



CS240 Algorithm Design and Analysis

Lecture 16

FPT (fixed parameter tractable)

Quan Li
Fall 2024
2024.11.14



Solving NP-hard Problems on Trees

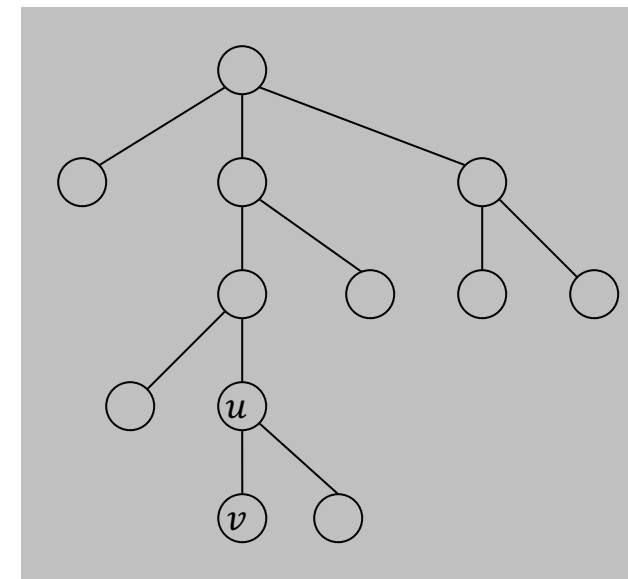




Independent set on trees



- **Problem definition.** Given a **tree**, find a maximum cardinality subset of nodes such that no two share an edge.
- **Def.** A leaf is a node with degree 1.
- **Key observation.** If v is a leaf, there exists a maximum size independent set containing v .
- **Pf.** (exchange argument)
 - Consider a max size independent set S .
 - If $v \in S$, we're done.
 - If $u \notin S$ and $v \notin S$, then $S \cup \{v\}$ is independent $\Rightarrow S$ not maximum.
 - If $u \in S$ and $v \notin S$, then $S \cup \{v\} - \{u\}$ is independent. ■





Independent Set on Trees: Greedy Algorithm



- **Theorem.** The following greedy algorithm finds a maximum cardinality independent set in forests (and hence trees)

Independent-Set-In-A-Forest(F) :

$S \leftarrow \emptyset$

while F has at least one edge **do**

 Let $e = (u, v)$ be an edge such that v is a leaf

 Add v to S

 Delete nodes u and v from F , and all edges
 incident to them.

 Add all remaining vertices to S

return S

- **Pf.** Correctness follows from the previous key observation. ■
- **Remark.** Can implement in $O(n)$ time by considering nodes in postorder

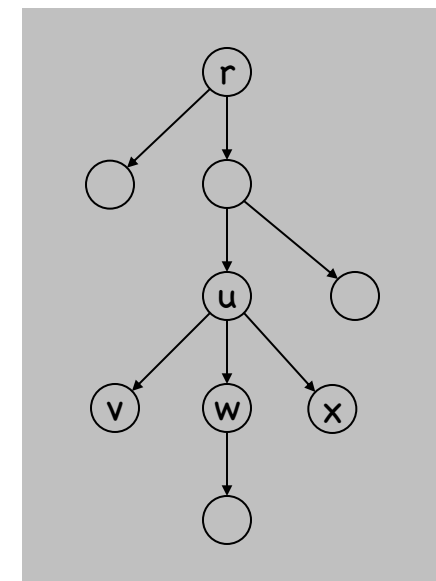




Weighted Independent Set on Trees



- **Problem definition.** Given a tree and node weights $w_v > 0$, find an independent set S that maximizes $\sum_{v \in S} w_v$.
- **Note:** Greedy doesn't work anymore.
- **Observation.** If (u, v) is an edge such that v is a leaf node, then either OPT includes u , or it includes all leaf nodes incident to u .
- **Dynamic programming solution.** Root tree at some node, say r .
 - $OPT_{in}(u)$ = max weight independent set of subtree rooted at u , containing u .
 - $OPT_{out}(u)$ = max weight independent set of subtree rooted at u , not containing u .
- $OPT_{in}(u) = w_u + \sum_{v \in \text{children}(u)} OPT_{out}(v)$
- $OPT_{out}(u) = \sum_{v \in \text{children}(u)} \max\{OPT_{in}(v), OPT_{out}(v)\}$



children(u) = { v , w , x }





Independent Set on Trees: DP Algorithm



- **Theorem.** The dynamic programming algorithm finds a maximum weighted independent set in trees in $O(n)$ time.

