, b-1] + \frac{1}{2} T[a-10, b-1]$ }

BASECASE

$T[a, 0] = 1$ if $a > 100$

BASE CASE

- $T[a, 0] = 1$ if $a > 100$

---

for $b = 1$ to K moves left

for $a = -10K$ to $10K$

$$T[a,b] = \max \begin{cases} \text{MOVE A} & \frac{1}{2} T[a+1, b-1] + \frac{1}{2} T[a-1, b-1] \\ \text{MOVE B} & \frac{1}{2} T[a+10, b-1] + \frac{1}{2} T[a-10, b-1] \end{cases}$$

$$\Pr\left[\text{win} \mid \begin{matrix} a \text{ points left} \\ b \text{ moves} \end{matrix}\right]$$

MOVE A

$$= \Pr\left[\begin{matrix}\text{Move A} \\ +1\end{matrix}\right] \cdot \Pr\left[\text{win} \mid \begin{matrix} a \text{ points} \\ b \text{ moves} \end{matrix}, \begin{matrix}+1 \\ -1\end{matrix}\right]$$

$$+ \Pr\left[\begin{matrix}\text{Move} \\ -1\end{matrix}\right] \cdot \Pr\left[\text{win} \mid \begin{matrix} a-1 \\ b-1 \end{matrix}\right]$$

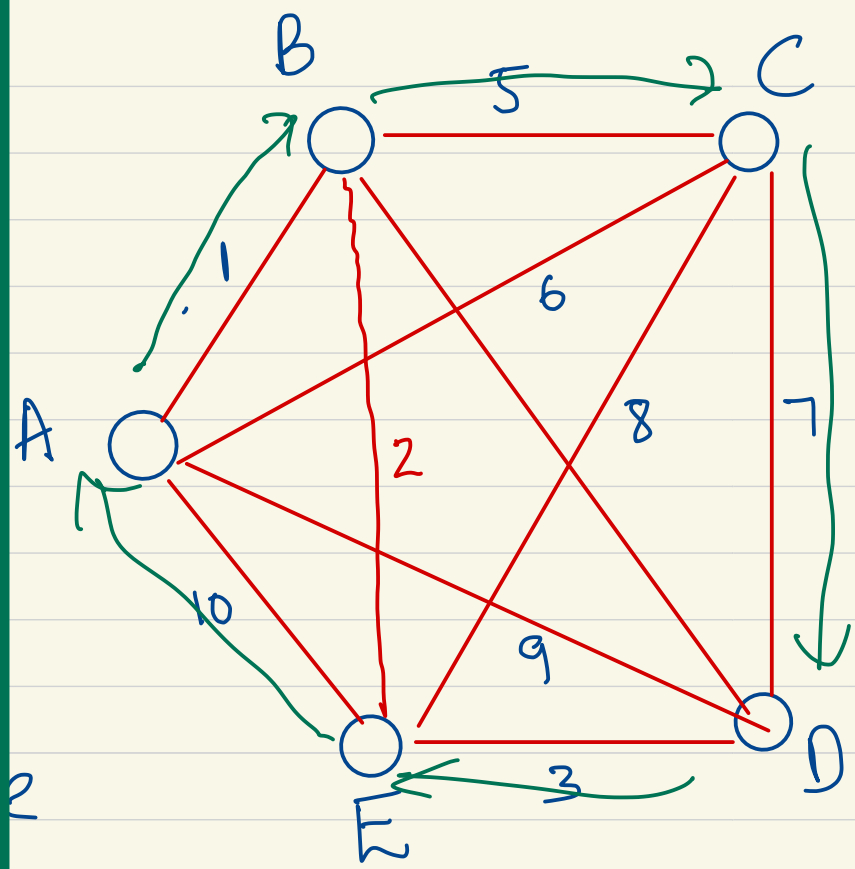
TRAVELLING SALESMAN PROBLEM:

INPUT: n cities with distances $\{d_{ij}\}$.

GOAL: Find the shortest tour 1) starting at A

2) visiting every city exactly once

3) ending at A.



