KAIST, School of Computing, Spring 2024 **Scribed By: Eunjung KIM**
Algorithms for NP-hard problems (CS492) **Lecture #13**
Lecture: Eunjung KIM **9 April 2024**

Contents

1 Dynamic programming

If a problem can be optimally solved by combining the solutions to a smaller problem, then dynamic programming approach can be used. We give two dynamic programming algorithm, one for TRAVELLING SALESMAN PERSON and another for STEINER TREE. Both runs in time $2^n \cdot n^{O(1)}$ and requires exponential space.

> TRAVELLING SALESMAN PERSON
> **Instance:** a complete graph $G = (V, E)$ with distance $d : \binom{V}{2} \to \mathbb{R}_{\geq 0}$
> **Question:** find a closed tour of minimum total distance visiting every vertex precisely once.

For a graph $G = (V, E)$, and a vertex subset $K \subseteq V$, a *steiner subgraph* for K is a connected subgraph H of G which contains all vertices of K. Intuitively, a steiner subgraph for K is an essential structure in G that pairwise connect the vertices of K The vertices of K are called *terminals*. For a subgraph H of an edge-weighted graph G with weight function $\omega : E \to \mathbb{R}_{\geq 0}$, the *weight of* H is the sum $\sum_{e \in E(H)} \omega(e)$ over all H's edges and will be denoted by $\omega(H)$. In this vein, we are interested in finding a steiner subgraph of minimum weight. With non-negative weights, a steiner subgraph of minimum edge count/weight sum can be assumed to be a tree and we call a steiner subgraph which is a tree a *steiner tree*, or K-steiner tree to emphasize the terminal set that the tree covers. This leads to the following fundamental problem.

> STEINER TREE
> **Instance:** an edge-weighted graph $G = (V, E)$ with weight function $\omega : E \to \mathbb{R}_{\geq 0}$, and a set of vertices $K \subseteq V$ (terminals)
> **Question:** find a K-steiner tree of minimum weight, if one exists.

1.1 DP for TRAVELLING SALESMAN PERSON

Fix a vertex s. For all subsets $s \in S \subseteq V$ and a vertex $v \in S$, we compute the value $P[S, v]$ of a minimum distance (s, v)-path in $G[S]$ visiting every vertex exactly once. Note that the value of a minimum distance closed tour in G visiting every vertex once equals

$$\min\{P[S, v] + d(v, s) : v \in V\}.$$

The base case is when $S = \{s\}$ and $v = s$, and we have $P[S, s] = 0$ trivially. For sets S containing s with $|S| \geq 2$, the next recursion for $P[S, v]$ is easy to see.

$$P[S, v] = \begin{cases} 0 & \text{if } v = s \\ \min\{P[S \setminus v, w] + d(w, v) : w \in S \setminus v\} & \text{if } v \neq s. \end{cases}$$

Each computation of $P[S, v]$ requires $O(|S|)$ look-ups of the table P constructed for sets of size $|S| - 1$. As there are $2^{n-1} \cdot n$ entries in the table, the algorithm takes $O(2^n \cdot n^2)$-time.

The above recursive formula and the resulting algorithm computes the value of an optimal TSP tour. How can we find a tour whose total distance equals the determined value?

1.2 DP for STEINER TREE

Assumption. If $|K| \leq 2$, then STEINER TREE has a trivial solution; if $|K| = 1$, a trivial steiner tree consisting of a a singleton is an optimal solution and a steiner tree which is a shortest path between two terminals is an optimal solution for the case when $|K| = 2$. Therefore, we assume $|K| \geq 3$. Moreover, G can be assumed to be connected; if K resides in more than one connected components of G, there is no K-steiner tree and we report so. If this is not the case, we can take as the input graph the unique connected component of G containing the entire set K.

Notations. For all subsets $\emptyset \neq K' \subseteq K$ and $v \in V$, let $t(T', v)$ be the weight of an optimal $(K' \cup v)$-steiner tree which takes v as a leaf. Note that any leaf of an optimal K-steiner tree can be assumed to be a terminal (i.e. a vertex in K) since otherwise one can remove a non-terminal leaf and get a K-steiner tree of less or equal weight. Therefore, the weight of an optimal K-steiner tree equals $\min\{t(K, s) : s \in K\}$. Figuring out how to construct an optimal steiner tree achieving this minimum weight is left to the readers as an exercise. We denote by $\mathsf{dist}(u, v)$ the total weight of a shortest (with respect to ω) (u, v)-path.

For $|K' \cup v| \leq 2$, the value of $t(K', v)$ equals 0 if $K' = \{v\}$ and $\mathsf{dist}_G(u, v)$ when $K' \cup v = \{u, v\}$. Therefore, we consider the case when $|K' \cup v| \geq 3$ and notice that $|K'| \geq 2$ in this case. We want to compute the table entry $t(K', v)$.