can also find independent set itself (not just value)

Weighted-Independent-Set-In-A-Tree (T) :

Root the tree at a node r

for each node u of T in postorder

if u is a leaf

$M_{in}[u] \leftarrow w_u$

$M_{out}[u] \leftarrow 0$

↑
ensures a node is visited after all its children

else

$M_{in}[u] \leftarrow \sum_{v \in \text{children}(u)} M_{out}[v] + w_u$

$M_{out}[u] \leftarrow \sum_{v \in \text{children}(u)} \max(M_{out}[v], M_{in}[v])$

return $\max(M_{in}[r], M_{out}[r])$

- **Pf.** Takes $O(n)$ time since we visit nodes in postorder and examine each edge exactly once. ■

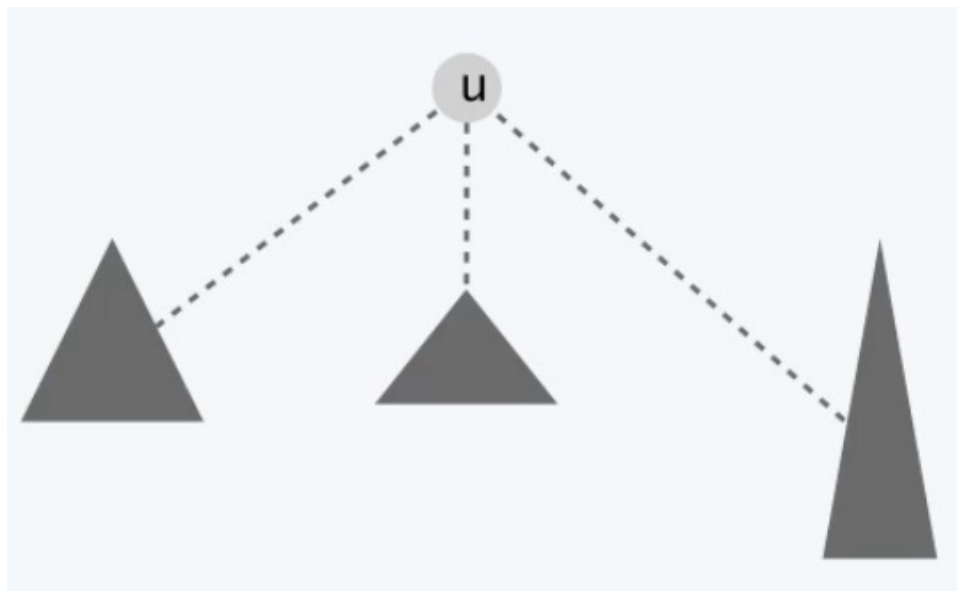




Context



- **Independent set on trees.** This structured special case is tractable because we can find a node that **breaks the communication** among the subproblems in different subtrees

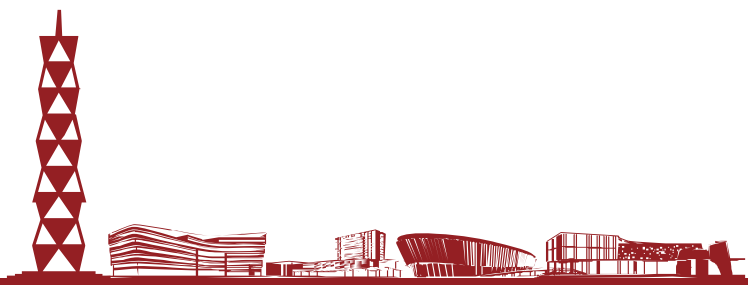


- **Graphs of bounded tree width.** Elegant generalization of trees that:
 - Captures a rich class of graphs that arise in practice
 - Enables decomposition into independent pieces





Circular Arc Covering





Wavelength-Division Multiplexing



Background. More than one communication links can share the same portion of a fiber optic cable, provided they are transmitted using different wavelengths.

