Lecture 13

CS 170

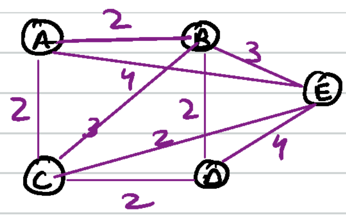
Sanjam Garg.

DP Approach

1) Define appropriate subproblems
2) Write a recurrence relation
3) Determine order of computation.
   ↳ induces a DAG structure

Travelling Salesman Problem.

Input: n cities & distances d_{ij} $(i \neq j)$
Goal: Find minimum distance path from city 1 back to city 1 visiting each city *exactly* once.



Brute-force: (i) $1 \to 2 \to 3 \cdots \to n$
(ii) $1 \to 3 \to 2 \to 4 \cdots \to n$
⋮
$n! \approx n^n \to$ costly.

$n!$

$$C(j) = \text{cost of minimum path from } 1 \to j$$

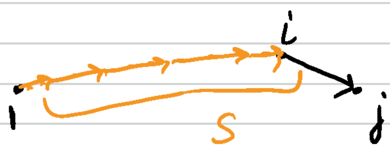
Simplification: TS can end in any of the n cities.

$C(S, j) \longrightarrow$ the least cost path that
① starts at 1
② visits all nodes in set S (exactly once)
③ ends in node j.

$2^n \times n$

$S \subseteq \{1 \ldots n\}$
s.t $j, 1 \in S$

For $|S| > 2$



$$C(S, j) = \min_{i \in S \setminus \{1, j\}} \{ C(S \setminus \{j\}, i) + d_{ij} \}$$

Base Case: $|S| = 2$ $\quad \forall j \neq 1 \quad C(S, j) = d_{1j}$
$S = \{1, j\}$

$$C(\{1\}, 1) = 0$$

$$C(S, 1) = \infty \quad \text{for all } |S| \geq 2$$

$$|S| \geq 2 \;\&\; j \neq 1 \quad C(S, j) = \min_{i \in S \setminus \{j\}} C(S \setminus \{j\}, i) + w_{ij}$$

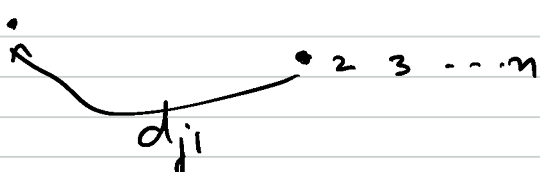
$$|S| > 2 \quad C(S, j) = \min_{i \in S \setminus \{j\}} \{ C(S \setminus \{j\}, i) + d_{ij} \}$$

$$\underline{C(S \setminus \{j\}, 1)}$$

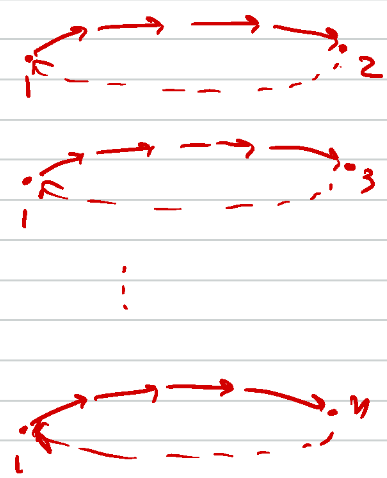
$$|S| = 2 \quad C(S, j) = C(S \setminus \{j\}, 1) + d_{1j}$$

Finding TS solution

$$\min_{j} C(\{1 \cdots n\}, j) + d_{j1}$$



1 2 3 4 5 6
1 3 2 4 5 6



Algorithm

$C(\{1\},1) = 0$

$O(n) \rightarrow$ for $s = 2$ to n

$O(2^n) \rightarrow$ for all subsets $S \subseteq \{1 \dots n\}$ of size s and containing 1:

$C(S,1) = \infty$

for all $j \in S$, $j \neq 1$

$O(n) \rightarrow$ $C(S,j) = \min\limits_{\substack{i \in S \\ i \neq j}} \{ C(S - \{j\}, i) + d_{ij} \}$

return $\min\limits_{j} \{ C(\{1 \dots n\}, j) + d_{j1} \}$

$2^n \times n \times n \rightarrow O(2^n n^2)$

Independent Set

Definition: Graph $G = (V,E)$, an independent set is a set $I \subseteq V$ such that $\forall\, u,v \in I$ we have that $(u,v) \notin E$.