$A \to B \to C \to D \to E \to A$

$1+5+7+3+10 = 25$

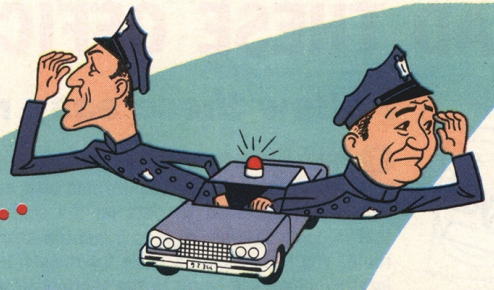
$A \to C \to D \to B \to E \to A$

$6+7+3+2+1 = 19$

NaiveAlg: $\Theta(n!)$ time.

GOAL: $\Theta(n^2 \cdot 2^n)$ algo

HELP! WE'RE LOST!



HELP "CAR 54"...AND WIN CASH

54...$1,000 PRIZES
ONE...$10,000 GRAND PRIZE

Map by Rand McNally

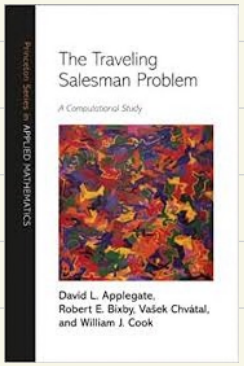
Help Toody and Muldoon find the shortest round trip route to visit all 33 locations shown on the map.

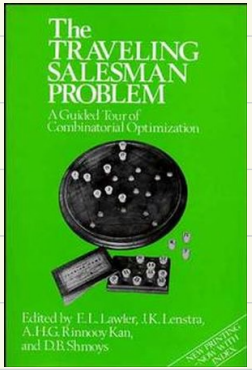
All you do is draw connecting straight lines from location to location to show the shortest round trip route.

HERE'S THE CORRECT START...

Begin at Chicago, Illinois. From there, lines show correct route as far as Erie, Pennsylvania. Next, do you go to Carlisle, Pennsylvania or Wana, West Virginia? Check the easy instructions on back of this entry blank for details.