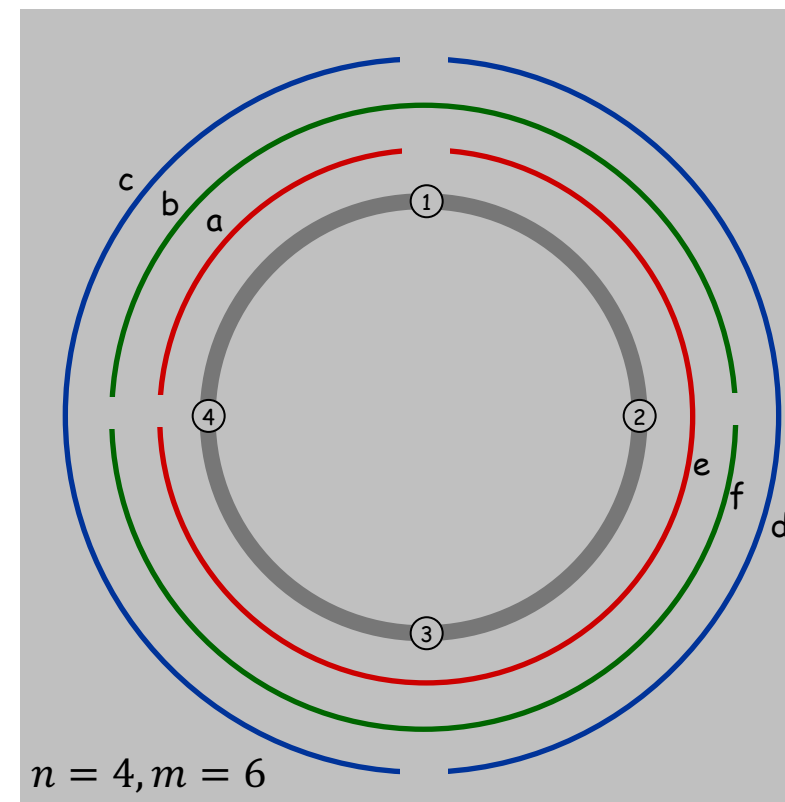
Problem definition. Given a graph $G = (V, E)$, and m paths p_1, p_2, \dots, p_m , assign each path a color so that any two paths that share an edge must have different colors. The goal is to use as few colors as possible.

Ring topology. Consider the special case is when graph is a **cycle** on n nodes.

Bad news. NP-complete, even on rings.

Brute force. Can determine if k colors suffice in $O(k^m)$ time by trying all k -colorings.

Goal. $f(k) \cdot \text{poly}(m, n)$ time.



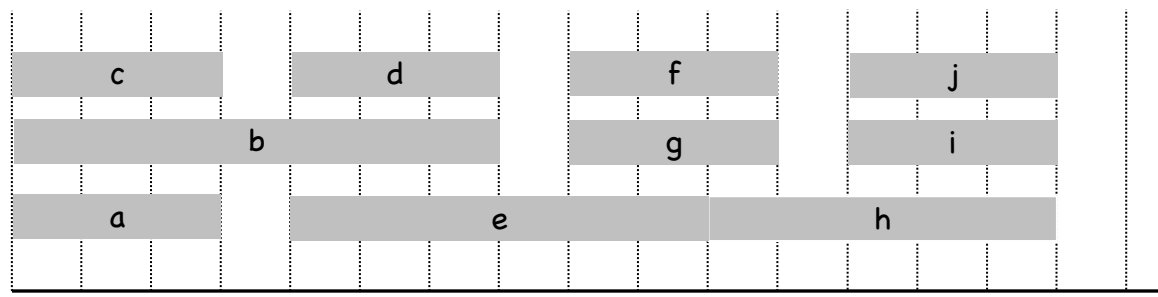


Review: Interval Coloring



Interval coloring. Greedy algorithm finds coloring such that number of colors equals depth of schedule.