Recursive formula. We prove that the following recursive formula holds for all subsets $\emptyset \neq K' \subseteq K$ and $v \in V$ with $|K' \cup v| \geq 3$.

$$(\star) \qquad t(K', v) = \min_{\substack{z \neq v \\ K^1 \uplus K^2 = K' \setminus v \\ K^i \neq \emptyset, i=1,2}} t(K^1, z) + t(K^2, z) + \mathsf{dist}_G(z, v)$$

Lemma 1. $t(K', v) \leq \min_{\substack{z \neq v \\ K^1 \uplus K^2 = K' \setminus v \\ K^i \neq \emptyset, i=1,2}} t(K^1, z) + t(K^2, z) + \mathit{dist}_G(z, v)$

Proof: Let T_i be a $(K^i \cup z)$-steiner tree having z as a leaf of minimum weight for $i = 1, 2$, and note that $\omega(T_i) = t(K^i, z)$. Let P be a shortest (z, v)-path of G. Then the graph $H := T_1 \cup T_2 \cup P$, i.e. the subgraph of G whose vertex set is $V(T_1) \cup V(T_2) \cup V(P)$ and takes $E(T_1) \cup E(T_2) \cup E(P)$ as an edge set, is a steiner subgraph for $K^1 \cup K^2 \cup \{z, v\}$ and thus for $K' \cup \{v\}$. It suffices to observe that $\omega(H) = t(K^1, z) + t(K^2, z) + \mathsf{dist}_G(z, v)$ and H contains a $(K' \cup v)$-steiner tree T of weight at most $\omega(H) = t(K^1, z) + t(K^2, z) + \mathsf{dist}_G(z, v)$ such that v is a leaf of T. □

Lemma 2. $t(K', v) \geq \min_{\substack{z \neq v \\ K^1 \uplus K^2 = K' \setminus v \\ K^i \neq \emptyset, i=1,2}} t(K^1, z) + t(K^2, z) + \mathit{dist}_G(z, v)$

Proof: Let T be an optimal $(K' \cup v)$-steiner tree in which v is a leaf which attains the total weight $t(K' \cup v)$. Let P be a maximal path in T such that one endpoint of P is v and none of the internal vertices of P is in K' and none of them is a branching vertex of T (i.e. degree at least three in T). Let z be the endpoint of P other than v. Such z is well-defined; indeed, choose a terminal vertex $z' \in K' \setminus v$ which is closest to v in T. If the (z', v)-subpath P' of P has no branching vertex of T as an internal vertex, then we take $z := z'$. Otherwise, choose a branching vertex on P' closest to v and take it as z. Clearly, any internal vertex of (z, v)-path is neither a terminal nor a branching vertex of T.

Notice that z is a branching vertex of T or a terminal, and $z \neq v$. There are two cases.

Case 1. z is a branching vertex. Let $T_1, \dots, T_\ell$ be the subtress of T obtained from T by removing all vertices of P. We observe that $\ell \geq 2$ and each subtrees T_i contains at least one terminal. Let T^1 be the subtree of T induced by the vertex set $V(T_1) \cup \{z\}$ and T^2 be the subtree of T induced by the vertex set $\bigcup_{i=2}^{\ell} V(T_i) \cup \{z\}$. Then for both $i = 1, 2$, T^i is a $(K^i \cup z)$-steiner tree with z being a leaf, where $K^i = K' \cap V(T^i)$. Moreover, we have $K^i \neq \emptyset$ for $i = 1, 2$ and (K^1, K^2) forms a bipartition of $K' \setminus v$. Clearly, P is a (z, v)-path of G. Therefore

$$\begin{aligned} t(K', v) = \omega(T) &= \omega(T^1 \cup T^2 \cup P) = \omega(T^1) + \omega(T^2) + \omega(P) \\ &\geq t(K^1, z) + t(K^2, z) + \mathsf{dist}_G(z, v) \\ &\geq \min_{\substack{z \neq v \\ K^1 \uplus K^2 = K' \setminus v \\ K^i \neq \emptyset, i=1,2}} t(K^1, z) + t(K^2, z) + \mathsf{dist}_G(z, v) \end{aligned}$$

which settles the inequality.