OFFICIAL RULES ON REVERSE SIDE

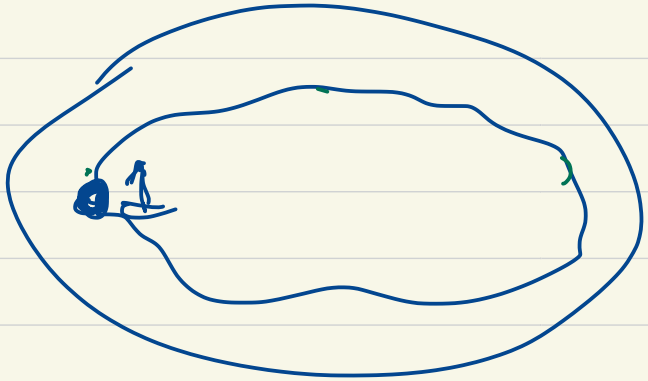


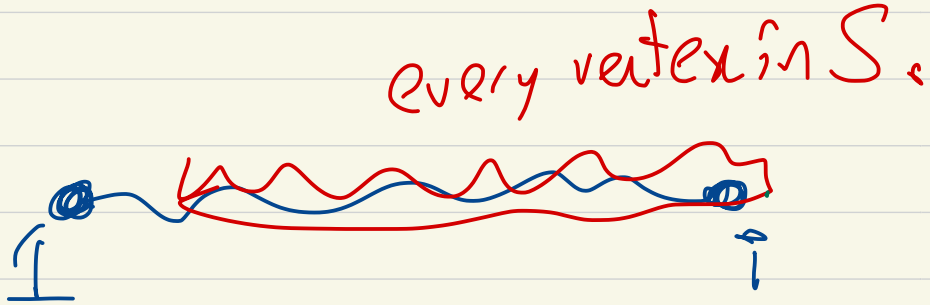


SUBPROBLEM: For S such that $[i \in S]$ and $[1 \in S]$

$T[S, i]$ = length of shortest path/tour that

1) starts at 1
2) visits every node in subset S
3) ends at i

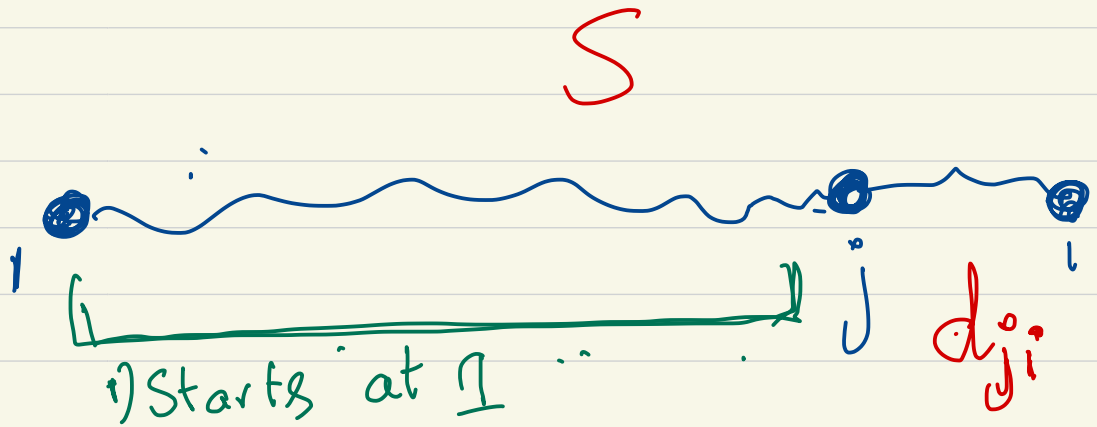




RECURRENCE RELATION

Let

$j :=$ penultimate vertex on the path $T[S, i]$.



1) Starts at 1
2) ends at j
3) visits $S \setminus \{i\}$

$= T[S \setminus \{i\}, j]$

$$T[S, i] = \min_{j \in S \setminus \{i\}} \left[T[S \setminus \{i\}, j] + d_{ij} \right]$$

Base Case: $T[\{1\}, 1] = 0$

ALG:

$2^n \rightarrow$ (for set_size = 2 to n)

$\times$ for subsets S, $|S| =$ set_size, $1 \in S$

n for $i \in S$

$\times$ $T[S,i] = \min_{\substack{j \in S \\ j \neq i}} \{T[S \setminus \{i\}, j] + d_{ji}\}$

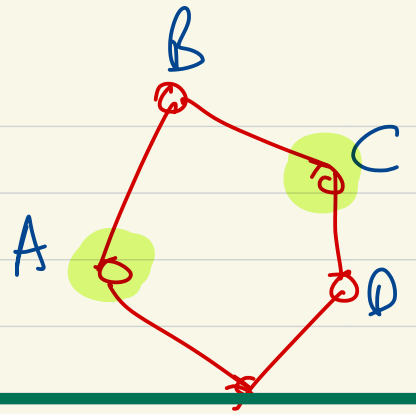
n

$\|$

$\Theta(2^n \cdot n^2)$

INDEPENDENT SET:

INPUT: Graph $G = (V, E)$



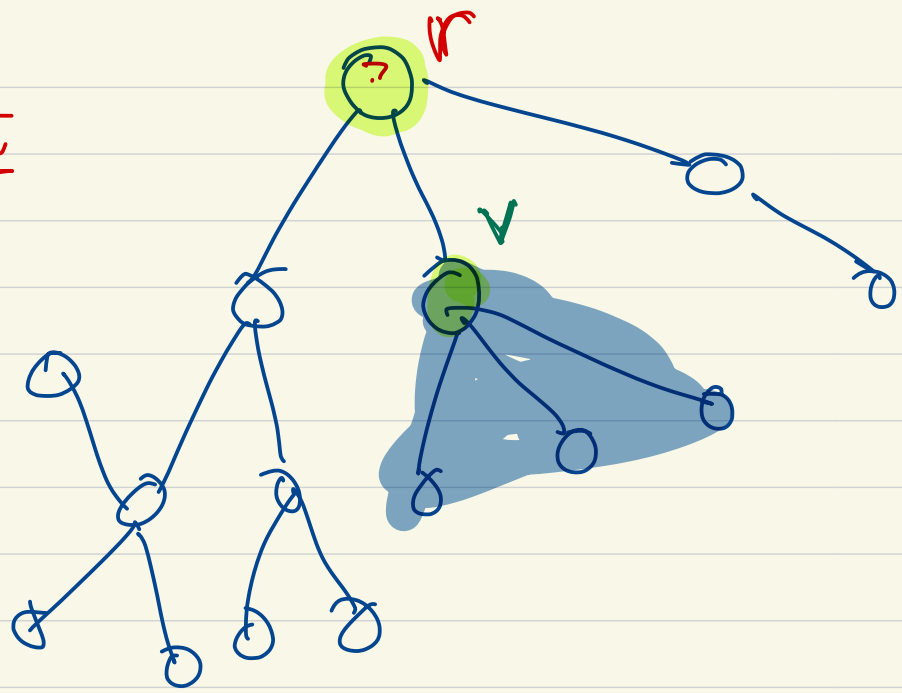
Defn: A set S is an independent set if NO EDGES inside S. i.e. $\forall u, v \in S$ $(u, v) \notin E$

GOAL: Find the largest independent set.