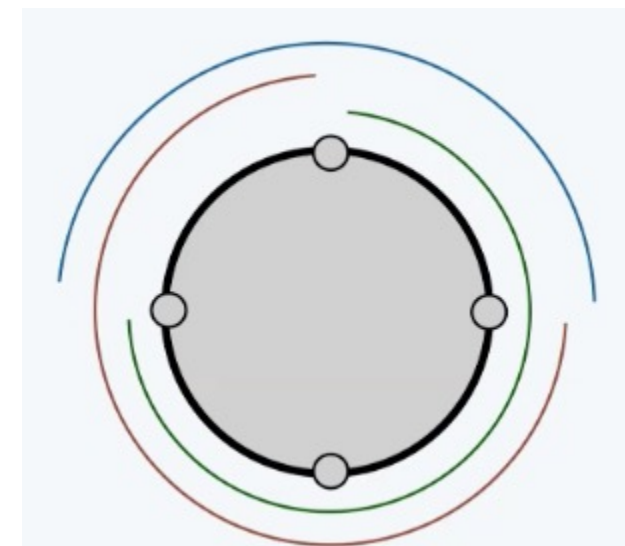
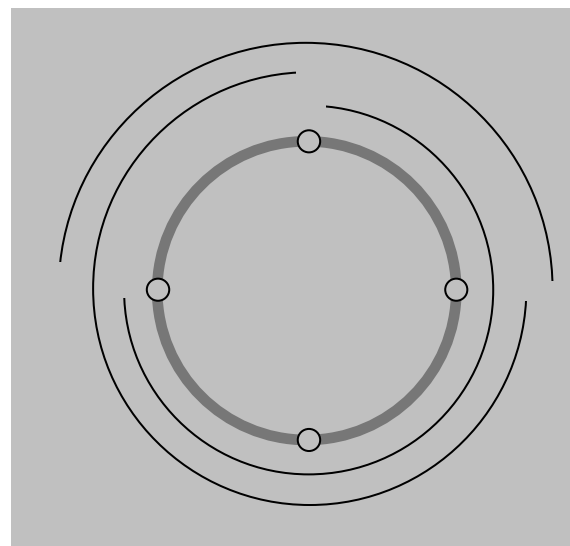
← maximum number of intervals at one location



Circular arc coloring.

- Weak duality: Number of colors \geq depth.
- Strong duality does not hold
- But the two may not be equal.

max depth = 2
min colors = 3

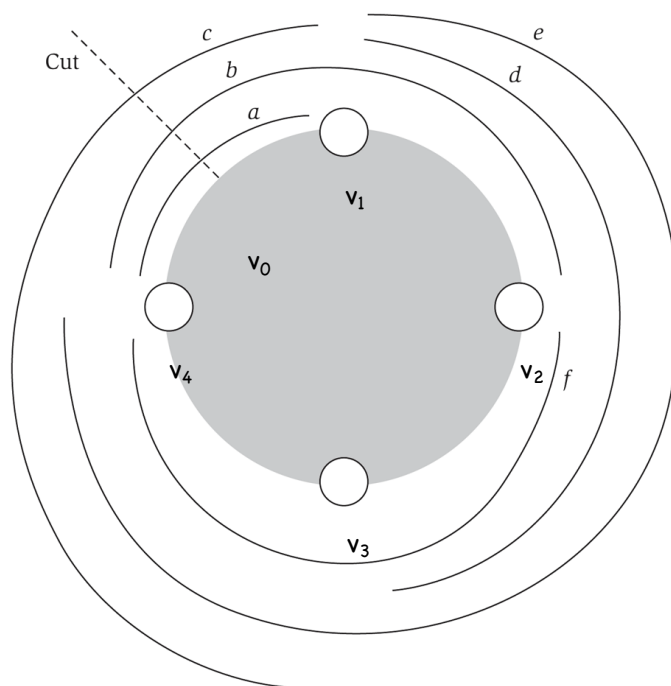




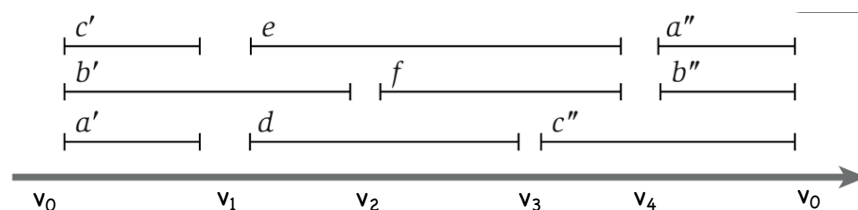
(Almost) Transforming Circular Arc Coloring to Interval Coloring

Circular arc coloring. Given a set of m arcs with depth $d \leq k$, can the arcs be colored with k colors?