Case 2. z is not a branching vertex. Note that z is a terminal in this case. Then let T^1 be the subtree of T obtained by deleting all vertices of P except for z, and let T^2 be the trivial tree consisting of the singleton $\{z\}$. It is straightforward to verify that the inequality holds. □

Algorithm and Runtime. Assuming that $t(K', v)$ have been computed for all $\emptyset \neq K' \subseteq K$ with $|K'| \leq i$ and for all $v \in V$, one can compute $t(K', v)$ for all $\emptyset \neq K' \subseteq K$ with $|K'| = i + 1$ and for all $v \in V$. This is because the computation of $t(K', v)$ requires access only to those entries of the form $t(K'', z)$ with $\emptyset \neq K'' \subsetneq K'$ and $z \in V$ such that $|K''| < |K|$. Therefore we can compute the full table, and in particular determine the weight of an optimal K-steiner tree by computing $\min\{t(K, s) : s \in K\}$.

To see the running time, observe that determining the value of $t(K', v)$ requires to inspect n different choices for z and at most $2^{|K'|}$ different bipartitions of $K' \setminus v$. Therefore, computing $t(K', v)$ takes at most $n \cdot 2^{|K'|}$ arithmetic operations. In total, it takes

$$\text{(Running time of all-pairs shortest paths problem)} + \sum_{i=2}^{|K|} n 2^i = O(n^3) + n 3^{|K|},$$

that is, $O(n^3 + n \cdot 3^{|K|})$-time.

2 Inclusion-Exclusion based algorithms

2.1 Inclusion-Exclusion formula

Theorem 1 (Inclusion-Exclusion, union version)**.** *Let A_i for $i = 1, \dots, n$ be finite sets. Then,*

$$|\bigcup_{i \in [n]} A_i| = \sum_{\emptyset \neq X \subseteq [n]} (-1)^{|X|+1} |\bigcap_{i \in X} A_i|.$$

Proof: Notice that an element not in $\bigcup_{i \in [n]} A_i$ contributes neither to any term of the right-hand side, nor to the left-hand side. For an element $x \in \bigcup_{i \in [n]} A_i$, its contribution to the left-hand side is 1. It remains to show that the sum of contribution of x to the right-hand side is precisely 1. Let $Y \subseteq [n]$ be the set of indices i such that $x \in A_i$. Then for every $\emptyset \neq X \subseteq Y$, $\bigcap_{i \in X} A_i$ contains x. Conversely, for every $\emptyset \neq X \nsubseteq Y$ we have $x \notin \bigcap_{i \in X} A_i$. Therefore, x creates the following terms of the right-hand side:

$$\begin{aligned}
\sum_{\emptyset \neq X \subseteq Y} (-1)^{|X|+1} \cdot 1 &= (-1) \sum_{\emptyset \neq X \subseteq Y} (-1)^{|X|} \\
&= -\sum_{i=1}^{|Y|} \sum_{X \subseteq Y, |X|=i} (-1)^i \\
&= -\sum_{i=1}^{|Y|} \binom{|Y|}{i} (-1)^i 1^{|Y|-i} \\
&= -\Big(\sum_{i=0}^{|Y|} \binom{|Y|}{i} (-1)^i 1^{|Y|-i} - 1 \Big) \\
&= 1 - (-1+1)^{|Y|} = 1.
\end{aligned}$$

□

Theorem 2 (Inclusion-Exclusion, intersection version)**.** *Let A_i for $i = 1, \ldots, n$ be sets of a finite universe U. Then,*

$$|\bigcap_{i \in [n]} A_i| = \sum_{X \subseteq [n]} (-1)^{|X|} |\bigcap_{i \in X} (U \setminus A_i)|.$$

Proof: First, we note that for finite sets B_i, $i \in [n]$,

$$U \setminus \bigcup_{i \in [n]} B_i = \bigcap_{i \in [n]} (U \setminus B_i). \tag{1}$$

Therefore, by Theorem 1 it holds that

$$\begin{aligned}
|U \setminus \bigcup_{i \in [n]} B_i| &= |U| + \sum_{\emptyset \neq X \subseteq [n]} (-1)^{|X|} |\bigcap_{i \in X} B_i| \\
&= \sum_{X \subseteq [n]} (-1)^{|X|} |\bigcap_{i \in X} B_i|.
\end{aligned} \tag{2}$$

Set $A_i = U \setminus B_i$ and combine the equations (1)-(2). Now,

$$\begin{aligned}
|\bigcap_{i \in [n]} A_i| = |\bigcap_{i \in [n]} (U \setminus B_i)| &= |U \setminus \bigcup_{i \in [n]} B_i| \\
&= |U| - \sum_{\emptyset \subsetneq X \subseteq [n]} (-1)^{|X|+1} |\bigcap_{i \in X} B_i| \\
&= \sum_{X \subseteq [n]} (-1)^{|X|} |\bigcap_{i \in X} (U \setminus A_i)|,
\end{aligned}$$

where the last equation follows from the convention of writing $U = \bigcap_{i \in \emptyset} B_i$. □

2.2 IE-based algorithm for Hamiltonian Cycle

Using the Inclusion-exclusion formula we can compute Hamiltonian Cycle in $2^n \cdot n^{O(1)}$-time. In fact we can count the number of Hamiltonian cycles in the same running time.

