Simon Ibssa
CSC 349

Quiz 3 — Greedy Algorithms

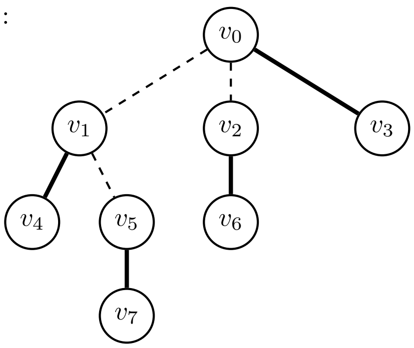
You have **1 day** to complete this quiz. Please ask me if you have any questions. Good luck!

- There are 2 pages in this quiz. Make sure you have all of them.
- This quiz is open-note and open-book, however, all answers must be an individual effort.

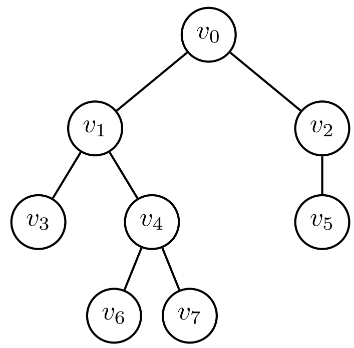
1. Given an undirected graph, a *matching* is a subset of its edges such that no two edges share an endpoint. A matching is said to be *perfect* if it covers all of the vertices in the graph.

   Suppose that we restrict this problem to trees: graphs that are connected and acyclic. For example:

T_1:



T_2:



   Here, the tree T_1 has a perfect matching, composed of the edges $\{(v_0, v_3), (v_1, v_4), (v_2, v_6), (v_5, v_7)\}$. There does not, however, exist any perfect matching for T_2.

   (a) Give pseudocode for an efficient greedy algorithm that, given a tree, constructs a perfect matching or determines that no such matching exists.

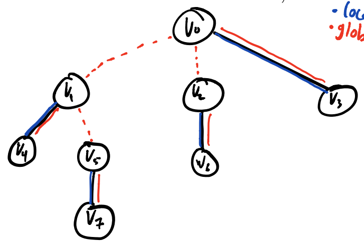
constructMatch (Tree T)
Input: A tree that is connected and acyclic.
Output: A perfect match, or null if it does not exist in T.

let Q = (V, ∅) be a subgraph of T.
let discovered = []
while E ≠ O do :
  let e be an edge connecting two vertecies, v and u, where u does not have children (v is the parent of u)
  if v in discovered, or u in discovered, then
    Add dashed edge e to G.
    Continue
  else, do:
    Remove e from T
    Add e to Q
    discovered.add (v)
    discovered.add (u)
if Q.size == 0, then
  return null   // returns null if no match
else do :
  return Q

(contined from the previous page)

(b) Briefly justify the correctness of your algorithm: identify the locally optimal choices it makes, and explain why those choices will always lead to a globally optimal solution.

[*Hint:* In this problem, the optimized quantity is the *number* of vertices covered by the matching: for correctness, all vertices must be covered.]



- locally optimal solutions in blue
- globally optimal solution in Red.

- constructMatch, by choosing edges w/ vertecies that are undiscovered and does not have grandchildren will ensure that the edge does not share any discovered endpoints.
- Since constructMatch loops over all vertecies of T, checking for this property, the algorithm optimizes for the number of edges with unique endpoints (a match), and correctly finds all vertecies.

$\therefore$ constructMatch correctly greedily chooses a locally optimal outcome that leads to a globally optimal solution (definition of Greedy Algorithm).

(c) Give the time complexity of your algorithm: Master thm:

$T(n) = a \cdot T(n/b) + O(g(n))$

$a = 0$

$b = 0 \quad d = 1$

$T(n) = 0 + O(n)$

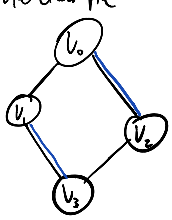
$\therefore T(n) = O(n^1) = O(n)$

$= \boxed{O(|V|), \text{ where } |V| \text{ is the number of vertecies in tree } T.}$

2. Suppose that the problem was not restricted to trees. Give a counterexample — an undirected graph that need not be connected and acyclic — where your greedy algorithm fails to construct a perfect matching, even though such a matching exists.

Counterexample:

$G =$



- perfect match in blue

- Suppose there exists a graph G that is connected and cyclic.
- There exists two perfect match edges in G $\{(V_0, V_2), (V_1, V_3)\}$
- Since there does not exist a neighbor of a vertex v with no children, constructMatch will incorrectly excludes all edges.
- Therefore, because constructMatch does not correctly return $\{(V_0, V_2), (V_1, V_3)\}$, the greedy algorithm fails when given a cyclic graph.