Equivalent problem. Cut the ring between nodes v_1 and v_n . The arcs can be colored with k colors iff the intervals can be colored with k colors in such a way that those “sliced” arcs have the same color.



colors of a' , b' , and c' must correspond to colors of a'' , b'' , and c''

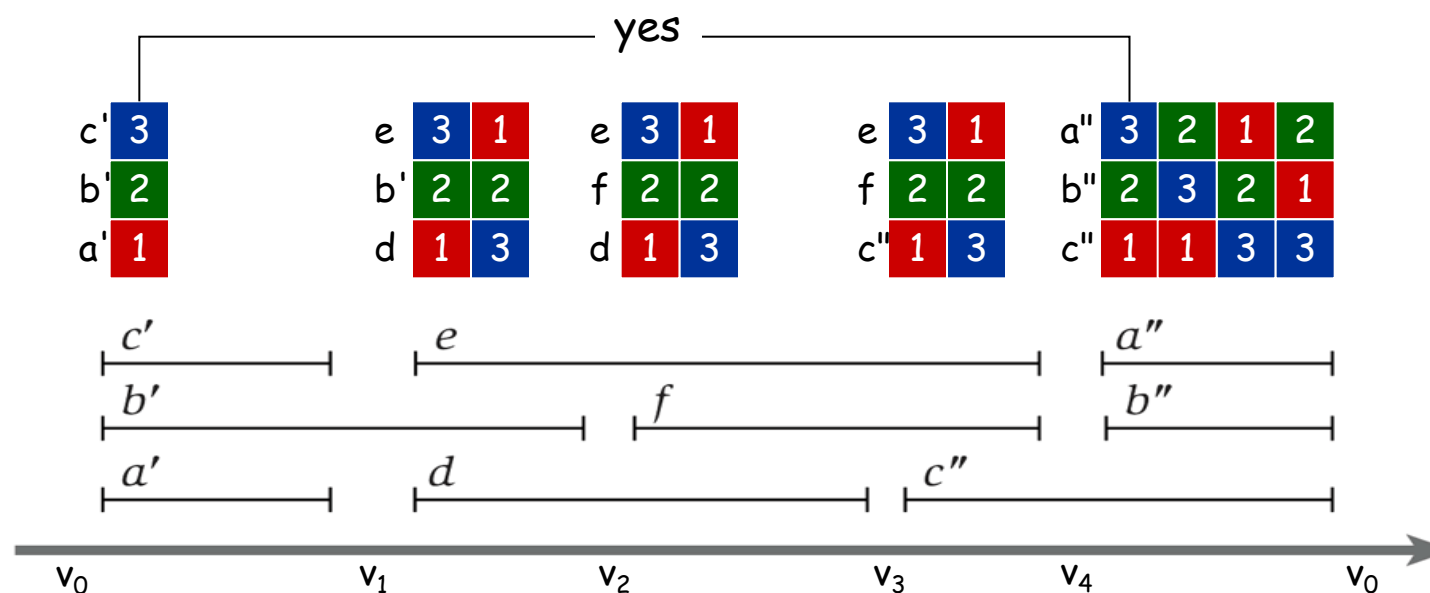




Circular Arc Coloring: Dynamic Programming Algorithm

Dynamic programming algorithm.

- Assign distinct color to each interval which begins at cut node v_0 .
- At each node v_i , some intervals may finish, and others may begin.
- Enumerate all k -colorings of the intervals through v_i that are consistent with the colorings of the intervals through v_{i-1} .
- The arcs are k -colorable iff some coloring of intervals ending at cut node v_0 is consistent with original coloring of the same intervals.





Circular Arc Coloring: Running Time



Running time. $O(k! \cdot n)$

- n phases of the algorithm.
- Bottleneck in each phase is enumerating all consistent colorings.
- There are at most k intervals through v_i , so there are at most $k!$ colorings to consider.

Remark. This is $\text{poly}(n)$ time if $k = O(\log n / \log \log n)$

This algorithm is practical for small values of k (say $k = 10$) even if the number of nodes n (or paths) is large