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

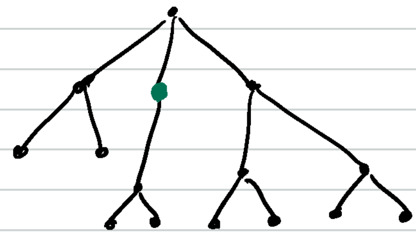
Find: $Ind(G) = \max\limits_{I \subseteq V} \{ |I| : I \text{ is an independent set} \}$

↳ is hard in general!

Maximal Independent Set for Tree

Given : Tree $G = (V, E)$

Find : $Ind(G) = \max_{I \subseteq V} \{ |I| : I \text{ is an independent set} \}$



$I(v) =$ size of the maximal independent set of the sub-tree rooted at v

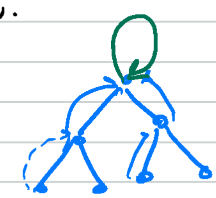
$$I(v) = \max \left\{ \sum_{u \in C(v)} I(u) \; , \; 1 + \sum_{u \in G(v)} I(u) \right\}$$

$C(v)$ ↳ children of v

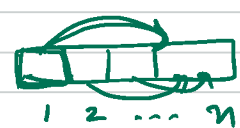
$G(v)$ ↳ grand-children

Base Case: $I(v) = 1$ if v is a leaf (a node with no children)

③ Order of computation : from leaves to the root.



Algorithm



Input: Tree on $V = \{1 \ldots n\}$
$\pi(v) =$ parent of v , $\pi(r) = r$

linear ← Topologically sort V (s.t. $\pi(v) > v \;\; \forall v \neq r$)

linear ← For $v \in V$ if $\pi(v) \neq v$ then add v to $C(\pi(v))$
& if $\pi(\pi(v)) \neq \pi(v)$ then add v to $G(\pi(\pi(v)))$

linear ← For $v \in V$ if $C(v) = \phi$ then $I(v) = 1$

For all $v \in V$

linear ← $I(v) = \max \left\{ \sum_{w \in C(v)} I(w) \; , \; 1 + \sum_{w \in G(v)} I(w) \right\}$

Output $I(r)$

$O(n)$

Knapsack Problem

Input: Total weight capacity of a knapsack W
List of n items with weights $w_1 \quad w_2 \quad \dots \quad w_n$ (integers)
values $v_1 \quad v_2 \quad \quad v_n$ (integers)

Goal: Find set of items that will maximize total value
with total weight $\leq W$

Example:

Item	Weight	Value
1	6	\$30
2	3	\$14
3	4	\$16
4	2	\$9

$W = 10$

Two variations: with repetition & without repetition

Without replacement: 1 & 3 → \$46

With replacement: 1 & 4 & 4 → \$48

Algorithm for the with replacement case.

Input: W, $v[1 \dots n]$, $w[1 \dots n]$

$K(0) = 0$
For $C = 1$ to W

$$K(C) = \max_{i : w_i \leq C} \{ v_i + K(C - w_i) \}$$

Output $K(W)$

→ $O(nW)$

what if all w_is are a multiple of 1000?

Memoization

Initialize: hash table, initially empty, holds $K(w)$ indexed by w

Knapsack(w)

if w is in hash table: return $K(W)$

$K(w) = \max_i \{ \text{knapsack}(w - w_i) + v_i : w_i \le w \}$

insert $K(w)$ into hash table with key w

return $K(w)$.

Coin denomination problem

Input: $x_1 \dots x_n$; V

Goal: Possible to make change of V using given coins?
Find: minimum # of coins needed.

Example: $5, 10; V = 15$ $\qquad 5, 10; V = 12$

$K(v)$ = minimum number of coins needed to give change for v (we set this to ∞ if no possible solution)

$$K(v) = \min \left\{ \min_{i : x_i \le V} \{ K(V - x_i) + 1 \}, \infty \right\}$$

Base Case: $K(0) = 0$

Input: V, $x_1 \ldots x_n$

$K(0) = 0$

For $C = 1$ to V $\longrightarrow O(V)$

$$K(C) = \min\left\{\infty, \min_{i : x_i \le C} \{1 + K(C - x_i)\}\right\} \longrightarrow n$$

Output $K(W)$

$\longrightarrow O(Vn)$