Let $G = (V, E)$ be on n vertices $v_1, \ldots, v_n$, and let $v_0 = v_n$. A *closed walk* is a sequence of vertices of G whose start and end vertices are identical, and any two consecutive vertices are adjacent in G. Notice that a vertex or an edge might appear in a walk multiple times. The length of a closed walk is the length of vertex sequence minus one. By v_0-walk, we mean a closed walk that begins and ends with v_0. To apply the (intersection version) of inclusion-exclusion formula, we define the ground set U as follows:

$$U = \{\text{all } v_0\text{-walks of length } n\}.$$

Now we can view a Hamiltonian cycle (with an orientation) as a v_0-walk of length n which visits every $v \in V$. Notice that each Hamiltonian cycle yields two v_0-walks of length n visiting every vertex v. Therefore with A_i defined as

$$A_i = \{\text{all } v_0\text{-walks of length } n \text{ visiting } v_i\},$$

the Hamiltonian cycles, the v_0-walks of length n visiting all $v \in V$ to be precise, are captured by $\bigcap_{i \in [n]} A_i$. Its cardinality can be computed by computing $|\bigcap_{i \in X}(U \setminus A_i)|$ for every $X \subseteq [n]$ thanks to Theorem 2.

So, what kind objects constitute $\bigcap_{i \in X}(U \setminus A_i)$? Observe that $U \setminus A_i$ are precisely the v_0-walks of length n which *avoid* v_i, and thus $\bigcap_{i \in X}(U \setminus A_i)$ are v_0-walks of length n which avoid all vertices corresponding to X. In other words, $\bigcap_{i \in X}(U \setminus A_i)$ are the set of all v_0-walks of length n in $G - X$ (formally $G - \{v_i : i \in X\}$).

Finally, the number of (v_i, v_j)-walks of length ℓ in a graph H can be computed in polynomial time by computing ℓ-th power of the adjacency matrix of H and reading off the (i, j)-entry of the resulting matrix. This completes the algorithm and it is straightforward to see that after 2^n steps all the terms of $\sum_{X \subseteq [n]} (-1)^{|X|} |\bigcap_{i \in X}(U \setminus A_i)|$ have summed up. We remark that this algorithm works both for directed and undirected graphs.

2.3 IE-based algorithm for k-Coloring

To apply the intersection version of inclusion-exclusion formula, we view a k-coloring as a k-tuple of independent sets of G. Namely, we define

$$U = \{(I_1, \ldots, I_k) : I_i \text{ is an independent set of } G\}.$$

Notice that two independent sets in a tuple may intersect and even coincide. Observe that there is a (proper) k-coloring if and only if there is k-tuple of independent sets covering all vertices of G. Therefore let

$$A_i = \{(I_1, \ldots, I_k) \in U : v_i \in I_1 \cup \cdots \cup I_k\},$$

and G admits a proper k-coloring if and only if $\bigcap_{i \in [n]} A_i \neq \emptyset$. Due to Theorem 2, we can decide this via computing the value $\sum_{\emptyset \neq X \subset [n]} (-1)^{|X|+1} |\bigcap_{i \in X}(U \setminus A_i)|$.

Again, $\bigcap_{i \in X}(U \setminus A_i)$ is the set of all k-tuples of independent sets avoiding the vertices in X altogether. In other words, it is the set of all k-tuples of independent sets of $G - X$. Let $i(G)$ be the number of independent sets of G and observe

$$|\bigcap_{i \in X}(U \setminus A_i)| = i(G - X)^k.$$

Now $i(G)$ can be computed with dynamic programming. Choose an arbitrary vertex $v \in G$ and note that

$$i(G) = i(G - v) + i(G - N[v])$$

where the first term in r.h.s counts the independent sets of G *not* containing v and the second term counts the independent sets of G containing v, thus excluding $N(v)$. The base case is $i(\emptyset)$ and $i(K_1)$, i.e. an empty graph and a graph on one vertex. The number of independent sets in each case is 1 and 2 respectively. This recursion indicates that $i(G[Z])$ over all subsets Z of V can be tabulated, and this can be done in time $2^n \cdot n^{O(1)}$.

With the above table containing values for $i(G - X)$ for all $X \subseteq [n]$, we can compute

$$\sum_{X \subseteq [n]} (-1)^{|X|+1} |\bigcap_{i \in X} (U \setminus A_i)| = \sum_{X \subseteq [n]} (-1)^{|X|+1} i^k(G - X)$$

in time $2^n \cdot n^{O(1)}$.