GOAL: A $\Theta(|V| + |E|)$ for Maximum independent set ON TREES

$G = (V, E)$ is a tree,

- Assign some vertex r = root



$\forall$ vertices v,

T_v = subtree hanging at v

SUBPROBLEM:

$I(v)$ = size of the largest independent set in subtree T_v {hanging at v}

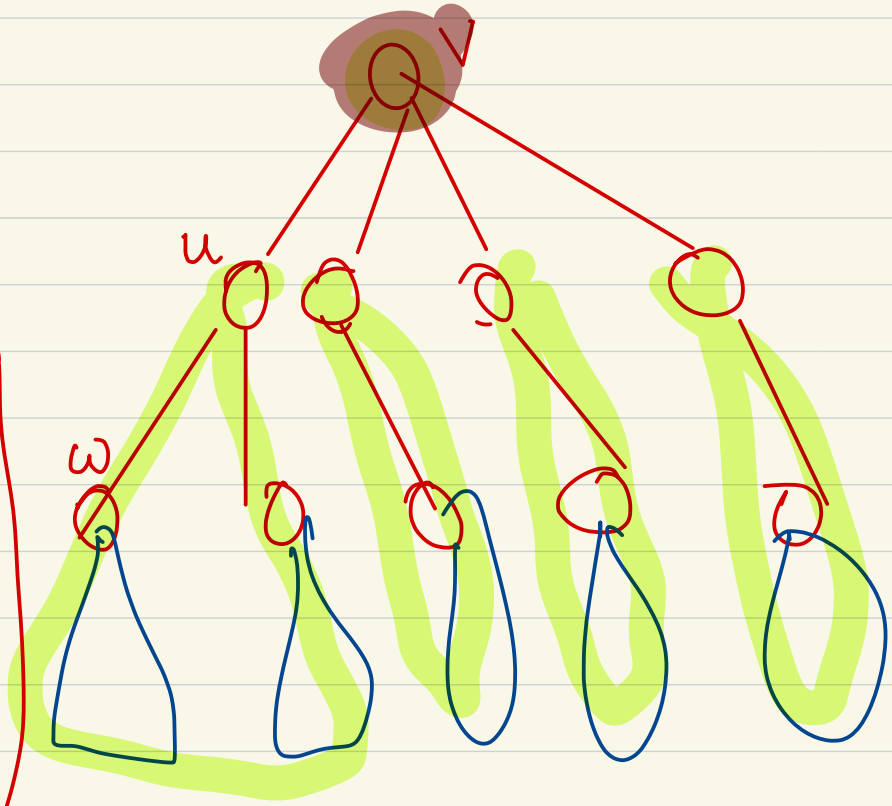
$I(\text{root}) = ??$

SUBPROBLEM:

$I(v)$ = size of the largest independent set in subtree T_v {hanging at v}

RECCURENCE

$$I[v] = \max \begin{cases} \text{Case 1: largest ind set includes } v \\ 1 + \sum_{w \in \text{grandchild}} I(w) \\ \text{Case 2: doesn't include } v \\ \sum_{\text{children } u} I(u) \end{cases}$$



ORDER: Bottom up on the tree.

Implementing via DFS:

```
explore (v) {
    visited[v] = TRUE
    for each edge (v,w) ∈ E
        if visited[w] = FALSE   explore(w)
}
```

$$I[v] = \max \begin{cases} \text{Case 1: largest ind set includes } v \\ 1 + \sum_{w \in \text{grandchild}} I(w) \\ \text{Case 2: doesn't include } v \\ \sum_{\text{children } u} I(u) \end{cases}$$

This Code is Incomplete

DOES NOT HANDLE BASECASE.